ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická
Katedra elektroniky a informačních technologií

DIPLOMOVÁ PRÁCE

SW řízení kolejiště

Autor práce: **Bc. Tomáš Nejedlo**
Vedoucí práce: **Ing. Petr Weissar, Ph.D.**

2023

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Tomáš NEJEDLO**
Osobní číslo:	**E21N0039P**
Studijní program:	**N0714A060013 Elektronika a informační technologie**
Specializace:	**Elektronika**
Téma práce:	**SW řízení kolejiště**
Zadávající katedra:	**Katedra elektroniky a informačních technologií**

Zásady pro vypracování

1. Analyzujte stávající HW a SW řešení kolejiště KEI.
2. Navrhněte možné způsoby řízení jednotlivých částí.
3. Navrhněte vhodné uživatelské rozhraní.
4. Navržené realizujte.

Rozsah diplomové práce: **40 – 60**
Rozsah grafických prací:
Forma zpracování diplomové práce: **elektronická**

Seznam doporučené literatury:

1. Yiu, Joseph. The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors, Elsevier Science & Technology, 2013.

Vedoucí diplomové práce: **Ing. Petr Weissar, Ph.D.**
Katedra elektroniky a informačních technologií

Datum zadání diplomové práce: **7. října 2022**
Termín odevzdání diplomové práce: **26. května 2023**

L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Doc. Ing. Jiří Hammerbauer, Ph.D.
vedoucí katedry

V Plzni dne 7. října 2022

Abstrakt

Náplní této diplomové práce je vytvoření nové funkční desktopové aplikace, která umožní řízení modelového kolejiště. V teoretické části je rozebráno řízení modelového kolejiště, sběrnice CAN bus a protokol DCC. V následující části je nejprve rozebrán aktuální stav jednotlivých jednotek umožňujících provoz kolejiště a popis původního softwaru včetně jeho kladů a záporů. V neposlední řadě následuje praktická část diplomové práce, ve které je představena logika způsobu řízení modelového kolejiště a tři aplikace, pomocí kterých lze modelové kolejiště ovládat. Cílem této práce bylo vymyslet a vyvinout aplikaci, která umožní plně manuální ovládání kolejiště a zároveň vyvinout aplikaci, která je v autonomním režimu sama schopna řídit modelové kolejiště a plnit funkci zabezpečení, díky kterému nemůže na modelové železnici nastat kolize.

Klíčová slova

Modelové kolejiště TT, DCC signál, digitální řízení, CAN bus, TCP server, řízení kolejiště, Visual Studio, programování v C#, vývoj desktopové aplikace

Abstract

The aim of this diploma thesis is to create a new functional desktop application that will enable the control of the model railway. The theoretical part discusses the control of the model railway, the CAN bus and the DCC standard protocol. In the following section, the current state of the individual units enabling the operation of the railway is first analysed, followed by a description of the original software, including its pros and cons. Last but not least, the practical part of the diploma thesis follows, in which the logic of the model railway and three applications by which the model railway can be controlled are introduced. The aim of this diploma thesis was to invent and develop an application that will allow fully manual control of the railway and at the same time to develop an application that will be able to control the model railway and perform the function of security in autonomous mode, thanks to which no collision can occur on the model railway.

Key Words

Model railway TT, DCC signal, digital commands, CAN bus, TCP server, railway control, Visual Studio, programming in C#, development of desktop application

Poděkování

Rád bych v této části vyjádřil mé hluboké poděkování vedoucímu diplomové práce Ing. Petru Weissarovi, Ph.D. za pomoc a odborné rady a připomínky při řešení diplomové práce.

Mé poděkování patří i mé rodině, která za mnou po celou dobu studia vždy stála a plně mě podporovala. V neposlední řadě bych rád poděkoval i své drahé přítelkyni.

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

Značka	Význam	Český překlad
ACK	Acknowledge Field	Bity pro potvrzení správnosti přijímače
A/D	Analog/Digital Converter	Analogově digitální převodník
CAN	Controller Area Network	Sdílená datová sběrnice
CAN FD	CAN with Flexible Data-Rate	CAN s flexibilní přenosovou rychlostí
CRC	Cyclick Redundancy Code	Cyklický redundantní kód
csv	Comma-separated values	Hodnoty oddělené čárkami
DCC	Digital Command Control	Digitální příkazy pro ovládání
DLC	Data Length	Počet přenášených bytů
ERC	End of CRC	Konec cyklického redundantního kódu
ID	Identifier	Identifikátor
kbit/s	Kilobit per second	Kilobit za sekundu
IDE	Identifier Extension	Rozšíření identifikátoru
JSON	JavaScript Object Notation	Objektový zápis JavaScriptu
LED	Light-Emitting Diode	Elektroluminiscenční dioda
ms	Milliseconds	Milisekundy
nA	Nanoampere	Nanoampér
NMRA	National Model Railroad Association	Národní asociace modelů železnic
PC	Personal computer	Počítač
PWM	Pulse Width Modulation	Pulzně šířková modulace
R	Read	Čtení
RJ-45	Registered Jack-45	Typ konektoru
RTR	Remote Transmission Request	Bit značící žádost o přenos dat
Rx	Reciever	Přijímač
s	Second	Sekunda
SOF	Start Of Frame	Start bit
TCP	Transmission Control Protocol	Protokol kontroly přenosu
TT	Table Top	Velikost železnice v poměru 1:120
TTL	Transistor-transistor-logic	Tranzistorová logika
Tx	Transmitter	Vysílač
USB	Universal Serial Bus	Univerzální sériová sběrnice
UTP	Unshielded Twisted Pair	Nestíněná kroucená dvoulinka
V	Volt	Volt
W	Write	Zápis
XOR	eXclusive OR	Exkluzivní disjunkce
Ω	Ohm	Velikost odporu
µs	Microsecond	Mikrosekunda

Úvod

Tato diplomová práce je zaměřena na modelové kolejiště, které se nachází na Fakultě elektrotechnické Západočeské univerzity v Plzni. První část diplomové práce je zaměřena na teoretickou část problematiky, ve které je popsáno modelové kolejiště, CAN sběrnice a řízení kolejiště digitálním signálem dle DCC standardu. V druhé části je nejprve provedena analýza aktuálního hardwaru a původní softwarové aplikace, a poté vytvořen návrh pro vytvoření desktopové aplikace a logiku řízení, která zajistí bezproblémový provoz kolejiště a bude tvořit jeho zabezpečení před kolizemi.

Důvodem vzniku této diplomové práce bylo vytvoření a připojení nových jednotek kolejových úseků a jednotek pro výhybky, což nyní umožňuje komplexní provoz na kolejišti oproti provozu, který bylo možné provozovat na kolejišti v době původní aplikace. Původní aplikace byla napsána v roce 2018 a byla vytvořena na původní hardware a na obsluhu kolejiště o pouhých 8 úsecích, na kterých probíhal provoz na vzájemně oddělených okruzích.

Cílem této diplomové práce je zanalyzování aktuálního stavu kolejiště a následně navrhnout a vytvořit aplikaci, která umožní jednak plně manuální řízení, při kterém celé kolejiště ovládá člověk, a zároveň částečně manuální a autonomní provoz na kolejišti, při kterém se aplikace sama stará o zabezpečovací logiku a řídí kolejiště způsobem, při kterém je umožněn bezpečný provoz.

Realizace řízení kolejiště je nicméně náročná z důvodu neznámé přesné polohy vlaku a úseku, na kterém se vlak nachází za jízdy. Jediné, z čeho lze tedy vytvořit zabezpečovací logiku je snímání odběrů proudů v jednotlivých izolovaných úsecích a z informace o tom, zdali vlak je v pohybu a jede vpřed či vzad. Z tohoto důvodu byla navržena logika, která zpracovává odebírané odběry proudů v jednotlivých úsecích a na jejich základě je lokálně aktualizovaná poloha vlaků.

1 Modelové kolejiště

1.1 Struktura modelového kolejiště

Jedná se o modelové kolejiště v měřítku TT, které je charakteristické svým standardním měřítkem 1:120 a má za cíl simulovat reálný provoz na železnici včetně řízení točny, řízení návěstidel, či výhybek. Toto modelové kolejiště se skládá ze tří stanic, které jsou pracovně nazvány Beroun, Karlštejn a Lhota. [1]

Beroun je největší z použitých stanic a obsahuje jedenáct samostatných kolejí s oddělenými úseky. V této stanici se dále nachází i točna a depo pro lokomotivy. I s ohledem na nejobsáhlejší počet oddělených kolejí je tato stanice nejvíce uzpůsobena rozsáhlému řízení kolejiště.

Druhá stanice, Karlštejn, je menší než výše zmíněná stanice Beroun a je tvořena šesti samostatnými kolejemi s oddělenými průjezdnými kolejovými úseky a jedním odstavným úsekem.

Poslední ze stanic je stanice Lhota. Jedná se o nejmenší ze stanic a je tvořena pouze třemi kolejovými úseky. Tato stanice není součástí opakujícího se okruhu, a proto musí tuto stanici opouštět vlaky v opačném směru, než v kterém do této stanice přijíždí. Z této stanice je možné se dostat i testovací okruh sloužící k otestování vlakových jednotek a elektroniky.

Mezi stanicemi Beroun a Karlštejn je umožněn dvoukolejní provoz, díky čemuž provoz mezi těmito stanicemi nejvíce odpovídá provozu na vytížených železničních koridorech. Do každé z těchto stanic jsou vestavěny cesty z obou stran a tím je mezi těmito stanicemi umožněn okružní provoz. Do stanice Lhota vede pouze jediná kolej ze stanice Beroun a je zakončena kusými kolejemi. Z tohoto důvodu není umožněn okružní provoz a jedná se pouze o jednokolejnou trať. Kusé koleje lze nalézt i ve stanicích Beroun a Karlštejn, a tak mohou některé koleje sloužit pouze pro odstavení vlakových jednotek.

1.2 Řízení modelového kolejiště

Modelové kolejiště je složeno z několika úrovní, které obstarávají možnost řízení kolejiště.

Obrázek 1-1: Koncept řízení modelového kolejiště

1.2.1 Blok USB/CAN

Blok USB/CAN obstarává připojení řídícího systému, kterým může být mikrokontroler či počítač v laboratoři, ve kterém se nachází software umožňující řízení provozu na modelovém kolejišti. Připojení převodníku je provedeno pomocí krouceného páru přes RJ-45 konektor, který slouží k připojení UTP kabelu. Ten slouží k propojení všech následujících jednotek. Program běžící v PC zároveň slouží ke sledování a zpracování veškerých zpráv zasílaných kolejišti či zpráv, které zasílají jednotky nasazené na modelovém kolejišti. [1] [2]

1.2.2 Blok DCC generátor

Blok DCC generátor slouží k přijímání CAN zpráv, ze kterých je následně složen DCC signál, který je veden dále. Pro celé modelové kolejiště je použit jediný DCC generátor, jelikož je totožný DCC signál rozeslán do všech úseků kolejiště. DCC generátor dále obsahuje watchdog časovač, který kontroluje, zda v pravidelném intervalu přichází CAN zprávy. V opačném případě nastaví rychlost všech lokomotiv uvedených v databázi na nulu, čímž zabrání potenciálním kolizím. [2]

1.2.3 Blok DCC zesilovač

Blok DCC zesilovač se skládá z velkého počtu jednotek, kdy každá jednotka může obstarávat napájení až 8 izolovaných úseků. Tyto jednotky se zde nachází, jelikož jediný DCC generátor nedokáže dosáhnout dostatečné výkonové úrovně. Zároveň tyto jednotky slouží pro měření odběru proudů v jednotlivých izolovaných úsecích a umožňují zasílat zpětnou vazbu o obsazenosti úseků, což je jejich primární využití a díky čemuž je možné

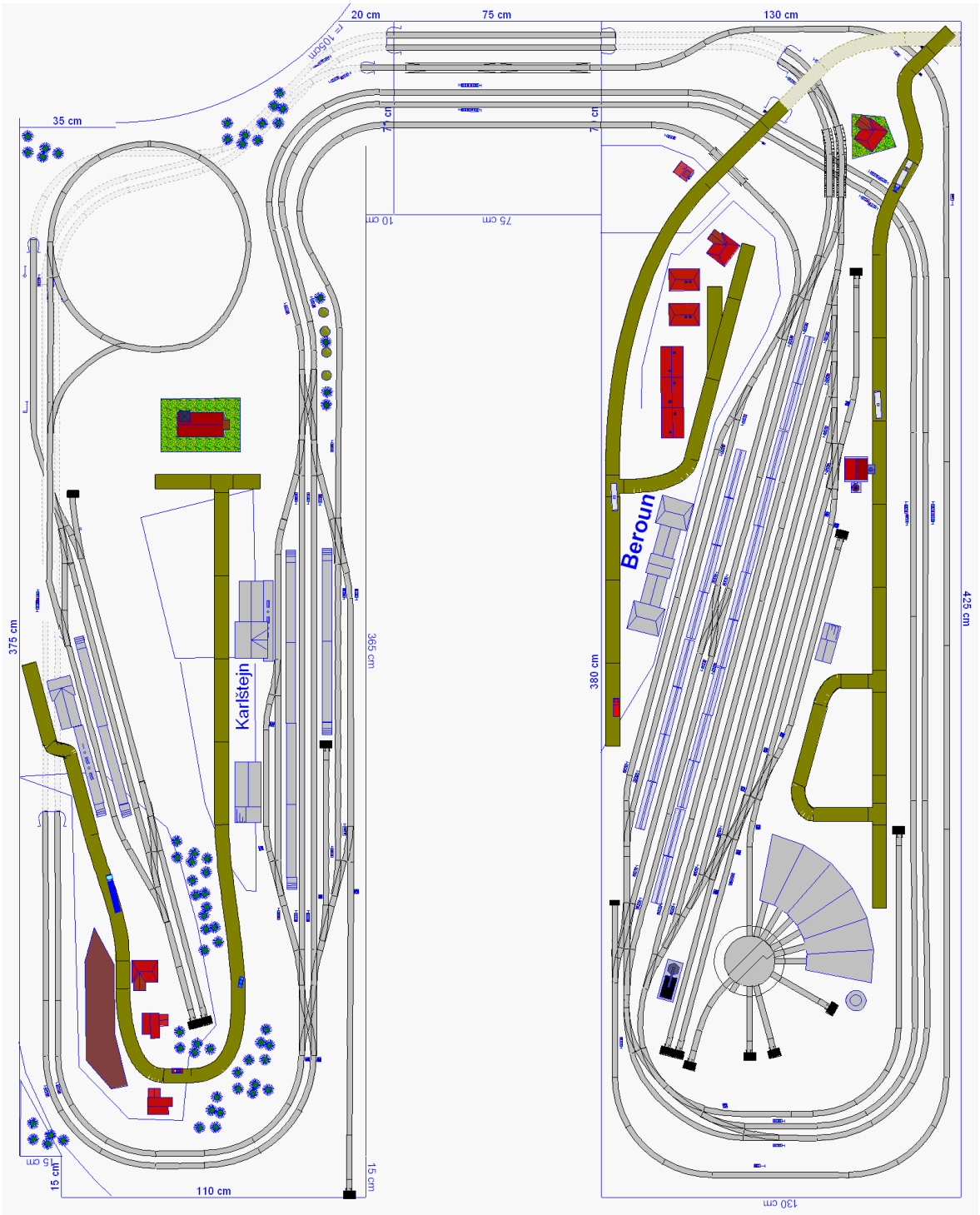
sledovat stav kolejiště v jednotlivých úsecích. Jednotky jsou schopny detekovat odběr proudu již v řádu nA, přičemž buzení modelového kolejiště je zajištěno přes H-můstek společně se step-down měničem. Zároveň byly tyto jednotky v minulém roce aktualizovány a vylepšeny. [2] [3]

1.2.4 Blok ovládání výhybek

Blok ovládání výhybek slouží pro nastavení výhybek do vhodné polohy podle informací obsažených v CAN zprávách. Tyto jednotky využívají servomotory, které slouží k nastavení výhybky do správné polohy. Výhybku lze natočit buď vlevo či vpravo podle definovaných softwarových dorazů, nebo softwarově na přesný úhel natočení. [2] [4]

1.2.5 Blok ovládání návěstidel

Blok ovládání návěstidel slouží k rozsvícení návěstních znaků na návěstidlech podle toho, zda je možno pokračovat do následujícího úseku. Komunikace s jednotlivými návěstidly probíhá opět přes CAN rozhraní, díky čemuž jsou jednotlivá návěstidla buzena a řízena. V současnosti je nicméně potřeba upravit hardware zajišťující jejich buzení a z tohoto důvodu jsou návěstidla mimo provoz. [2]



Obrázek 1-2: Mapa modelového kolejiště

2 Sběrnice CAN

Sběrnice CAN je sériový komunikační protokol, který byl vyvinut v 80. letech minulého století firmou Bosch z důvodu snížení spotřeby kabelů a zabezpečení komunikace v automobilech. Právě i díky jeho vlastnostem, kdy je umožněna komunikace velkého počtu dílčích systému, vysoké přenosové rychlosti, spolehlivosti a jednoduchosti, došlo k jeho rozšíření. [5]

Sběrnice CAN slouží pro komunikaci jednotlivých prvků modelového kolejiště s řídícím systémem, který je připojen přes převodník z USB na CAN sběrnici. Po sběrnici jsou z řídícího systému zasílány instrukce do DCC generátoru a DCC zesilovačů, ve kterých je signál převeden na DCC signál, který slouží k řízení lokomotiv.

2.1 Princip sběrnice CAN

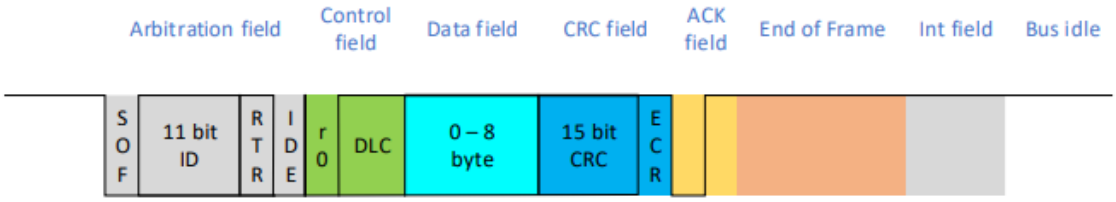
Princip sběrnice CAN spočívá ve vytvoření sdílené sběrnice, která disponuje prioritním rozhodováním na základě identifikátoru zprávy. Komunikace probíhá na základě zasílání rámců, kde je standardně definována maximální velikost datového rámce na 8 bytů. Zároveň se jedná o protokol multi-master, který umožní každému uzlu zaslat komunikační rámec a neobsahuje žádný master uzel, jelikož každý z uzlů se může chovat jako master, nicméně vzájemné kolize jsou řešeny na základě prioritního rozhodování. Odesílaná zpráva, ať už žádost o data či datová zpráva, je odeslána vždy všem uzlům na sběrnici, nicméně každá zpráva je uvozena specifickým identifikátorem, díky kterému lze zajistit, aby uzel přijímal pouze zprávy s předem definovaným identifikátorem. V případě detekované chyby je automaticky znova zaslán nekorektně přijatý rámec. [5]

2.2 Komunikační protokol CAN

Komunikační protokol má normou definovány tři druhy, kterými jsou CAN 2.0 A, CAN 2.0 B a CAN FD. Rozdíl mezi verzí A a B spočívá pouze v délce identifikátoru, kdy CAN 2.0 A disponuje 11bitovým identifikátorem a jedná se o standardní formát, kdežto CAN 2.0 B má 29bitový identifikátor a jedná se o rozšířený formát. Zároveň platí, že standardní formát má vždy vyšší prioritu než rozšířený formát. Nutno podotknout, že identifikátor definuje prioritu a obsah zprávy, ale nedefinuje konkrétního příjemce, kterému má být zpráva doručena. CAN FD je obdoba CAN 2.0 A, umožňuje však přenos většího

množství dat a to až 64 bytů. Zároveň umožňuje využití odlišných přenosových rychlostí pro arbitrážní a datovou fázi, což může způsobit zkrácení doby přenosu. V další části bude blíže popsán pouze CAN 2.0 A s 11bitovým identifikátorem, neboť je přítomen na modelovém kolejišti a je schopen vytvořit 2048 unikátních identifikátorů. [5] [6]

Standardní formát zprávy pro CAN 2.0 A lze vidět na obrázku 2-1. Rozšířený formát by obsahoval za bitem IDentifier Extension ještě další 18bitové pole pro druhou část identifikátoru.



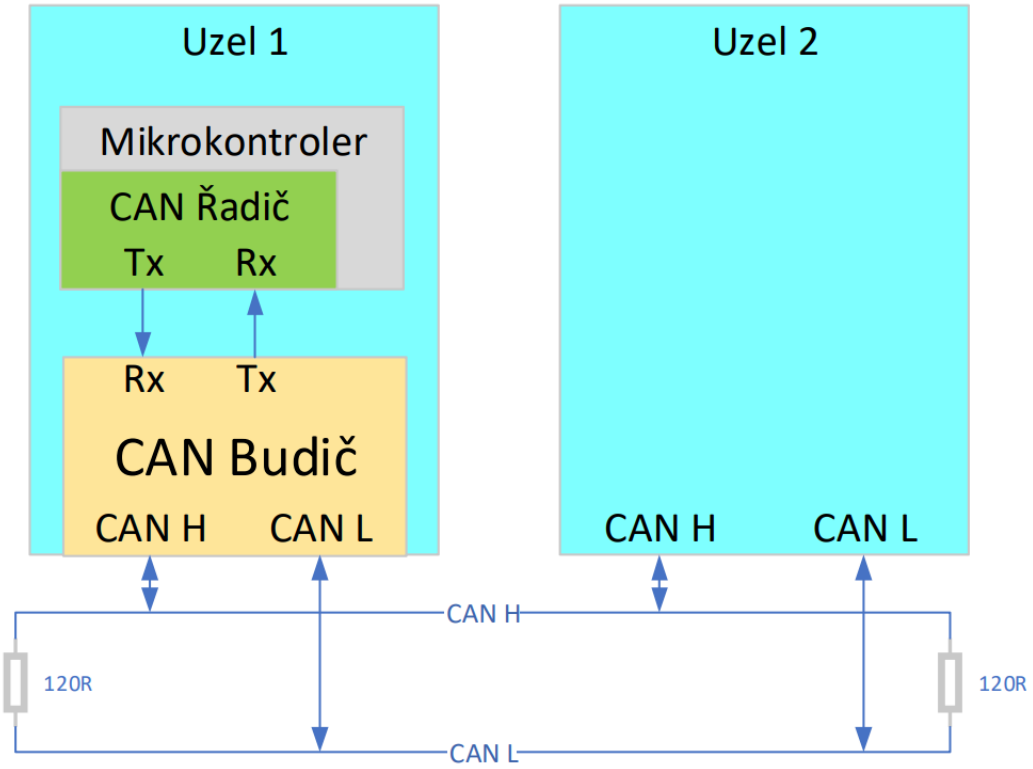
Obrázek 2-1: Standardní formát zprávy pro CAN 2.0 A

Standardní odeslaná zpráva je složena z několika částí. Nejprve je odeslán dominantní start bit, kterým je logická nula. Po něm následuje 11bitový identifikátor zprávy, který zároveň slouží k určení priority zaslané zprávy. Jako další je zaslán RTR bit, který slouží k určení, zda je odeslána datová zpráva nebo je zpráva odeslána bez dat a jedná se o žádost o data. Následuje IDE bit, který definuje, jestli se jedná o standardní formát pro CAN 2.0 A, či o rozšířený formát pro CAN 2.0 B a bit r0, který je pouze rezervním bitem. Poté následuje pole DLC o velikosti 4 bitů, které definuje velikost datového pole. Po něm následuje 15bitový cyklický redundantní kód, který je vygenerován z charakteristického polynomu a v případě detekování CRC chyby libovolným uzlem dojde k vygenerování chyby. Poté následuje bit značící ukončení cyklického polynomu v poli CRC. Nato jsou vyhrazeny 2 bity pro přijímače, jelikož během této doby jsou schopny detekovat správnost zaslaných dat a tím jsou schopny vysílači sdělit, zda byla data doručena alespoň jednomu příjemci bez chyb. Jestliže data byla přijata v pořádku, zašlou vysílači dominantní bit pro potvrzení správného přenosu. V opačném případě vysílač bude zaslání zprávy opakovat. Na to navazuje 7bitové pole značící konec zprávy, během kterého je umožněno přijímačům zaslat zprávu o chybě v cyklickém redundantním kódu vysílači a ohlásit tím chybu. Poslední, třetí, bitové pole slouží k vytvoření prodlevy mezi zasláním jednotlivých zpráv. [6]

2.3 Připojení na sběrnici

Připojení na sběrnici je zpravidla realizováno pomocí diferenční sběrnice dle normy, která se obvykle skládá ze dvou vodičů, označených CAN_H a CAN_L, kde jsou dominantní a recesivní úrovně definovány rozdílovým napětím na těchto vodičích. Pro High speed je definován recesivní stav logickou jedničkou, kde CAN_H = 3,5 V a CAN_L = 1,5 V, zatímco v případě dominantního stavu, kterým je logická nula, je definováno, že CAN_H = CAN_L = 2,5 V. Oproti tomu pro Low speed je dominantní stav definován úrovněmi CAN_H = 0 V a CAN_L = 5 V. Pro recesivní bit je definována úroveň CAN_H = 3,6 V a CAN_L = 1,4 V. Z číselných hodnot si lze všimnout, že signálové vodiče CAN_H a CAN_L jsou vzájemně invertované. [5] [6] [7]

Jednotlivé uzly jsou následně připojovány k této sběrnici pomocí konektorů a jejich počet je dán typem budičů. Budiče jsou využívány k převodu mezi TTL a CAN úrovněmi signálu. Sběrnice je zakončena na obou koncích odporem o velikosti 120 Ω.



Obrázek 2-2: Obecné schéma připojení ke sběrnici [6]

3 Digitální řízení modelové železnice

Modelová kolejiště lze obecně řídit dvěma způsoby, analogově či digitálně. I přesto, že oba způsoby vyžadují přivedení elektrické energie, velmi se odlišují.

Při analogovém řízení je do kolejnic přivedeno stejnosměrné či střídavé napětí, které slouží k napájení lokomotiv, ve kterých se nachází stejnosměrný motor a umožňuje to pohyb v daném napájeném úseku. Toto řízení ovšem neumožňuje plynulost pohybu a rychlost je dána velikostí amplitudy. Hlavní nevýhodou analogového řízení kolejiště je nemožnost odděleného řízení jednotlivých lokomotiv, jelikož vždy jsou napájeny všechny lokomotivy v daném úseku. Není tedy možné zpomalit pouze jedinou lokomotivu v daném napájeném úseku kolejiště. Z tohoto důvodu by bylo nutné rozdělit kolejiště na velké množství izolovaných úseku, čímž by bylo možné řídit i jednotlivé lokomotivy, nicméně by musel být navržen systém, který by při průjezdu vlakové jednotky izolovaným úsekem připojil napájení a po jejím opuštění úseku by došlo k odpojení jednotky. Další nevýhodou tohoto řízení je pouhé přivedení napájení lokomotiv, kdy není umožněno řídit signálem v kolejnicích přestavníky či návěstidla. [8]

Modelové kolejiště je ovšem řízeno digitálně, a proto je princip tohoto řízení více detailně rozebrán v této kapitole.

3.1 Digitální řízení modelového kolejiště

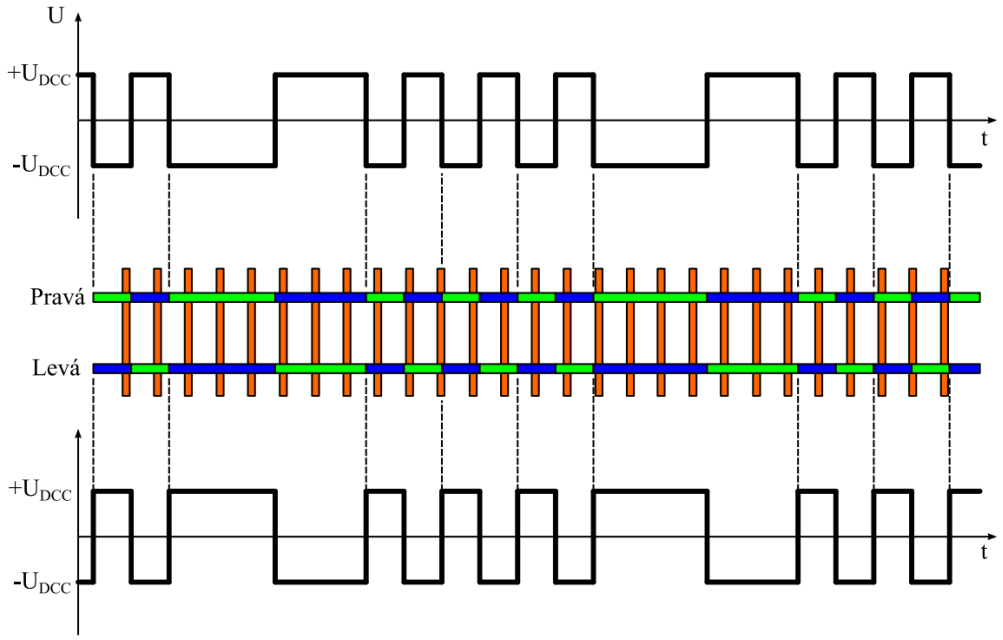
Digitální řízení kolejiště nabízí výrazně rozsáhlejší možnosti řízení, kdy kromě napájecího signálu s konstantní amplitudou jsou v signálu obsaženy i další informace pro řízení. Každá lokomotiva má navíc svoji unikátní adresu, díky které lze každé lokomotivě zaslat v signálu jedinečné příkazy pro rozjezd, zastavení či rozsvícení světel. Tento signál se nazývá DCC signál a je generován řídící jednotkou. Musí však být dostatečně zesílen, jelikož slouží pro napájení všech zařízení na kolejišti včetně lokomotiv, přestavníku či návěstidel. Pro digitální řízení kolejiště je nezbytné, aby se v každé lokomotivě nacházel dekodér, který snímá veškeré signály a využije pouze ty, které jsou určeny dané lokomotivě a díky tomu je možné řídit více vlakových jednotek ve stejném úseku kolejiště. DCC signál je veden v kolejnicích a jeho hlavní výhoda spočívá v adresování signálu pouze konkrétní vlakové jednotce. [8] [9]

3.2 DCC signál

Komunikační protokol DCC je způsob řízení, který byl vyvinut společností NMRA pro příkazové řízení modelové železnice. Tento signál je využíván jednak pro napájení lokomotiv a zároveň i pro jejich řízení. Daný signál je vždy symetrický a mění se periodicky. DCC signál je veden v kolejích, přičemž se jedná o rozdílový signál, kdy v levé kolejnici je signál opačný k signálu, který se nachází v pravé kolejnici. [10]

Logická jednička je v signálu reprezentována jako kladný signál v pravé kolejnici v rozmezí od 55 do 61 µs. Po tuto dobu musí být v druhé (levé) kolejnici záporná hodnota signálu. Logická nula v pravé kolejnici je definována jako kladný signál od 95 do 9900 µs pro dobu trvání jednoho bitu. Zároveň platí, že pro logickou jedničku musí střída dosahovat 50 %, což není pro logickou nulu podmínkou. [10] [11]

DCC standard zároveň definuje požadované hodnoty napětí, které musí DCC zesilovač dodávat pro správný chod kolejiště. Normou je definováno, že pro fungování systému musí špičková hodnota napětí dosahovat minimálně 7 V a maximálně 22 V, přičemž typická hodnota napětí je mezi 12 až 16 V. Toto jsou typické hodnoty napětí, nicméně minimální, maximální a typické hodnoty se mohou mírně lišit v závislosti na měřítku. [10] [11]



Obrázek 3-1: Možný průběh DCC signálu v levé a pravé kolejnici [4]

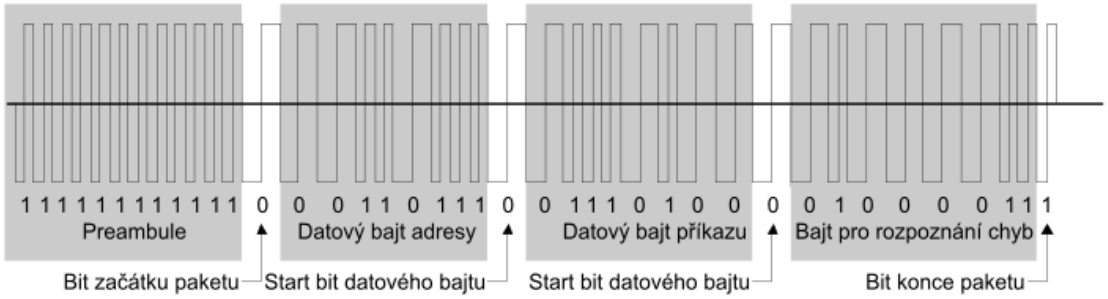
Na obrázku lze vidět, jak vypadá průběh DCC signálu v kolejnicích. Daný signál je možno bitově zapsat jako 1011 1011 a lze na něm vidět, že signál značící logickou jedničku má shodné trvání poměru signálu pro levou a pravou kolejnici. Zároveň se jedná

o symetrický signál, a proto je v druhé kolejnici signál o stejném intervalu, ale obrácené polaritě. Pro signál značící logickou nulu je výrazně delší doba trvání pulsu a zároveň nemusí docházet ke shodě v délce trvání obou period.

Pro logickou nulu je využíván dostatečně dlouhý interval, aby bylo umožněno využívat na kolejišti stále i analogové lokomotivy. Pro tyto lokomotivy představuje digitální signál PWM modulaci, kdy změna střídy způsobí rozdíl potenciálů a umožní rozjezd analogové lokomotivy. [10]

3.3 DCC komunikace

Zajištění komunikace, a tedy i řízení DCC signálem, probíhá formou paketů, kdy každý paket je zpráva s jasně definovaným začátkem a koncem z důvodu jasné rozlišitelnosti v toku obsáhlého množství dat. Každý paket je dělen na čtyři části, kterými jsou preambule, adresová část, datová část a část kontrolní, přičemž každá z těchto částí je oddělena od další části logickou nulou a paket je zakončen logickou jedničkou.



Obrázek 3-2: Příklad DCC paketu [10]

Jako první je zaslána preambule, která je tvořena alespoň 10, v praxi však minimálně 14, logickými jedničkami za sebou, které značí začátek vysílání dat. Poté následuje logická nula značící začátek 8 datových bitů pro adresu příjemce. Po další logické nule oddělující konec datového bytu následuje jeden či několik datových bytů, které jsou od sebe vždy odděleny logickou nulou. Posledních 8 bitů slouží ke kontrole a detekci chyb. Tento byte je logickou funkcí XOR všech předchozích adresových a datových bytů. [6][7]

3.4 Adresy DCC

Jelikož jsou neustále zasílány různé signály, je zapotřebí dodržet jistá pravidla stanovená normou NMRA pro adresování. Adresový prostor je rozdělen do jednotlivých oblastí, kde každá skupina charakterizuje jiné použití, přičemž největší zastoupení mají dekodéry lokomotiv. V tabulce 3-1 jsou zobrazeny jednotlivé druhy adres, kdy písmeno A značí bity, které se pro daný rozsah mění, kdy například pro dekodéry příslušenství lze zapsat jejich binární rozsah od 1000 000 do 1011 1111. [12] [13]

Tabulka 3-1: Rozdělení adres pro DCC standard. [10]

Adresa (bin)	Rozsah	Použití
0000 0000	0	Broadcast – paket pro všechny dekodéry
0AAAAAAA	1-127	Lokomotivní (multifunkční) dekodéry se 7bitovou adresou
10AAAAAA	128-191	Dekodéry příslušenství (s 9 nebo 11bitovou adresou)
110AAAAA + 11100AAA	192-231	Lokomotivní (multifunkční) dekodéry se 14bitovou adresou
11101AAA	232-254	Vyhrazeno pro budoucí použití
11111111	255	NO-cast – idle paket (paket pro nikoho)

3.5 DCC pakety

Jednotlivé DCC pakety mají přesně definovaný tvar a jejich konkrétní podobu lze nalézt v normě NMRA S 9.2, nicméně si zde dovolím uvézt alespoň ty nejdůležitější.

3.5.1 Rychlostní a směrový paket

Rychlostní paket je jedním z nejdůležitějších paketů, jelikož zajišťuje pohyb lokomotivy po kolejišti. Preambule, značící začátek přenosu, má standardní tvar několika jedniček. Poté následuje logická nula značící začátek přenosu, na kterou navazuje první (adresový) byte, který definuje, pro kterou lokomotivu je daný paket určen. Po datovém bytu je opět přítomna logická nula, která značí začátek přenosu datového bytu. Ten je určen prefixem 01, který slouží k indikaci paketu pro určení směru a rychlosti. Po prefixu následuje bit D značící orientaci pohybu lokomotivy. V případě logické jedničky bude lokomotiva pokračovat po směru, pro logickou nulu bude lokomotiva uvedena do pohybu opačným směrem. Bit C

obsahuje aditivní rychlostní bit a jedná se o nejnižší bit, který se oproti zavedeným zvyklostem nenachází na poslední pozici. Zbytek bytu je zabrán bity určujícími rychlost lokomotivy. Poté následuje bit pro ukončení datového bytu a kontrolní byte, který jak již bylo zmíněno dříve, je prostou funkcí XOR. Celá sekvence je poté ukončena logickou jedničkou, která znázorňuje ukončení zaslaného paketu. [13]

Tabulka 3-2: Složení paketu pro určení směru a rychlosti

Preambule		Adresový byte		Datový byte		Kontrolní byte	
111111111111	0	0AAA AAAA	0	01DC SSSS	0	EEEE EEEE	1

3.5.2 Reset paket

Druhým ze základních paketů je reset paket, který slouží k resetování všech DCC dekodérů do výchozí polohy. Rozložení paketu je stejné jako u rychlostního paketu, ovšem adresový byte i datový byte je tvořen v obou případech logickými nulami, kdy vzhledem k povaze těchto bytů je i kontrolní byte tvořen samými nulami. [13]

Tabulka 3-3: Složení paketu pro resetování dekodérů

Preambule		Adresový byte		Datový byte		Kontrolní byte	
111111111111	0	0000 0000	0	0000 0000	0	0000 0000	1

3.5.3 Idle paket

Další ze základních paketů je idle paket, který není zasílán žádnému dekodéru. Tento paket má definovaný adresový byte jako samé jedničky a datový byte plný nul. V případě přijetí tohoto pakety neprovedou digitální dekodéry žádnou akci. [13]

Tabulka 3-4: Složení idle paketu

Preambule		Adresový byte		Datový byte		Kontrolní byte	
111111111111	0	1111 1111	0	0000 0000	0	1111 1111	1

3.5.4 Broadcast paket

Další z paketů je broadcast paket, který má stejný adresový byte jako idle paket, nicméně datový byte je odlišný. Tento paket je význačný tím, že ho musí přijmout každý digitální dekodér a všechny dekodéry tak musí obsaženou instrukci zpracovat. Datový byte je tvořen prefixem 01, který značí, že se bude tento paket zabývat rychlostí lokomotiv. Pokud se nachází v bitu S logická jednička, digitální dekodéry musí zastavit všechny lokomotivy. Bit D značí směrovost lokomotivy, nicméně jestliže je hodnota bitu C je logická jednička, je tento bit zanedbatelný. Kontrolní byte slouží jako v předchozích případech pro rozpoznávání chyb. Jak již lze vyčíst z popisu tohoto paketu, tak broadcast paket je využit pro nouzové zastavení všech lokomotiv na kolejišti. [13]

Tabulka 3-5: Složení broadcast paketu pro zastavení všech lokomotiv

Preambule		Adresový byte		Datový byte		Kontrolní byte	
111111111111	0	0000 0000	0	01DC 000S	0	EEEE EEEE	1

3.5.5 Ostatní pakety

V předchozích případech byly uvedeny nejzákladnější pakety, které je potřeba na kolejišti sledovat a jednalo se o pakety pro pohyb vlaků po kolejišti či jejich nouzové zastavení. Dále jsou normou definovány pakety pro pohyb lokomotivy, které mají adresu delší než jeden byte, pakety pro dekodéry příslušenství či pakety pro ovládání funkcí lokomotiv, kterými jsou například rozsvícení světel lokomotiv, či pakety, které mohou být využity pro napájení příslušenství připojeného ke kolejišti. Definované pakety lze nalézt v normách NRMA S-9.2 a S-9.2.1. [12] [13] [14]

4 Analýza modelového kolejiště z hlediska hardwaru

Modelové kolejiště je po více než posledních 15 let postupně vylepšováno nově navrženými elektronickými zařízeními a i aplikacemi, které slouží pro její řízení. Vše je vyvíjeno převážně studenty v rámci semestrálních, bakalářských a diplomových prací. Tato kapitola se věnuje současnému stavu kolejiště z hlediska řídící elektroniky. Lze ji obecně rozdělit do tří částí, které slouží pro řízení modelového kolejiště a těmi jsou generátor, zesilovač a kontrolér přestavníků. Propojení těchto částí je zajištěno pomocí konektorů RJ-45 a UTP kabely s řídícím systémem. Do budoucna jsou v plánu renovace kontroléru točny a světelných návěstidel, nicméně v momentální době nejsou tyto bloky v provozu. V neposlední řadě se zaměřím na vysvětlení komunikačního protokolu pro komunikaci s jednotlivými řídícími bloky kolejiště.

4.1 Generátor DCC signálu

Generátor DCC signálu byl vytvořen v roce 2018 (viz [2]) a slouží ke generování DCC signálu podle přijatých dat do kolejnic modelového kolejiště. Obstarává tedy příkazy pro jízdu jednotlivých lokomotiv, nastavení výhybek a řízení všech prvků kolejiště vytvořeným signálem. Tyto signály jsou přijímány od řídícího systému, kterým je v tomto případě počítač, a s kterým komunikuje pomocí sběrnice CAN. Tato jednotka disponuje LED signalizací, která značí, zda se jedná o řízení přes sběrnici CAN. V takovém případě svítí zelenou barvou. V případě manuálního řízení z desky je rozsvícena modrá LED. Pokud však DCC generátor neobdrží po dobu delší než 1,5 s žádná data přes sběrnici CAN, dojde k rozsvícení červené barvy. Rozsvícení této LED rovněž značí, že kromě neobdržení zprávy došlo také k přetečení watchdog časovače, který slouží k zajištění funkčnosti systému. Po jeho přetečení dojde k zastavení všech lokomotiv na modelovém kolejišti, aby v případě nefunkčnosti CAN komunikace nedošlo ke kolizi. [2]

V principu je tedy vytvořen v řídícím systému například příkaz k rozjetí lokomotivy Ragulin. Z daného příkazu se vytvoří paket, který obsahuje data značící, že se jedná o příkaz pro DCC generátor, unikátní ID zvolené lokomotivy, rychlost pohybu a směr pohybu. Generátor z těchto dat poté vytvoří DCC paket, který je zaslán do kolejnic a jsou v něm obsaženy všechna potřebná data. Jednotlivé lokomotivy poté přijímají tyto DCC pakety a lokomotiva, která má dané unikátní ID přijme příkaz a vykoná pohyb zvolenou rychlostí a směrem.

Další z výhod tohoto generátoru je možnost ovládání pohybu všech jednotek nacházejících se na napájených částech kolejiště, což umožňuje jednoduché otestování funkčnosti jednotlivých lokomotiv. V tomto případě je využita adresa 1, která zašle data pro jízdu do všech lokomotivních dekodérů. Nevýhodou tohoto generátoru je jeho příležitostné přetížení, kdy nastávají situace, za kterých je zaslán příkaz generátoru pro vytvoření DCC paketu pro rozjezd lokomotivy, nicméně generátor není schopen pakety vytvořit a uvést lokomotivy do pohybu.

4.2 Zesilovač DCC signálu

Zesilovač DCC signálu byl navržen a inovován v roce 2022 (viz [4]) a jedná se o řídící jednotku, která se stará o napájení jednotlivých úseků modelového kolejiště. Jednotlivé zesilovače DCC signálu zesilují signál zaslaný generátorem a napájí až 8 izolovaných úseků. Díky jejich připojení ke kolejnicím je DCC signál šířen jednotlivými úseky kolejiště a umožňuje lokomotivním dekodérům přijímat povely, které jim jsou zaslány z generátoru.

Druhým z úkonů zesilovačů je sledování obsazenosti úseků a měření v nich proudový odběr. Ten je měřen pomocí A/D převodníků a je periodicky odesílán do řídícího systému. Proudový odběr je změřen vždy 8x na každém kanále či izolovaném úseků a následně je tato hodnota zprůměrována. Změřená hodnota je poté pro každý úsek zaslána vždy v jednom bytu, kdy hodnota odběru proudu může dosahovat hodnot 0 až 255. Zároveň je každý zesilovač vybaven LED diodami, které zobrazují, zda v daném úseku dochází k odběr proudu. [4]

Naměřené hodnoty proudu jsou zasílány zpět v nastaveném intervalu, který lze pro každou jednotku nastavit od 10 do 2500 ms, případně lze nastavit zasílání proudového odběru pouze na vyžádání. Zároveň lze řídící jednotku zesilovače konfigurovat z řídícího PC, díky čemuž je možné softwarově resetovat H-můstek v případě zkratu na kolejišti či restartovat celou řídící jednotku. Mezi další z možností této jednotky patří nastavení napájecího zdroje nebo vyžádání si zaslání aktuálního stavu nastavení jednotky, čímž lze ve zpětné vazbě získat informaci, zda jednotka funguje správně či že došlo k nějaké chybě. [4]

Během vývoje aplikace pro řízení modelového kolejiště bylo zjištěno několik chyb a nepřesností, které se nacházely ve zdrojovém kódu. Mezi nejpodstatnější patří chybné odesílání proudového odběru, který byl podle rozsvícení LED sloužících k vizualizaci proudového odběru správně změřen, nicméně nebyl zaslán přes sběrnici CAN ve správném formátu a nebylo tedy možné kolejiště řídit pomocí proudového odběru. Zároveň byla

chybně zpracovávána data v případě přetížení H-můstku, kvůli čemuž nebylo možné zaslat chybu řídícímu systému a řídící jednotka zasílala na zažádání data, která měla být zaslána v případě funkčnosti jednotky. Tyto chyby byly odstraněny, což nyní umožňuje měření proudového odběru na všech osmi kanálech a v případě přetížení H-můstku či zkratu na kolejišti jsou tato data okamžitě zaslána řídícímu systému. Mezi další z nevýhod aktuálního softwaru patří chybějící možnost nastavení čísla jednotky softwarově a konkrétní adresa je tak přiřazována staticky již při nahrávání firmwaru do desky.

4.3 Řídící jednotka výhybek

Řídící jednotka výhybek slouží k nastavování výhybek do správné polohy podle zaslané zprávy z řídícího systému. Tato jednotka byla navržena v roce 2020 (viz [3]) a řídící software pro ni byl vyvinut v roce 2022 (viz [4]). Zároveň byla v roce 2022 vytvořena i deska sloužící k uchycení servopohonů k modelovému kolejišti.

Každá z jednotek je schopna obstarat řízení až 8 výhybek, přičemž pro každou výhybku je využit digitální servomotor umístěný zespodu modelového kolejiště. Každý ze servopohonů umožňuje natočení od 0° do 180° s přesností na 1°, kdy je tento interval vyjádřen hexadecimálně od 0x00 do 0xB5. Celý interval ovšem není pro přenastavení výhybky zapotřebí. Z tohoto důvodu jsou softwarově implementovány maximální hodnoty dorazů, které ve výchozím stavu umožňují maximální natočení servomotoru v intervalu od 90° do 110°. Software ovšem umožňuje pro jednodušší ovládání výhybek i zjednodušené nastavení, které umožňuje provést prosté natočení výhybky vlevo a vpravo, kdy mezními hodnotami jsou definované dorazy. Z tohoto důvodu je při nastavování výhybek potřeba vždy provést natočení vlevo nebo vpravo, přičemž v případě natočení výhybky do shodné polohy se servomotor bude nacházet již v dané maximální poloze a nedojde tak k jeho přenastavení. [4]

Kromě přesného nastavení výhybky s přesností na 1°, nastavení výhybky do polohy vpravo a vlevo a nastavení softwarových dorazů umožňuje daný systém i provést resetování jednotky, získaní aktuálního nastavení jednotky či nastavení prodlevy před přetočením výhybky. [4]

Pro aktuální řídící software bylo úspěšně otestováno přenastavení výhybek vlevo a vpravo, což je hlavní využití této jednotky při řízení modelového kolejiště. Dané řešení nicméně obsahuje značnou nevýhodu v podobě chybějící hardwarové zpětné vazby. Nastavení všech výhybek je prováděno softwarově, kdy jsou zaslána data pro nastavení

výhybky, nicméně v případě samovolného přenastavení výhybky či uvolnění servomotoru není zajištěno okamžité zaslání zpětné vazby, díky které by bylo možné zjistit skutečnou polohu servomotoru.

4.4 Komunikační protokol

Komunikace mezi jednotlivými jednotkami modelového kolejiště a řídícím systémem je realizována pomocí sběrnice CAN o rychlosti 500 kbit/s. Každá zpráva je tvořena z několika nezbytně nutných částí, ovšem tou nejdůležitější je unikátní identifikátor, kterým jsou zaslána data pro specifickou řídící jednotku. Na celém kolejišti je využit standardní identifikátor, který má délku 11 bitů, kde první 4 bity značí konkrétní typ řídící jednotky a zbylých 7 bitů tvoří číslo jednotky.

Tabulka 4-1: Tabulka jednotlivých typů a adres pro řídící jednotky a jejich využití (horní 4 bity) [4]

Typ zprávy	Typ jednotky a využití
0	Nouzový signál (W)
1	DCC Generátor (W) – pohyb lokomotiv
2	DCC Generátor (R) – watchdog zprávy
3	DCC Zesilovač (R) – zprávy o odběrech proudu
4	DCC Zesilovač (R) – zprávy o nastavení jednotky
5	DCC Zesilovač (W) – nastavení jednotky
6	DCC Zesilovač (W) – nevyužito
7	Jednotka výhybek (W) – nastavení jednotky a výhybek
8	Jednotka výhybek (R) – zprávy od řídící jednotky o nastavení či chybách
9	Jednotka návěstidel (W) – čeká se na redesign
10	Jednotka točny (W) – čeká se na redesign
11	Jednotka točny (R) – čeká se na redesign

Zpráva na rozjezd lokomotivy určená DCC generátoru má proto typ zprávy 1 a číslo jednotky je též 1. Tento identifikátor tedy lze binárně zapsat jako 0001 0000001.

Každý typ zprávy má specifickou délku dat, která musí být vždy dodržena, jinak nelze data úspěšně pomocí DCC paketů odeslat. Data jsou tvořena jednotlivými byty dat, přičemž v jedné zprávě se může nacházet až 8 bytů. Každý typ zprávy má přesně definovaný počet datových bytů a v případě jejich nevyužití je zbytek bytů doplněn hodnotami 0x00.

4.4.1 Komunikační protokol DCC generátoru

Jak již bylo zmíněno dříve, tak pro správný chod modelového kolejiště stačí využít jediný generátor DCC signálu. Pro zápis do této jednotky je využit typ zprávy s hodnotou 1 a číslo jednotky je též 1. Zprávy pro DCC generátor obsahují pouze dva datové byty, kdy v prvním datovém bytu je obsaženo unikátní ID lokomotivy a v druhém datovém bytu jsou dány informace o rychlosti pohybu a orientaci směru lokomotivy. [2]

4.4.2 Komunikační protokol DCC zesilovače

DCC zesilovač obsahuje 3 typy zpráv, které jsou rozděleny dle jejich využití. V této práci je definováno pouze jejich základní rozdělení a charakteristiky. Jejich detailní popis lze nalézt v diplomové práci Patrik Albrechta (viz [4]).

Prvním typem je typ zprávy s hodnotou 3, který zasílá řídící jednotka do řídícího systému a slouží k zaslání konkrétních hodnot proudu. Tento typ zprávy je tvořen 8 datovými byty, kde datový byte 0 obsahuje naměřené proudy na kanálu 1 a poslední, 7., datový byte obsahuje hodnoty naměřeného proudu na kanále 8. [4]

Druhý typ zprávy je s hodnotou 4, který poskytuje zprávy o nastavení řídící jednotky. Opět se jedná o zprávu od řídící jednotky a obsahuje 4 datové byty a konkrétní typ zprávy lze určit z tohoto prvního bytu. V případě hodnoty 0x10 se jedná o zprávu vyžádanou řídícím systémem a slouží k zaslání zprávy jednotlivým klientům. Jednotlivé datové byty obsahují informace o periodě odesílání odběrů proudů, přetížení H-můstku, zapnutí zdroje či na kterých kanálech probíhá měření odběrů proudů. V druhém případě obsahuje první datový byte hodnotu 0xFF, která slouží pro zobrazení chybových hlášek. V druhém datovém bytu jsou obsaženy chyby detekované samotným systémem. Ve třetím datovém bytu se nachází chyby, které nastaly při konfiguraci řídící jednotky. [4]

Třetí typ zprávy s hodnotou 5, binárně vyjádřené jako 0101, slouží pro nastavení a konfiguraci řídící jednotky. Zároveň se jedná o jediný typ zprávy, který je zasílán řídící jednotce. Počet datových bytů je 4 a jejich rozdělení je provedeno pomocí nultého datového bytu, díky jehož hodnotě je získána základní instrukce. První datový byte slouží pro upřesnění dané instrukce a zbylé dva byty nejsou využity. [4]

Tabulka 4-2: Rozdělení základních instrukcí pro konfiguraci řídící jednotky podle nultého bytu [4]

Hodnota	Využití
0x01	Reset řídící jednotky
0x02	Nastavení periody odesílání odběrů proudů
0x03	Reset H-můstku
0x04	Nastavení zdroje
0x05	Nastavení měření na jednotlivých kanálech
0x10	Žádost řídícího systému o zaslání aktuálního stavu řídící jednotky

4.4.3 Komunikační protokol jednotky výhybek

Komunikační protokol řídící jednotky výhybek je tvořen pouze 2 typy zpráv a v této práci jej opět uvádím pouze okrajově. Jejich detailní popis lze nalézt v práci Patrika Albrechta (viz [4]).

První typ zprávy je s decimální hodnotou 6 a slouží pro konfigurační zprávy od řídícího systému do řídící jednotky výhybek. Délka tohoto typu zprávy je 8 bytů a rozdělení jednotlivých typů zpráv je opět dle nultého bytu. Řídící software této jednotky umožňuje provedení resetu řídící jednotky, nastavení prodlevy před natočením výhybek, nastavení softwarových dorazů či nastavení výhybky do požadované polohy. Pro nastavení výhybky mohou být použity dvě zprávy. První slouží pro nastavení výhybky s přesností na 1°, přičemž každý z osmi bytů odpovídá jedné výhybce. Druhá možnost nastavení výhybek spočívá pod hodnotou nultého bytu 0xBA, kdy jsou v prvním bytu vybrány výhybky určené k přenastavení, přičemž vybrané výhybky mají na dané pozici v binárním zápisu bytu logickou jedničku. Druhý byte poté obsahuje na pozicích, na kterých se má výhybka přenastavit, logickou nulu pro natočení vlevo a logickou jedničku pro natočení vpravo. Konkrétní způsob přenastavení výhybky lze vysvětlit na následujícím příkladu, kde se v prvním bytu nachází binární hodnota 1010 1010 a druhý byte je tvořen hodnotou 1100 1100. První byte značí, že dojde k nastavení výhybek 2, 4, 6 a 8, konkrétně k jejich natočení vlevo, pokud se na shodné pozici v druhém bytu nachází logická nula, či vpravo, když se na shodné pozici nachází logická jednička. Z tohoto důvodu se výhybka 4 a 8 mající na těchto pozicích bit v logické jedničce natočí vlevo a výhybky s bity v logické jedničce na pozicích 2 a 6 se přetočí vpravo. [4]

Tabulka 4-3: Rozdělení základních instrukcí řídící jednotky výhybek dle nultého bytu [4]

Hodnota	Využití
0x00 – 0xB5	Přesné nastavení výhybek s přesností na jeden stupeň
0xB6	Resetování řídící jednotky
0xB7	Nastavení časové prodlevy před otočením servopohonů
0xB8	Žádost řídícího systému o zaslání aktuálního stavu natočení servopohonů
0xB9	Nastavení softwarových dorazů servopohonů
0xBA	Výběr a nastavení výhybek pro přetočení vlevo a vpravo.

Druhý typ zprávy řídící jednotky výhybek je určen od řídící jednotky do řídícího počítače, má hodnotu 7 a jedná se o zprávy o velikosti 8 bytů. Tyto zprávy by měly být odeslány řídícímu systému po každém přetočení výhybek a řídící systém tak musí být schopen zpracovávat 3 různé druhy zpráv. První dvě zprávy slouží pro informaci obsahující natočení výhybek, kdy první formát obsahuje datové byty v rozmezí hodnot od 0x00 do 0xB5, přičemž tyto hodnoty značí přesné natočení mikro servopohonů ve stupních. Druhý formát zprávy se nachází obsahuje hodnoty v intervalu od 0xF0 do 0xF3 a značí, zda je servopohon natočen vlevo, vpravo či uprostřed. Poslední zpráva zaslaná řídícímu systému slouží k oznámení chybové hlášky a je tvořena 3 datovými byty a 5 nevyužitými byty, kde nultý a druhý byte mají hodnotu 0xFF a binární zápis prvního bytu slouží k identifikaci konkrétní chyby, která na jednotce nastala. [4]

4.5 Lokomotivy

Na modelovém kolejišti se momentálně nachází 12 různých jednotek, které mohou být řízeny. Každá jednotka má vlastní specifickou adresu, díky čemuž je dekodér schopen u každé lokomotivy poznat, zda zaslaný DCC signál dané lokomotivě patří. Po otestování daných lokomotiv však bylo zjištěno, že plně funkční jsou pouze lokomotivy Brejlovec, Ragulin, ICE, Herkules Priessnitz a Taurus EVB, nicméně i u těchto lokomotiv lze pozorovat odlišnosti v podobě maximální rychlosti či problémy v případě tažení přídavných vagónů. U lokomotiv Desiro (Kamera) a Desiro DB642 133-3 je problém s kontakty a nejsou tak na železnici schopny plynulého pohybu. Lokomotiva Taurus DHL přijímá a zpracovává DCC signál, nicméně u ní bylo objeveno mechanické poškození, kvůli kterému dochází k prokluzování kol a není tedy schopna rozjezdu. Lokomotivy ES363 a Para 555 se

nepovedlo uvést do pohybu podle jejich adres a bude potřeba jejich oprava či aktualizace firmwaru. [1]

Tabulka 4-4: Tabulka lokomotiv nacházejících se na kolejišti a jejich adresy [1]

Adresa lokomotivy	**Lokomotiva**
0x03	Brejlovec
0x05	Desiro (Kamera)
0x06	Taurus Railion
0x07	DB204 274-5
0x09	Ragulin
0x0A	ICE
0x0B	Taurus DHL
0x0C	Herkules Priessnitz
0x0D	Para 555
0x0E	T334 Rosnicka
0x0F	Desiro DB642 133-3
0x10	Taurus EVB
0x11	ES363

5 Analýza modelového kolejiště z hlediska softwaru

Poslední verze softwaru pro modelové kolejiště byla vytvořena v roce 2019 Vojtěchem Lapuníkem (viz [15]) a jedná se o verzi, která funguje na principu TCP komunikace. Díky tomu je umožněna asynchronní komunikaci mezi TCP serverem a jednotlivými klienty, kteří jsou ke kolejišti připojeni. V této kapitole bude popsána původní aplikace a budou vysvětleny některé balíky a třídy, které jsou použity i v nové verzi aplikace.

5.1 Řízení modelové železnice

Řízení modelové železnice je umožněno třemi způsoby. Dva z nich jsou v podobě „klikací“ aplikace vytvořené ve *Windows Form*. Třetí způsob komunikace spočívá v podobě textových zpráv pomocí TCP komunikace.

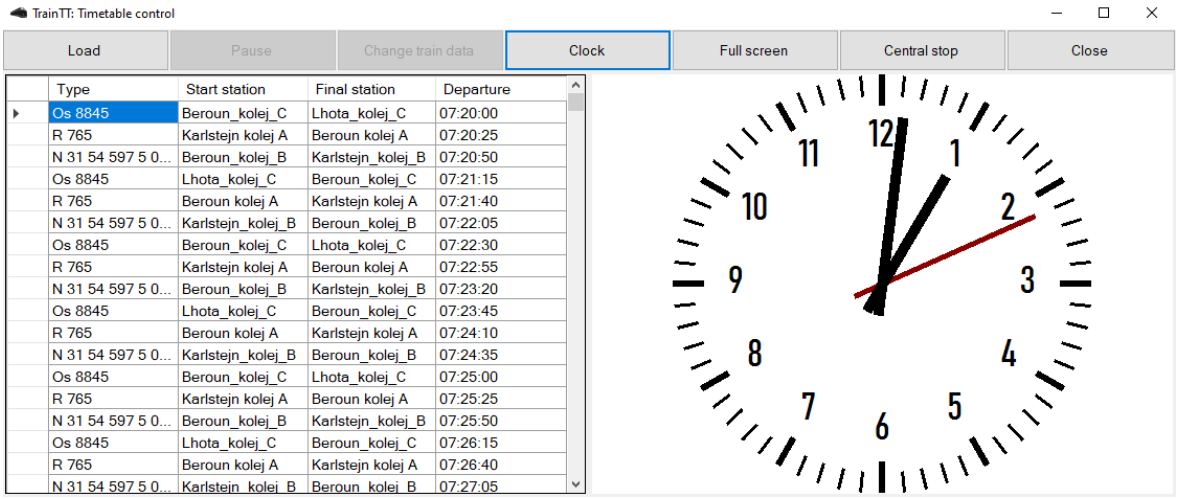
5.1.1 Řízení pomocí TCP komunikace

TCP komunikace spočívá v řízení kolejiště pomocí jasně definovaného typu zpráv z TCP terminálu. Jedná se o nejrychlejší možnost řízení kolejiště za předpokladu, že uživatel zná základní typy zpráv. Na začátku každé zprávy se nachází její typ, kterým je příkaz pro řídící jednotku, pohyb vlaku či nastavení dodatkové funkce vlaku. Tento typ zprávy je oddělen od následujícího obsahu pomocí dvojtečky. Poté již následují konkrétní data, která definují pevně zadaný obsah. V tomto obsahu je v případě dat pro lokomotivu její název, který se musí shodovat s definovaným názvem v konfiguračním souboru, a instrukce této lokomotivě, kterými mohou být rychlost či směr pohybu vlaku, přičemž všechna data v tomto obsahu jsou vždy oddělena čárkou. V případě údajů pro řídící jednotku kolejových úseků je vždy obsah tvořen instrukcí, která pevně definuje akci, která se má vykonat. [15]

5.1.2 Řízení pomocí jízdního řádu

Tato metoda spočívá v nejjednodušším řízení aplikace, kdy veškeré povely pro rozjetí a zastavení vlaku obstarává sám systém. Po spuštění této aplikace se zobrazí okno s několika tlačítky v jeho horní části a prázdný prostor s předpřipraveným polem *DataGridView*, který zajišťuje správné zobrazení jízdního řádu. Ten se nahrává pomocí tlačítka *Load* v horním panelu. Po jeho nahrání jsou zobrazena všechna data v této tabulce, která obsahují název vlaku, počáteční stanici, cílovou stanici a plánovaný odjezd. Všechny dřívější vlaky, které

mají odjezd před současným časem jsou smazány a odstraněny. Následně kliknutím na tlačítko *Start* dojde ke změně textu, které toto tlačítko obsahuje a vlaky se v čase odjezdu samy rozejdou do cílové stanice, ve které po určitém čase při detekci cílové koleje zastaví. Tyto data však lze upravit prostřednictvím tlačítka *Change train data*. Po stisknutí tohoto tlačítka je zobrazeno nové okno, ve kterém je možné upravit rychlost vlaku či dobu, po kterou má vlak pokračovat v pohybu ve stanici pro úplné dojetí do cílové pozice. Dále tato aplikace obsahuje tlačítko *Central stop*, které slouží k okamžitému zastavení všech vlaků na kolejišti a tlačítko *Clock*, po jehož stisknutí jsou zobrazeny či skryty digitální hodiny.

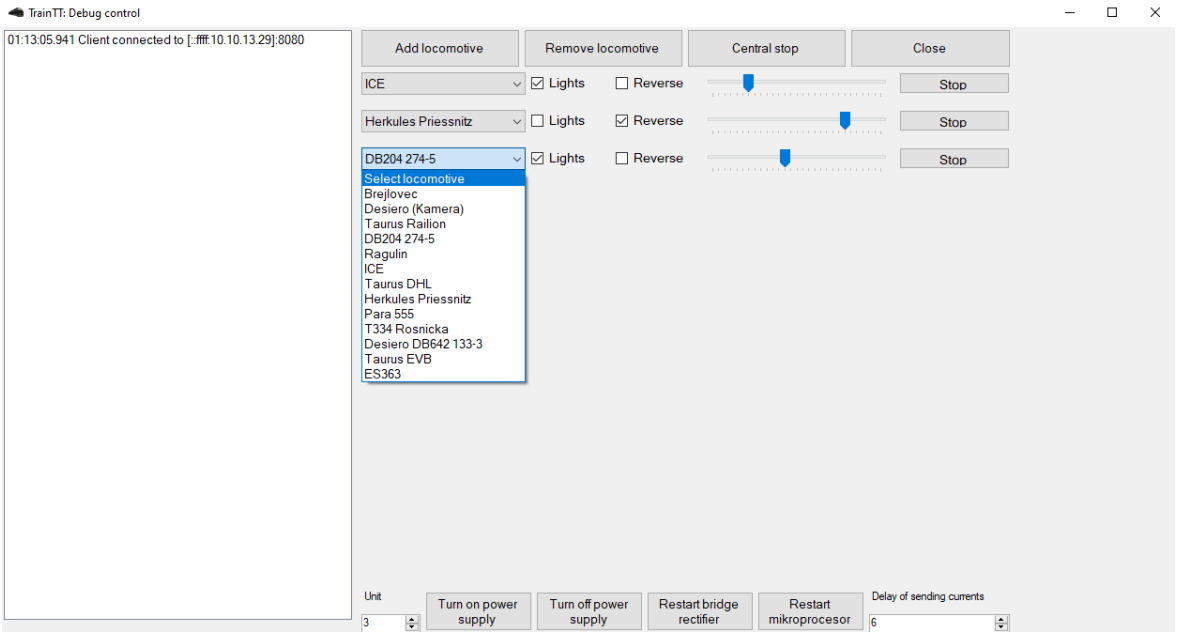


Obrázek 5-1: Vzhled aplikace pro řízení jízdním řádem

5.1.3 Řízení v manuálním režimu

Tato možnost řízení je založena na plně manuálním řízení. Po spuštění se uživateli zobrazí okno, které lze vidět na obrázku 5-2. V levé části se nachází panel, který slouží k zobrazení všech příchozích zpráv od řídící jednotky a prostřední část je určena po pohyb lokomotiv. Stiskem tlačítka *Add Locomotive* dojde k přidání nového řádku, ve kterém lze následně vybrat konkrétní lokomotivu z comboboxu a určit, zda má mít lokomotiva rozsvícená světla, jet popředu či pozadu a jaká má být její rychlost. Následně lze tlačítky *Start/Stop* danou lokomotivu rozjet či zastavit. Druhým z tlačítek v horní liště je *Remove locomotive*, kterým dojde k odstranění poslední přidané lokomotivy. Tato funkce nicméně není dořešená, jelikož při stisknutí tlačítka v případě, kdy není přidána žádná lokomotiva, dojde k samovolnému ukončení aplikace v důsledků neošetřené chyby. Třetí tlačítko je *Central stop* a slouží k okamžitému zastavení všech lokomotiv na kolejišti a zároveň k vymazání veškerých dat,

která byla pro lokomotivy přidána. Posledním z tlačítek na tomto panelu je *Close*, které slouží k prostému ukončení celé aplikace. Ve spodní části se nachází tlačítka určená pro nastavení úsekových jednotek. Nejprve je zobrazena volba pro určení čísla jednotky DCC zesilovače, jelikož na celém modelovém kolejišti se jich nachází hned několik. Následují tlačítka, kterými lze zapnout či vypnout napájení DCC zesilovače, restartovat jeho H-můstek či resetovat mikroprocesor. V neposlední řadě je zde zobrazena možnost zvolit hodnotu, po které má řídící jednotka zasílat data o obsazenosti jednotlivých úseků. Zvolená hodnota je poté vynásobena stem a výsledkem je prodleva mezi odesíláním dat v řádu milisekund. V případě nesprávného připojení k sériovému portu je zobrazeno vyskakovací okno, které uživatele upozorní na danou chybu.



Obrázek 5-2: Vzhled aplikace pro manuální řízení

5.2 Struktura aplikace

Celá aplikace je rozdělena do několika odlišných balíků a tříd. Některé z těchto tříd budou v této části detailně rozebrány, neboť došlo k jejich použití i v nové verzi aplikace po jejich úpravě a odstranění chyb.

5.2.1 Knihovna *TrainTTLibrary*

Tato knihovna tvoří základní stavební kámen celého programu a obsahuje čtyři soubory, kterými jsou *TCPBase*, *TCPClient* a *TCPServer* a *Packet*. Tyto soubory slouží pro vytváření

paketů zaslaných přes sériový port jednotlivým řídícím jednotkám, zpracování informací z přijatých dat ze sériového portu a zprostředkování asynchronní TCP komunikace. Tato knihovna je využita i v současné verzi práce, nicméně došlo k několika úpravám a odstranění chyb ve třídě *Packet.*

První tři vyjmenované soubory byly vytvořeny v minulosti vedoucím této diplomové práce a umožňují vytvoření TCP serveru, díky kterému je možná komunikace od klientů směrem k jednotlivým řídícím jednotkám. Hlavní z těchto tříd je *TCPBase*, která slouží jako rodič zbylých dvou tříd. Tato třída je tvořena několika metodami, které slouží pro zpracování přijatých dat a jejich odeslání určitým klientům či přes server a CAN převodník do modelového kolejiště. Třída *TCPClient* je využita pro zpracování a rozeslání dat na straně klienta. Naopak třída *TCPServer* slouží pro data určená serveru.

Poslední ze souborů v této knihovně je třída *Packet*. Jedná se o hlavní třídu, která slouží jako rodič dalších tříd, které též slouží pro vytváření či zpracování paketů, nicméně základní vlastnosti mají všechny tyto třídy shodné. Tato třída má pevně definované listy a proměnné pro hlavičku, adresu, data a kontrolní součet a nachází se v ní metody pro získání dat z bytového paketu či TCP paketu. Dále tato třída obsahuje metody pro zjištění typu zprávy podle horních 4 bitů identifikátoru, výpočet kontrolního součtu, vytvoření hexadecimální hodnoty odpovídající adrese a číslu jednotky, výpočet adresy a čísla jednotky z hexadecimální hodnoty z 11bitového identifikátoru či metoda, která z dat vytvoří bytový paket, který následně lze odeslat. Tyto obecné vlastnosti třídy Packet jsou následně děděny 7 potomky, kteří jsou dále rozdělení na základě jejich funkcí.

```
public class Packet
Packet methods

public class TrainMotionPacket : Packet
TrainMotionPacket methods

public class TrainMotionInstructionPacket : Packet
TrainMotionInstructionPacket methods

public class TrainFunctionPacket : Packet
TrainFunctionPacket methods

public class UnitInstructionPacket : Packet
UnitInstructionPacket methods

public class UnitInfoPacket : Packet
UnitInfoPacket methods

public class OccupancySectionPacket : Packet
OccupancySectionPacket methods

public class OLEDInformationPacket : Packet
OLEDInformationPacket methods
```

Obrázek 5-3: Znázornění dědičnosti jednotlivých tříd od třídy Packet

TrainMotionPacket je třída, která slouží pro data určená DCC generátoru. Ten z dat vytvořených v této třídě vytvoří DCC signál, pomocí kterého je možné uvést jednotlivé lokomotivy do pohybu. Tato metoda oproti nadřazené třídě využívá navíc vlastnosti typu jméno lokomotivy *Name*, rychlost vlaku *Speed*, adresu lokomotivy *ID* a orientaci pohybu po kolejišti *Reverse*. Tato třída využívá tři konstruktory pro zpracování dat, kdy první zpracovává data přijatá ve formě bytů, druhý zpracovává data přijatá ve formě TCP zprávy a třetí zpracovává data dle zadaných informací o lokomotivě, rychlosti a směru jízdy. Mimo to se v této třídě nachází metody pro vytvoření datových bytů, ve kterých je zakódována adresa a číslo jednotky a informace o rychlosti a směru lokomotivy.

TrainMotionInstructionPacket je rovněž třída využívána k pohybu vlaku, nicméně úlohou této třídy je dojet vlakem na předem určenou kolej a po detekci cílové koleje pokračovat v jízdě po předem stanovený interval, který zajistí spolehlivé dojetí lokomotivy do cílové stanice. Kromě vlastností ve třídě *TrainMotionPacket* ji navíc tvoří parametry s názvem úseku *Section* a doba přejezdu ve stanici *WaitTime*. Tato třída má také tři konstruktory, a to na zpracování příchozích dat ve formě bytů, TCP zprávy či z dat obsahujících informace o lokomotivě, rychlosti vlaku, směru pohybu, cílové stanici a doby přejezdu v cílové stanici.

TrainFunctionPacket je třída, která slouží pro zaslání instrukce pro příslušenství jednotlivých lokomotiv, kterým je rozsvícení světel. Tato třída obsahuje vlastnosti *Lights* obsahující logickou hodnotu značící, zda mají být rozsvícena světla, jméno a adresu lokomotivy a stejně jako předchozí třídy využívá tři konstruktory pro inicializaci, a to opět ve formě bytového paketu, TCP zprávy, dat o lokomotivě a informace, zda mají být její světla rozsvícena či nikoliv. Navíc tato metoda opět obsahuje dvě metody pro vytvoření příslušných datových bytů.

UnitInstructionPacket je třída pro zaslání dat řídícím jednotkám kolejových úseků, přičemž se jedná primárně o její konfiguraci. Tato třída je tvořena pouze vlastností typu *UnitInstruction,* pro kterou jsou přesně nadefinovány jednotlivé příkazy určené jednotkám. Jsou v ní obsažena přesná data určená pro řídící jednotku a podle daného typu jsou vytvořeny příslušné datové byty. Tato třída má opět tři konstruktory pro inicializaci, a to z bytového paketu, TCP zprávy a z dat, ve kterých je obsažená konkrétní instrukce předdefinována v *UnitInstruction*, čísla jednotky a jednoho bytu dat. Dále se v této třídě nachází metoda, která slouží pro převedení vstupního řetězce na daný typ odpovídající *UnitInstruction.*

UnitInfoPacket je oproti předchozí třídě určená pro data z řídící jednotky pro řídící systém. V této třídě je opět obsažena vlastnost typu *UnitInstruction*, jelikož jednotlivé adresy jsou pro obě třídy totožné. Zároveň má tato třída shodné konstruktory s třídou *UnitInstructionPacket.*

OccupancySectionPacket je předposlední třída dědící data od svého předka a jedná se o třídu, která je využita pro zpracování dat od řídící jednotky kolejových úseků o obsazenosti izolovaných úseků modelového kolejiště. Její jedinou přidanou vlastností je vlastnost typu *Sections*, ve které jsou zpracována přijatá data. Tato třída má pouze dva konstruktory, a to z pole bytů a z TCP zprávy, jelikož data pro tuto třídu nejsou nijako vytvářena pro odeslání, ale slouží pouze pro zpracování přijatých dat. V této třídě jsou obsaženy dvě metody, kterými jsou *RecognizeSection* a *RecognizeUnit*. *RecognizeSection* slouží pro získání čísla řídící jednotky kolejových úseků a uložení dat z přijaté TCP zprávy. Oproti tomu metoda *RecognizeUnit* je využita pro zjištění čísla jednotky a přiřazení dat o obsazenosti úseků ke správným úsekům.

OLEDInformationPacket je poslední ze tříd, které dědí od třídy Packet a je tvořena dvěma konstruktory. První spočívá ve formě přijaté TCP zprávy, kdežto druhý v seznamu přijatých zpráv. Využití této třídy spočívá ve zobrazení přijatých zpráv na přiloženém mikrokontroleru a jediná vlastnost této třídy je pole zpráv.

Kromě tříd využívajících vlastnosti svého rodiče jsou v tomto souboru třídy obsahující určitou část kolejiště. Těmi jsou jednotlivé úseky a lokomotivy. Třídy *LocomotiveInfo*, respektive *SectionInfo* slouží pro načtení dat z konfiguračních souborů pro jednotlivé lokomotivy či izolované úseky. Třída *Locomotive* poté slouží k získání informací o lokomotivě z jejího ID či jména. Třída *Section* slouží pro práci s jednotlivými izolovanými úseky, kdy v této třídě jsou z konstruktorů získány hodnoty odběru proudů a zda je daný úsek obsazen.

5.2.2 Knihovna *TCPServerTrain*

Tato knihovna je tvořena jednou třídou a jedná se o konzolovou aplikaci. Zároveň se jedná o nejdůležitější knihovnu celé práce a její program musí být vždy spuštěn, neboť program v této knihovně zajišťuje propojení jednotlivých klientů a modelového kolejiště přes sériový port. Zároveň tato knihovna zajišťuje příjem dat z modelového kolejiště a v opačném případě i odeslání dat přes sériový port.

Po spuštění tohoto programu dojde při inicializaci nejprve k vytvoření a spuštění TCP serveru pomocí metody *StartTCPServer*, ve které dojde k vytvoření příslušných eventů, přičemž ten nejdůležitější se týká detekce přijaté TCP zprávy. Následně dojde k propojení aplikace a kolejiště přes sériový port metodou *ConnectToSerialPort*, ve které též dojde k vytvoření eventů zajišťujících správnou komunikaci a mezi kterými se nachází i nejdůležitější událost pro příjem dat ze sériového portu. Posledním krokem v inicializaci je spuštění časovačů metodou *StartTimers*, čímž je zajištěno periodické sledovaní, zda nepřišla data přes sériový port od modelového kolejiště a data od jednoho z klientů v podobě TCP zprávy. V neposlední řadě zajišťuje periodické odeslání dat DCC generátoru v případě příkazu pro pohyb lokomotiv.

Metoda *SerialDataReceived* slouží pro asynchronní příjem dat ze sériového portu. Tato metoda je volána vždy, když je vyvolán event přijetím bytu ze sériového portu. Jednotlivé byty jsou průběžně ukládány a testuje se, zda přijatá data odpovídají formátu zprávy a jsou tvořena hlavičkou, adresou, daty a kontrolním součtem. V případě přijetí kompletního paketu je tento paket následně zpracován v metodě *SendTCPData_Tick*. Zde jsou postupně porovnávány přijaté pakety a jestliže se jedná o paket od řídící jednotky nebo o paket s odběry proudů, dojde k vypsání obdržených hodnot. V případě paketu o odběrech proudu v izolovaných úsecích dojde k uložení přijatých dat.

Metoda *TCP_DataRecv* zajišťuje asynchronní příjem zpráv od jednotlivých klientů a jejich následné zpracování. Přijatá data ve formě objektu jsou přetypována na textový řetězec a v případě úspěšné konverze je provedeno rozpoznání typu a podle toho je poté vytvořena nová instance daného potomka třídy *Packet* pomocí konstruktoru ve formě TCP zprávy. Po vytvoření konkrétního paketu dojde k vypsání jeho hodnoty do příkazového řádku a k jeho zpracování.

Zpracování daného paketu se liší podle typu zprávy. Pro pakety určené řídící jednotce kolejových úseků, řídící jednotce výhybek a pakety pro příslušenství lokomotiv jsou okamžitě zaslány přes sériový port do modelového kolejiště. Toto však neplatí pro pakety zajišťující pohyb vlaků po modelovém kolejišti, kterými jsou *TrainMotionPacket* a *TrainMotionInstructionPacket*. Tyto pakety obsahují data určená pro pohyb lokomotiv a musí tedy být zasílány do modelového kolejiště v pravidelných intervalech, který činí 50 ms. Z tohoto důvodu je nejprve zkoumáno, zda v seznamu pravidelně zasílaných příkazů není již uložen jiný paket pro lokomotivu se shodným ID s ID lokomotivy v přijatém paketu. Pokud paket pro danou lokomotivu přišel již dříve, dojde k jeho odstranění a na jeho místo je vložen paket nový, který bude periodicky odesílán. V opačném případě je nový paket vložen do daného seznamu, který je odesílán v pravidelných intervalech každých 50 ms.

5.2.3 Knihovna *TimetableControlTrainTT*

Knihovna *TimetableControlTrainTT* slouží pro řízení modelové železnice pomocí jízdního řádu a je složena z programů *Form1*, *ChangeTrainData* a *NoteInTimetable*. Tato knihovna však není v současné verzi využita, a proto zde bude popsána velmi střídmě.

Soubor *ChangeTrainData* slouží pro aktualizaci jízdního řádu v aplikaci. Nejprve je vytvořeno nové rozložení *Windows Form* a po aktualizaci dat jsou tyto data uložena a pomocí změněných dat probíhá řízení kolejiště.

Soubor *NoteInTimetable* je použit především pro správné načtení všech dat ze souboru a umožňuje jejich následné zpracování.

Hlavním souborem této knihovny je *Form1*, která obstarává celé řízení pomocí jízdního řádu. Nejprve je vytvořen TCP server, který umožňuje řízení modelové železnice. Následně se zde nachází metody umožňující zaslání dat přes TCP server, připojení k němu či jeho odpojení a ukončení. Další z metod v této knihovně je metoda, která umožní úspěšné načtení jízdního řádu. V aplikaci se dále nachází několik metod definujících akce po stisknutí jednotlivých tlačítek a vytvoření správného rozložení aplikace. V neposlední řadě se

v aplikaci nachází metoda pro periodické kontrolování jízdního řádu a v případě, že nastane čas odjezdu vlaku, dojde k vytvoření paketu a zaslání dat přes TCP server. Tato periodická kontrola jízdního řádu zároveň slouží i k jeho správnému zobrazení a zajišťuje, aby vždy byly zobrazeny nejaktuálnější odjezdy vlaků.

5.2.4 Knihovna *VisualDebugControlTrainTT*

Tato knihovna slouží pro manuální řízení vlaků modelové železnice a je tvořena soubory *Form1*, *Diode* a *DiodePanel*.

Soubory a třídy *Diode* a *DiodePanel* slouží k zobrazení informací o obsazenosti jednotlivých úseků prostřednictvím svítících LED, které jsou vyobrazeny v pravé straně aplikace. V případě, že v daném úseku dochází k odběru proudu, dojde k rozsvícení diody zelenou barvou.

Stejně jako v předchozím případě je i zde hlavním souborem *Form1*, díky kterému je umožněno manuální řízení kolejiště. Po spuštění této aplikace dojde k okamžitému spuštění TCP serveru. Stejně tak tato aplikace obsahuje metody, které umožňují odeslání dat od klienta přes TCP server, jeho odpojení od TCP serveru či potvrzení o připojení se k serveru. Dále tato aplikace, mimo metody definující události po stisknutí tlačítek či *check boxů,* obsahuje metody pro uložení dat přijatých od řídící jednotky kolejových úseků pro zobrazení odběru proudu. V neposlední řadě aplikace obsahuje metody pro zaslání vybraných dat lokomotivám či zastavení všech lokomotiv definovaných v konfiguračním souboru.

5.2.5 Knihovna *CommunicatorOLED*

Knihovna *CommunicatorOLED* je použita k periodickému zobrazení dat určených mikrokontroleru s LED obrazovkou, na kterém jsou v případě řízení modelového kolejiště jízdním řádem zobrazena data o nejbližším odjezdu vlaku. Nejprve probíhá připojení k TCP serveru a v případě, že jsou obdržena data určená pro mikrokontroler s OLED zobrazovačem, dojde ke zobrazení přijatých dat na displeji.

5.3 Zhodnocení aplikace verze 2.0

Druhá verze aplikace umožňuje řízení modelové železnice pomocí textové komunikace, jízdního řádu či manuálním řízením. Značnou výhodou této aplikace oproti verzi 1.0 je její funkčnost na principu TCP komunikace a DCC paketů, kdy i v nové verzi jsou využity již dříve napsané knihovny *TCPServerTrain* a *TrainTTLibrary* pro zpracování DCC paketů

a TCP komunikaci. I přesto však bylo pro správné fungování aplikace nezbytné provést patřičné úpravy, které budou uvedeny později.

Prvním z nedostatků této aplikace je její nedostatečné otestování a ošetření jednotlivých akcí po stisknutí některých tlačítek. Bylo tak nalezeno několik chyb, při kterých dochází k samovolnému ukončení aplikace v důsledku neošetřené chyby, čímž může docházet k ne zcela efektivnímu řízení modelového kolejiště.

Dalším nedostatkem je vzhled této aplikace, která působí poněkud zastaralým dojmem a v některých případech není její ovládání pro nového uživatele zcela intuitivní. Veškeré akce jsou navíc prováděny na jediné obrazovce, což z důvodů nových funkcí a možností nedovoluje aplikaci vhodné rozšíření, jelikož jsou již obrazovky zcela zaplněny a přidání nových funkcionalit do původní aplikace při zachování původního konceptu a vzhledu aplikace není prakticky možné. Aplikace rovněž umožňuje zobrazení proudového odběru pouze v 8 izolovaných úsecích v podobě LED, nicméně na celém kolejišti bude po jeho osazení izolovaných úseků přes 300.

Hlavní nedostatek této aplikace spočívá v jejím přizpůsobení na hardware, který byl v době vývoje k dispozici. Při vývoji aplikace verze 2.0 bylo k dispozici pouze 8 izolovaných úseků, čímž docházelo k provozu na modelovém kolejišti v oddělených okruzích. Zároveň se na modelovém kolejišti nenacházel hardware a software, který by umožňoval řízení a nastavení výhybek. Aplikace tedy sice umožňuje rozjezd, zastavení a jízdu lokomotivy, nicméně už neřeší logiku zabezpečení před kolizí. K té totiž v minulosti, za předpokladu, že 2 lokomotivy nebyly umístěny na stejný okruh, nemohlo dojít právě díky odděleným okruhům.

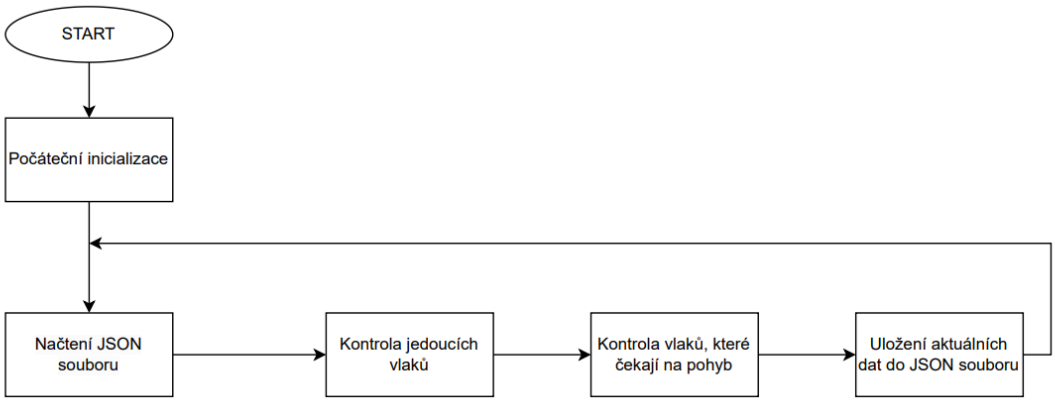
S ohledem na výše zmíněné nedokonalosti aplikace vznikla nová verze aplikace 3.0, která pracuje s novými komponentami vyvinutými do roku 2022 a která dodává aplikaci zcela nový vzhled. Tato aplikace umožňuje řízení, které využívá implementovanou logiku pro řízení modelového kolejiště a slouží pro zabezpečení před kolizí. Zároveň je v nové verzi aplikace jednoduše umožněno vytvořit nové obrazovky, což umožní zobrazení dalších funkcí, pomocí *User Controlu,* jehož zobrazení se uskuteční pouhým přidáním nového tlačítka do postranního panelu s odkazem na jeho zobrazení.

6 Návrh vhodné logiky řízení

Po provedení důkladné analýzy poslední verze hardwaru a softwaru bylo zapotřebí vymyslet vhodný způsob ovládání modelového kolejiště, který by umožňoval jednak plně manuální režim a zároveň režim, ve kterém by se uživatel staral pouze o rozjezdy jednotlivých lokomotiv, výběr cílových stanic či nastavení řídících jednotek, ale samotné řízení modelového kolejiště by prováděl systém sám. Ten musí celé kolejiště monitorovat a sledovat obsazenost jednotlivých kolejích a zároveň dbát na to, aby nemohlo dojít ke srážce vlaků či aby byly nastaveny výhybky do správné polohy.

6.1 Koncept řídící logiky modelového kolejiště

Pro vytvoření řídícího systému modelového kolejiště bylo nutné vymyslet způsob řízení, který je schopen sledovat pohyb vlaků, zabraňovat kolizím a aktualizovat pozice vlaků, ačkoliv nikdy není známá jejich přesná poloha. Z omezených možností pro řízení kolejiště bylo tedy nutné vycházet pouze z informací, zda je daný vlak v pohybu a jaké jsou odběry proudu v jednotlivých izolovaných úsecích. Z tohoto důvodu bylo vyvinut systém řízení, jehož koncept je znázorněn na obrázku 6-1.



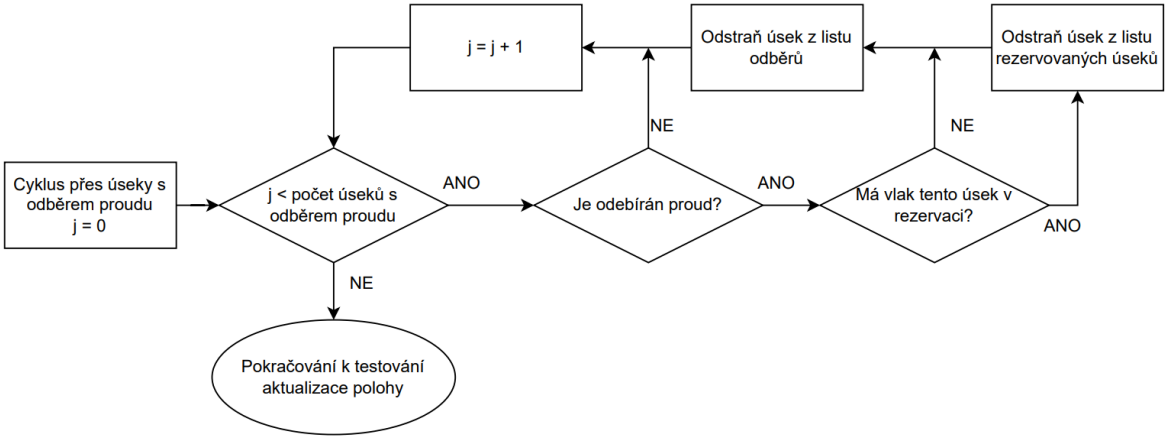
Obrázek 6-1: Koncept systému řízení modelového kolejiště

Tento způsob řízení využívá NoSQL databázi v podobě JSON databáze, která slouží k uchování aktuálních dat kolejiště. Zároveň jsou v ní uložena data pro jednotlivé vlaky včetně aktuální polohy, díky čemuž je možné dané kolejiště sledovat a řídit. Systém řízení modelového kolejiště lze obecně rozdělit do pěti základních bloků. Nejprve dojde k inicializaci všech využitých časovačů, eventů, proměnných nebo listů. Následně dojde k načtení aktuálních dat z JSON databáze do listu pro snazší práci s daty, která jsou v ní

uložena. Jednotlivé fáze řídícího systému jsou vysvětleny později, nicméně obecně dochází nejprve k testování všech jedoucích vlaků, aby se zabránilo případným kolizím. Následně dochází k testování vlaků, které teprve čekají na povolení k rozjezdu a v neposlední řadě dochází k opětovnému uložení dat do databáze. Tím je dosaženo neustálé aktualizace dat v databázi. Zároveň jsou načtena vždy aktuální data, jelikož do databáze mají přístup i jiná vlákna v aplikaci. Celá logika je prováděna ve smyčce, kterou řídí časovač a tato smyčka je vykonána vždy jednou za 250 ms. Tím je zajištěna dostatečná aktuálnost databáze při pohybu vlaků a jejich včasné zastavení v případě, že by mohlo dojít ke kolizi.

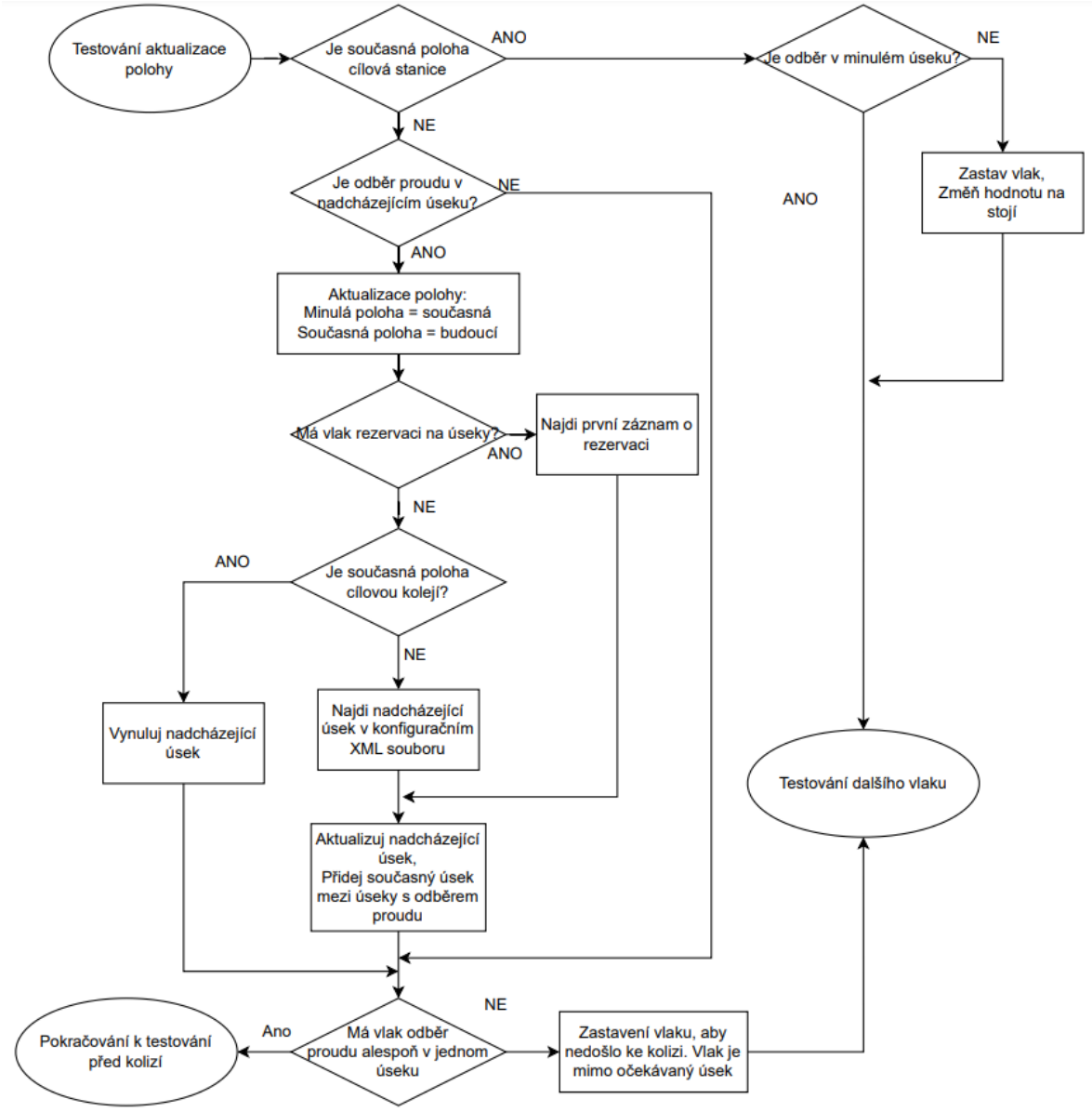
6.2 Řízení jedoucích vlaků

Po načtení databáze do lokálního listu probíhá nejprve testování jedoucích vlaků. Nejprve je provedena smyčka skrz všechny známé lokomotivy na kolejišti a testuje se, zdali vlak obsahuje příznak pohybu. Obecně lze testování jedoucích vlaků rozdělit na čtyři části.



Obrázek 6-2: Testování odběrů proudu

První část se zabývá kontrolou odběru proudů v úsecích, ve kterých byl detekován odběr proudu v minulém kroku testování. Pro testování odběrů proudu jsou jednotlivé odběry uchovány v listu, ve kterém je k dispozici ID úseku a vlak, který v daném úseku proud odebíral. Při každé smyčce testovaného vlaku jsou postupně testovány úseky a zda v nich stále dochází k odběru proudu. Když je v daném úseku odběr proudu nulový, dojde k odstranění daného úseku z listu tvořeného úseky, ve kterých dochází k odběrům proudu. Pokud tento úsek měl vlak zároveň v seznamu rezervovaných úseků, přes které musí jet u nádraží, dojde rovněž k odstranění úseku i z tohoto seznamu.



Obrázek 6-3: Testování změny polohy

Druhou částí testovací cyklu pro jedoucí vlaky je testování aktualizace polohy. Nejdříve je testováno, zda je vlak již v cílové stanici a zdali má dojít k jeho zastavení. Následuje vyhodnocení podmínky, jestli vlak odebírá nějaké proudy v nadcházejícím úseku, čímž je zjištěno, zda došlo ke změně polohy. Pokud dochází k odběru proudu, dojde k aktualizaci dat o poloze daného vlaku, kdy ze současného úseku se stane úsek minulý a z nadcházející úseku se stane úsek současný. Mimo to je aktualizován i úsek nadcházející. Ten je zjištěn buďto z rezervovaných úseků vlaku, když se vlak nachází u stanice mezi výhybkami, nebo z konfiguračního souboru, ve kterém jsou nadefinovány všechny úseky modelového kolejiště. Zároveň je v této části testováno, zda vlak odebírá proud v alespoň jednom úseku.

Pokud vlak neodebírá proud v žádném úseku, dochází okamžitému zastavení všech vlaků, aby nedošlo ke kolizi s ostatními vlaky, neboť je jeho poloha pravděpodobně neznámá.

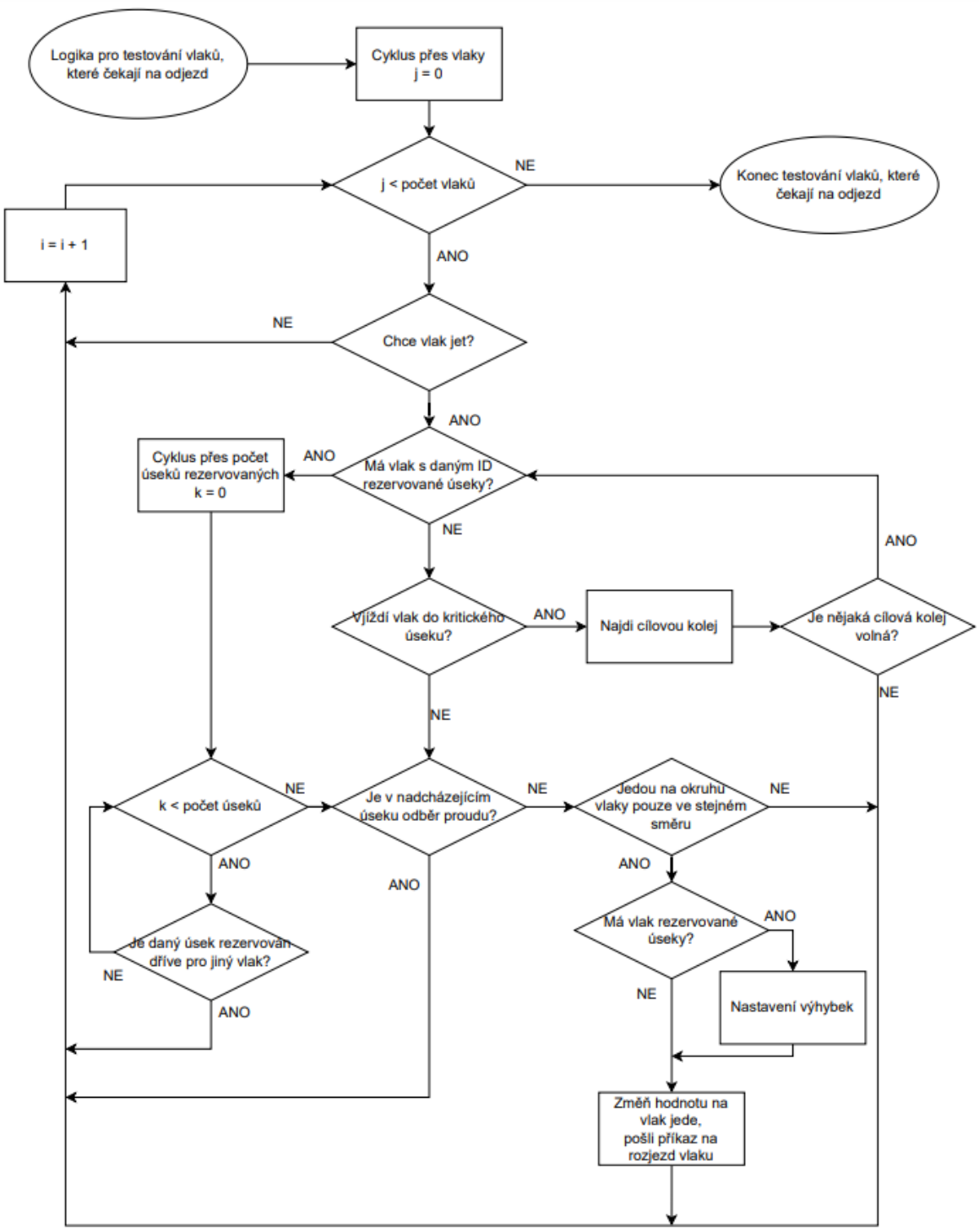
Třetí část slouží k testování vlaku před kolizí. V této části dochází k otestování několika podmínek, které zajišťují bezpečný provoz vlaků po modelovém kolejišti. První podmínka testuje, zda v nadcházejícím úseku nedochází k odběru proudu v listu uložených odběrů proudů jednotlivých vlaků. I z tohoto důvodu dochází vždy nejprve k aktualizaci polohy vlaku a až poté ke kontrole před kolizí, jelikož je tím zajištěna aktuálnost polohy vlaků. Druhá podmínka testuje, zda do nadcházejícího úseku nejede jiný vlak či jestli se v nadcházejícím úseku jiný vlak nachází. Poslední z podmínek testuje, zda je dodržen odstup od dalšího vlaku alespoň o jeden úsek, kdy je testováno, jestli minulá poloha některého z vlaků není nadcházející poloha současného vlaku. V případě, že některá z podmínek není splněna, dojde k okamžitému zastavení vlaku.

Poslední část testuje, zda vlak má vjíždět v nadcházejícím úseku do oblasti nádraží. Při splnění této podmínky dojde k vyhledání cesty na nádraží skrze jednotlivé výhybky a přidání těchto úseků mezi rezervované úseky. Cesta na nádraží je hledána dvěma způsoby. Buď je zadána cílová stanice a řídící systém sám hledá nejvhodnější cestu na nádraží na cílovou kolej, která není obsazena, nebo je zadána cílová kolej a vlak hledá úseky, přes které musí vykonat cestu, do cílové pozice. Po vyhledání cesty na nádraží dochází ke kontrole, zda jsou dané úseky již rezervovány jiným vlakem. Pokud tyto úseky nejsou rezervovány nebo pokud si je daný vlak rezervoval jako první, dojde k přenastavení výhybek do správné polohy a vlak pokračuje do cílové stanice.

6.3 Řízení vlaků čekajících na jízdu

Po otestování vlaků pohybujících se po kolejišti následuje testování vlaků, které mají příznak požadavku k pohybu. U nich se nejprve testuje, zda má vlak již rezervované úseky. V případě rezervovaných úseků dojde ke kontrole, zda má vlak tyto úseky rezervovány jako první, a tím má na tyto úseky přednostní právo, nebo jestli už stejné úseky rezervovala odlišná vlaková jednotka dříve a nelze tak zahájit pohyb. Pokud vlak nemá rezervovány žádné úseky, dochází k testování, zda má řídící systém nějaké úseky rezervovat z důvodu jízdy vlaku do stanice, respektive ze stanice. Při splnění této podmínky dojde k rezervování úseků a opětovnému testu, zda může vlak tyto úseky rezervovat. Poté následuje kontrola odběru proudu a zdali může vlak do nadcházejícího úseku vjet bez možné kolize, jelikož dalším z důvodů čekání vlaku k pohybu může být zabezpečení před kolizí při testování jedoucích

vlaků a zjištění odběru proudu v nadcházejícím úseku jiným vlakem. Pokud je odebírán v nadcházejícím úseku proud, dojde k ukončení testování možnosti pohybu daného vlaku. Jestliže v nadcházejícím úseků nedochází k odběru proudu, dojde k provedení poslední podmínky, která testuje, zda může vlak vjet na okruh s ohledem na orientaci pohybu po daném okruhu. Pokud se na daném okruhu pohybují vlaky pouze ve stejné orientaci či pokud tam žádné vlaky nejsou, dojde ke splnění všech podmínek a k rozjetí vlaku.



Obrázek 6-4: Testování vlaků čekajících na pohyb

6.4 Databáze JSON pro uchování dat

Pro uchování dat o pohybu vlaků je využita nerelační databáze typu JSON, která využívá syntaxi podobnou JavaScriptu. Tato databáze byla zvolena i s ohledem na časovou náročnost a jednoduchost práce, a proto se jedná pravděpodobně o nejjednodušší způsob uchování dat.

```
{
  "id": 9,
  "name": "Ragulin",
  "currentPosition": "u013",
  "lastPosition": "u017",
  "nextPosition": "u012",
  "move": 0,
  "speed": 11,
  "reverse": false,
  "mapOrientation": "prevConnection",
  "circuit": 2,
  "startPosition": "Karlstejn",
  "finalPosition": "Beroun"
},
```


Obrázek 6-5: Struktura záznamu pro vlak v JSONu

Na obrázku 6-5 je uvedena struktura záznamu pro každý vlak modelového kolejiště. Jedná se o databázi, která je na začátku každého testovacího cyklu vyvolaného časovačem načtena a po otestování logiky řízení dojde k jejímu opětovnému uložení. Databáze je načítána v každém cyklu znovu, protože do databáze mají přístup i jiná vlákna aplikace, která aktualizují data nepohybujících vlaků, u kterých má být zahájen pohyb.

Tato databáze je tvořena několika proměnnými, které definují data zajišťující plynulý pohyb po kolejišti. První z proměnných je proměnná *id*, ve které je zobrazeno unikátní ID každé lokomotivy, které slouží k jedinečné identifikaci vlaku. Druhou proměnnou je *name*, která nese název lokomotivy, čímž je umožněna jednodušší práce s vlaky. Zároveň jméno vlaku odpovídá názvu vlaku, které je uvedeno v konfiguračním souboru. Další proměnnou je textový řetězec *currentPosition*, který obsahuje aktuální polohu vlaku. Obdobně proměnné *lastPosition* a *nextPosition* obsahují data o minulé, respektive budoucí poloze vlaku. Tato data jsou během pohybu neustále aktualizována a díky těmto datům je možné sledovat a měnit aktuální polohu každého pohybujícího se vlaku a zároveň tato data umožňují řízení modelového kolejiště. Další proměnnou je *move*. Jedná se o číselnou proměnnou, která nabývá pouze hodnot 0, 1 a 2. Hodnota 0 značí, že vlak se nepohybuje a ani nemá žádné příkazy k pohybu, hodnota 1 znázorňuje pohybující se vlak a hodnota 2 značí, že vlak čeká na vyhodnocení podmínek pro vlak čekající na pohyb a po jejich splnění umožní řídící systém rozjezd daného vlaku. Další z proměnných je hodnota *speed*, která

značí rychlost vlaku. Tato hodnota je aktualizována pouze po přijetí žádosti na pohyb vlaku a tím je docíleno toho, že i v případě zastavení z důvodu zabezpečení před kolizí pojede vlak po opětovném započetí svého pohybu stejnou rychlostí, jaká mu byla uživatelem zadána. Po této proměnné následuje booleovská proměnná *reverse*, která slouží k informaci, jestli má vlak vykonávat pohyb vpřed či vzad. Další z proměnných je *mapOrientation*, která značí pohyb po kolejišti. Tato hodnota určuje, kterým směrem se vlak pohybuje po okruhu a díky které je možné provádět aktualizace polohy jednodušeji. Tato hodnota nabývá pouze dvou hodnot a těmi jsou orientace po kladné orientaci kolejiště (*nextConnection*), která je zvolena ve směru ze stanice Beroun směrem na Lhotu či Karlštejn, a záporná orientace po kolejišti (*prevConnection*). Další z proměnných je *circuit*. Tato hodnota uchovává číslo okruhu na kolejišti, ve kterém se vlak nachází. Zároveň je testováno, že na okruhy, které nepatří do oblasti nádraží, smí být vpuštěny vlaky pouze ve stejné orientaci. V neposlední řadě jsou zaznamenávány proměnné *startPosition* a *finalPosition*, které slouží k uchování dat o počáteční a koncové stanici pro větší množství informací o pohybu vlaku. Obě hodnoty obsahují buď název nádraží, ve kterém začínají, respektive končí, nebo konkrétní kolej. Díky tomu řídící systém během pohybu vlaku ví, zda při vyhledání cesty na nádraží má hledat nejvhodnější cestu nebo cestu, která byla zadána.

6.5 Konfigurační soubor kolejiště

Společně s vytvořením databáze bylo zapotřebí vytvořit konfigurační soubor, který by obsahoval potřebné informace o jednotlivých úsecích modelového kolejiště, informace o jednotlivých stanicích a cestách. Zároveň bylo cílem vytvořit konfigurační soubor takovým způsobem, aby aplikace nemusela být při nasazení nových hardwarových jednotek upravována a patřičné změny tak postačí provést pouze v tomto konfiguračním souboru.

Konfigurační soubor lze rozdělit na několik částí, které budou vysvětleny v nadcházející části.

6.5.1 Definování izolovaných úseků kolejiště

Izolované úseky modelového kolejiště jsou definovány v sekci *layout* a obsahují všechny informace o úsecích. Položka *id* znázorňuje jméno daného úseku, přičemž každý izolovaný úsek má svůj vlastní název a tento název je zobrazován spolu s aktuálním odběrem proudu ve vývojové konzoli. Dalšími položkami jsou *prevsec* a *nextsec*, respektive *prevsections* a

nextsections. Tyto proměnné obsahují úseky, které se nacházejí v bezprostřední blízkosti konkrétního úseku. Jedná se o úseky začínající slovem „*next*“, což značí orientaci kladným směrem po kolejišti, a okolní úseky začínající slovem „*prev*“, což značí orientaci zápornou. Jak lze vidět na obrázku, tato část může být zaznamenána dvěma proměnnými. Díky tomu je možno zjistit, zda se v nadcházejícím úseku nachází výhybky (v případě *nextsections*) či nikoliv a v tom případě je využita proměnná *nextsec*. Další z proměnných je hodnota *circuit,* která značí okruh daného úseku. Další z proměnných je hodnota *mapcoordinates*. Tato hodnota slouží k uchování souřadnic daného úseku na mapě kolejiště, díky čemuž je umožněno sledovat aktuální polohu jedoucích vlaků na mapě. V neposlední řadě je každý úsek definován hodnotami *moduleid* a *moduleposition*. Tyto hodnoty odpovídají zapojení úseku k zesilovači DCC signálu, který vždy napájí maximálně 8 úseků. První z těchto hodnot tedy odpovídá unikátnímu ID DCC zesilovače a druhá hodnota značí připojení ke konkrétnímu měřícímu kanálu zesilovače a může nabývat hodnot 0 až 7.

```
<section id = "u019" type="track">
    <prevsec>u017</prevsec>
    <nextsec>u023</nextsec>
    <circuit>2</circuit>
    <mapcoordinates>
        <xpos>685</xpos>
        <ypos>1485</ypos>
    </mapcoordinates>
    <moduleid>1</moduleid>
    <moduleposition>2</moduleposition>
</section>
<section id = "u034" type="switch">
    <prevsections>
        <prevsec1>u037</prevsec1>
        <prevsec2>u045</prevsec2>
    </prevsections>
    <nextsections>
        <nextsec1>u001</nextsec1>
        <nextsec2>u033</nextsec2>
    </nextsections>
    <circuit>0</circuit>
    <mapcoordinates>
        <xpos>490</xpos>
        <ypos>1015</ypos>
    </mapcoordinates>
    <moduleid>5</moduleid>
    <moduleposition>1</moduleposition>
</section>
```

Obrázek 6-6: Definice izolovaných úseků v konfiguračním souboru

6.5.2 Definování jednotlivých cest na nádraží

Definování jednotlivých cest na nádraží je prováděno v sekci *routes* a jsou v ní nadefinovány všechny povolené cesty mezi výhybkami z bodu A do bodu B, kde tyto body

jsou buď úseky před výhybkami, a tedy v místě, kde by měla být vyhledávána cesta, nebo v místě konečné koleje ve stanici. Druhou z možností by bylo vyhledávání cesty pomocí algoritmu vyhledávání do hloubky, nicméně i s ohledem na nezbytné nadefinování dalších parametrů, které jsou potřebné pro logiku řízení bylo od této možnosti upuštěno. Parametry *from* a *to* obsahují názvy úseků a cestu právě mezi dvěma body skrze výhybky na nádraží, respektive z nádraží. Další ze seznamu hodnot je *parts*, která obsahuje seznam všech úseků, které si vlak rezervuje jako požadované pro uskutečnění cesty. Tyto hodnoty jsou přidávány do listu rezervovaných hodnot a jedná se o hodnoty, které je systém řízení nucen kontrolovat, pokud by vlak chtěl jet skrze tyto úseky, jak již bylo popsáno výše. Hodnota *switches* reprezentuje potřebná nastavení výhybek do správné polohy pro bezpečné vykonání pohybu vlaku. Pod touto hodnotou je definováno vždy nastavení jednotlivých jednotek, kdy pro každou jednotku jsou definovány hodnoty *unit* značící ID řídící jednotky výhybek, *turnouts* značící výběr jednotlivých výhybek k přehození podle definované DCC komunikace, a hodnota *value* značí nastavení vybraných výhybek do správné polohy (viz 4.4.3).

```
<from id = "u017">
    <to id = "Beroun_kolej_1">
        <parts>
            <part1>u013</part1>
            <part2>u012</part2>
            <part3>u011</part3>
            <part4>u010</part4>
            <part5>u009</part5>
            <part6>u039</part6>
            <part7>Beroun_kolej_1</part7>
            <part8>Beroun_pred_koleji_1</part8>
        </parts>
        <switches>
            <unit1>
                <unit>1</unit>
                <turnouts>26</turnouts>
                <value>18</value>
            </unit1>
            <unit2>
                <unit>2</unit>
                <turnouts>128</turnouts>
                <value>183</value>
            </unit2>
        </switches>
        <toFinal>Beroun</toFinal>
        <fromStart>Karlstejn</fromStart>
            <circuit>0</circuit>
        </to>
```

Obrázek 6-7: Definice jednotlivých cest skrze výhybky

6.5.3 Definování jednotlivých úseků otevřené trati

Pro jednodušší práci s vyhledáváním cílových stanic jsou nadefinovány jednotlivé úseky na otevřené trati v sekci *circuits*. Zde se nachází seznam všech úseků spadajících do daného okruhu a zároveň je tím jasně znázorněno, ze kterých úseků může vlak dojet do specifické cílové stanice. Položka *items* obsahuje výčet jednotlivých úseků na daném okruhu, kdy tyto úseky jsou zadány v pořadí, ve kterém se nachází na modelovém kolejišti. V neposlední řadě jsou v této sekci zobrazeny názvy cílových stanic daného okruhu.

```
<circuits>
    <cir id = "1">
        <fromStartOutside id = "Beroun">
            <items>
                <item1>u003</item1>
                <item2>u004</item2>
                <item3>u018</item3>
                <item4>u020</item4>
                <item5>u024</item5>
                <item6>u026</item6>
                <item7>u028</item7>
            </items>
            <toFinalPosition>Karlstejn</toFinalPosition>
        </fromStartOutside>
```

Obrázek 6-8: Definice úseků pro jednotlivé okruhy

6.5.4 Definování úseků pro vyhledávání cest na nádraží

Další sekcí konfiguračního souboru jsou *criticalroutes*. Tato sekce definuje úseky pro vjezd na nádraží, které jsou testovány vždy pro pohybující se vlaky. V závislosti na současné poloze *current* a minulé poloze *last* je řídící systém schopen určit, jestli bude vlak vjíždět do oblasti nádraží a zároveň pro takový vlak nalezne na nádraží konkrétní cestu včetně rezervace jednotlivých úseků potřebných pro jízdu do stanice.

```
<criticalroutes>
    <crit id = "cr1">
        <last>u020</last>
        <current>u018</current>
    </crit>
    <crit id = "cr2">
        <last>u019</last>
        <current>u017</current>
    </crit>
    <crit id = "cr3">
        <last>u007</last>
        <current>u006</current>
    </crit>
</criticalroutes>
```

Obrázek 6-9: Definice vjezdu do kritického úseku

6.5.5 Definování cílových kolejí pro jednotlivá nádraží

V neposlední řadě se v konfiguračním souboru nachází sekce *stationTracks*, ve které jsou definovány jednotlivé cílové koleje, které se nachází na modelovém kolejišti pro jednotlivá nádraží. Tato sekce je definována hodnotou *nameOfStation*, která obsahuje název nádraží. Tento název je uživateli zároveň zobrazován pro výběr cílové stanice pohybu vlaku. Dalším z parametrů je seznam *items*, který obsahuje názvy jednotlivých kolejí modelového kolejiště. Zároveň tyto hodnoty korespondují s hodnotami, které jsou znázorněny u vyhledávání cest z, respektive do, nádraží. V neposlední řadě je v této sekci uvedena proměnná *circuit*, která značí číslo okruhu nádraží.

```
<stationTracks>
    <nameOfStation id = "Beroun">
        <items>
            <item2>Beroun_kolej_1</item2>
            <item3>Beroun_kolej_2</item3>
            <item4>Beroun_kolej_3</item4>
            <item5>Beroun_kolej_4</item5>
            <item6>Beroun_kolej_5</item6>
        </items>
        <circuit>0</circuit>
    </nameOfStation>
```

Obrázek 6-10: Definice jednotlivých kolejí pro každé nádraží

6.5.6 Definování softwarových dorazů výhybek

Poslední ze sekcí v konfiguračním souboru je sekce označená jako *turnoutStopDefinitions*. Tato sekce slouží k definování hodnot softwarových dorazů výhybek nasazených na kolejišti, jelikož každá musí mít tyto hodnoty nastaveny jinak. To je způsobeno manuální instalací každé z výhybek, kdy žádné dvě výhybky nejsou nainstalovány pod shodným úhlem a tím mají i odlišný úhel natočení pro jejich nastavení do polohy vpravo či vlevo. Nastavení softwarových dorazů jednotlivých výhybek je v podobě seznamu jednotlivých položek a každá má definované stejné hodnoty. První z těchto hodnot je *unit*. Tato hodnota definuje ID jednotky, na které se výhybka nachází. Další z hodnot je *pos*, která definuje ID kanálu, ke kterému je výhybka připojená na dané jednotce, přičemž každá jednotka výhybek má 8 kanálů, hodnota *pos* může tedy nabývat hodnot od 0 do 7. Další z hodnot jsou hodnoty *leftStop* a *rightStop*. Tyto hodnoty slouží k definování softwarových dorazů pro natočení výhybky vlevo nebo vpravo. Výhybky je sice možné přetočit na určitý úhel natočení, nicméně i s ohledem na jednoduchost jejich nastavení za provozu je nezbytné toto definování učinit. Po přijetí příkazu pro natočení vlevo se tedy

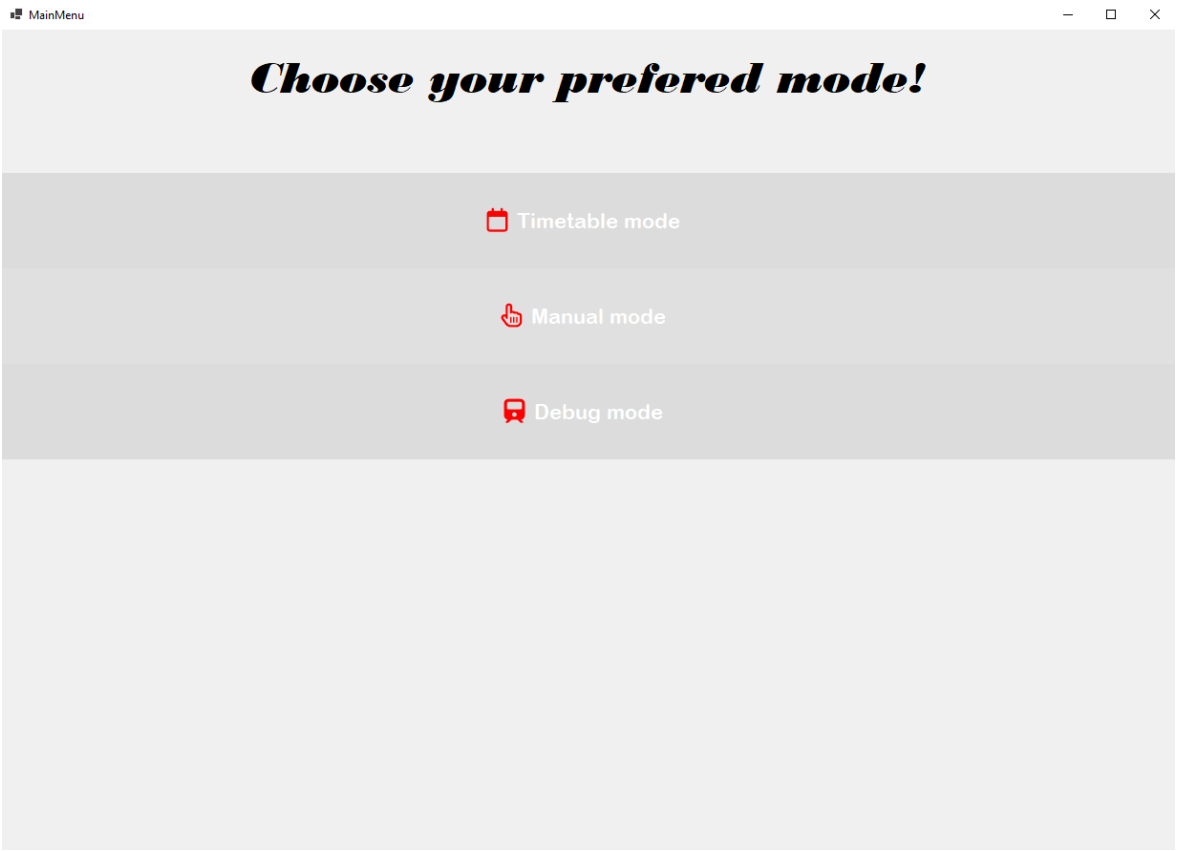
servomotor dané výhybky přetočí do úhlu, který je definován právě těmito softwarovými dorazy a obdobně tomu je i v případě přenastavení výhybky vpravo. Softwarové dorazy lze v aplikaci svévolně měnit, nicméně pro zajištění spolehlivého přetočení výhybky dochází k inicializaci softwarových dorazů již během inicializace aplikace ihned po spuštění TCP serveru. Za zmínku jistě stojí také to, že softwarové dorazy pro výhybku 6, která má hodnoty *unit = 1* a *pos = 5*, nejsou nastaveny zcela správně, jelikož tato výhybka disponuje mechanickým poškozením, a tudíž přes ni vlaky mohou jezdit pouze směrem rovně. I s přihlédnutím k tomu faktu je třeba danou výhybku i při pohybu vlaků pečlivě sledovat, jinak v tomto místě dojde k vykolejení vlaku a přetížení H-můstku, čímž dojde k zastavení provozu.

```
<turnoutStopDefinitions>
    <item1>
        <unit>1</unit>
        <pos>0</pos>
        <leftStop>90</leftStop>
        <rightStop>110</rightStop>
    </item1>
```

Obrázek 6-11: Definice softwarových dorazů výhybek

7 Software pro řízení modelového kolejiště

Pro řízení modelového kolejiště byl vyvinut software disponující třemi odlišnými aplikacemi, které uživateli nabízí odlišné způsoby řízení kolejiště. Po spuštění aplikace je uživateli nejprve zobrazeno menu vyzývající uživatele k vybrání vhodného módu řízení kolejiště, přičemž toto menu obsahuje tři tlačítka. Těmito tlačítky si uživatel zvolí požadovaný režim pro provoz kolejiště, a to buď *Debug mode*, který uživateli umožní plně řídit celé kolejiště a řídící systém zůstane mimo provoz, nebo režimy využívající řídící systém pro správný chod modelového kolejiště. Těmito režimy jsou *Timetable mode*, který umožní řídit kolejiště pomocí jízdního řádu vytvořeného ve formátu csv a *Manual mode*, ve kterém uživatel vybírá lokomotivy, které se mají pohybovat, jejich směr a stejně tak i cílové stanice. Zároveň bylo při vytváření aplikací dbáno na to, aby bylo co nejjednodušší jejich rozšíření o další obrazovky a funkce, které by uživatel mohl potřebovat. I z tohoto důvodu bylo pro aplikace kromě základní *Windows form* definující danou aplikaci a zajišťující její chování a zasílání dat, navrženo několik obrazovek přes *User control*, které jsou po stisknutí tlačítek zobrazeny. Tím je zajištěno snadné přidání dalších obrazovek, neboť stačí vždy vytvořit pouze novou *User Control* komponentu, ve které jsou definovány požadované akce, které má obrazovka vykonávat. Následně stačí přidat tlačítko do postranního panelu a připojit event na zobrazení dané obrazovky po stisknutí tlačítka. V případě vykonávání akce v komponentě, která má být zaslána systému, poté stačí vytvořit v kódu dané obrazovky nový event, který bude zpracován ve *Windows form* dané aplikace a zaslán do systému.



Obrázek 7-1: Hlavní menu aplikace

7.1 Soubory obsaženy v projektu

Celá struktura projektu je rozdělena do několika balíků. První z balíků je *TestDesignTT*, který je tvořen jednotlivými komponentami. Tyto komponenty obstarávají vzhled aplikace a také řízení modelového kolejiště pomocí jízdního řádu, manuálního řízení nebo plně manuálního řízení, nicméně tyto aplikace jsou vysvětleny v nadcházející části této kapitoly.

Druhým z balíků je *TCPServerTrainTT*, který slouží k vytvoření TCP serveru a byl vysvětlen již v kapitole 5.2.2. Tento balík je tvořen jediným souborem, který vytvoří nejprve TCP server a následně zpracovává data přijatá přes sériový port a rozesílá je jednotlivým klientům a zároveň obstarává odesílání dat z přijaté TCP komunikace do modelového kolejiště.

Třetím z balíků je *TrainTTLibrary,* který byl již vysvětlen v kapitole 5.2.1. Jedná se o balík, který se stará o vytváření jednotlivých paketů, ať už z přijatých dat od některého z klientů nebo z dat, která byla obdržena přes sériový port. Nicméně pro správný chod aplikace bylo nutno původní knihovnu upravit a doplnit. První z úprav se týkala výpočtu hexadecimální adresy z adresy jednotky a čísla jednotky, jelikož původní metoda zajišťovala

správnost dat pouze pro některá čísla jednotek, ovšem pro mnoho jiných byly vypočteny špatně. Dále bylo nutno provést úpravu tříd týkajících se řídící jednotky úseků, neboť došlo ke změně komunikačního protokolu. Všechny metody této třídy byly upraveny dle komunikačního protokolu uvedeného v [4]. Dále bylo nutno vytvořit novou třídu pro řídící jednotky výhybek, jelikož původní software práci s výhybkami nikterak neřešil, protože byl provoz řízen pouze na vzájemně oddělených úsecích. Pro tyto účely byly vytvořeny 2 třídy, kterými jsou *TurnoutInstructionPacket* pro zasílání dat řídící jednotce a *TurnoutInfoPacket* pro příjem dat od řídící jednotky. Tyto třídy byly též vytvořeny podle komunikačního protokolu uvedeného v [4]. V neposlední řadě došlo k úpravě třídy *OccupancySectionPacket*, která se stará o zpracování a ukládání dat o obsazenosti úseků. V této třídě docházelo ke špatnému vyhodnocování zpracování dat a tato data tak nebyla správným způsobem zpracována a neumožnovala řízení modelového kolejiště. Závěrem bylo upraveno načítání informací o jednotlivých úsecích modelového kolejiště z konfiguračního souboru a stejně tak načtení dat o lokomotivách z konfiguračního souboru.

Posledním z balíků je *ControlLogic*, ve kterém se nachází řídící systém. Jedná se tedy o soubory, které obstarávají nejen řízení modelového kolejiště, ale také přístup k jednotlivým informacím.

V souboru *MainLogic* je implementován řídící systém, který je popsán v kapitole 6 a který obstarává bezpečný provoz na modelovém kolejišti.

Druhým souborem v tomto balíku je *SearchLogic*. Tato třída je tvořena několika statickými metodami, které umožňují vyhledávání jednotlivých dat a parametrů v konfiguračním souboru kolejiště. V této třídě se nachází metody od vyhledávání následujících úseků, vyhledávání ID řídících jednotek na modelovém kolejišti či vyhledávání okruhů, přes které pojede vlak až po vyhledávání kritických úseků a cest na nádraží, respektive z nádraží či možné finální koleje ve stanici pro vlak, ať už se nachází v kterémkoliv místě na kolejišti.

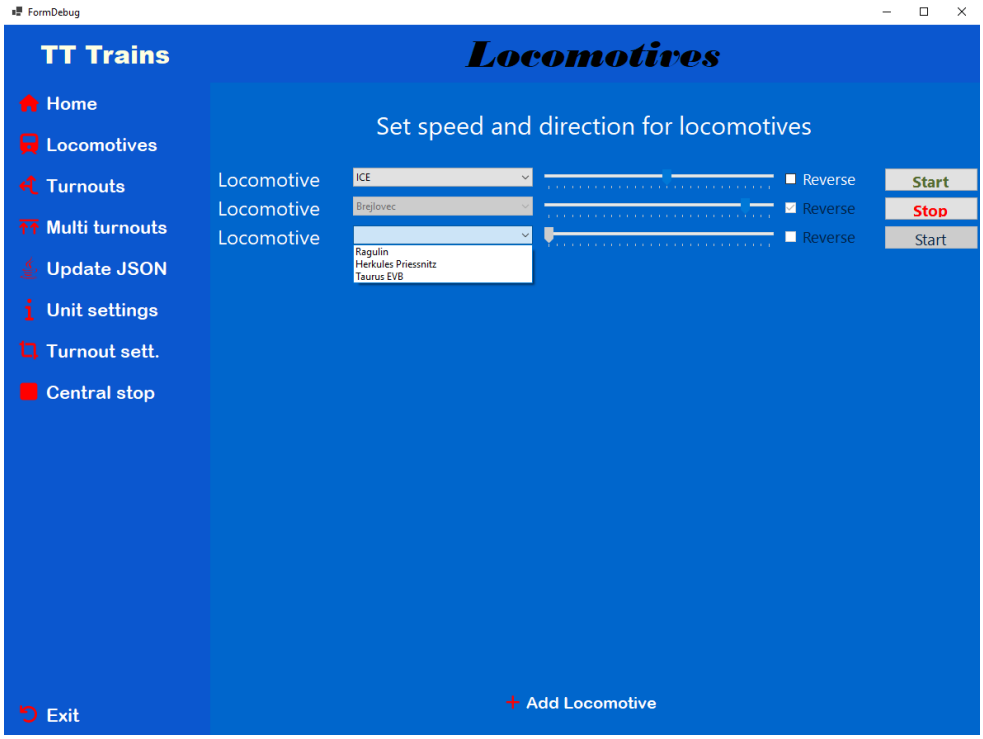
Dalším ze souborů v tomto balíků je *JsonLogic*. Tato třída má na starost veškerou práci s databází a nachází se v ní tři metody. První z těchto metod načte aktuální data z databáze pro řízení kolejiště a zobrazení aktuálních dat. Druhá metoda slouží pro uložení dat z načteného listu, čímž dojde k aktualizaci všech vlaků. Poslední metoda je použita k aktualizaci dat pouze pro vybraný vlak, přičemž v tomto případě dojde k aktualizaci dat pouze vybraných proměnných.

Zbylé dva soubory v tomto balíku jsou *conf_kolejiste* a *train_data*. První z těchto souborů je konfigurační soubor modelového kolejiště a je tvořen všemi daty o kolejišti, které byly

popsány v kapitole 6.5. Druhý z těchto souborů slouží jako databáze o datech jednotlivých vlaků a byl blíže vysvětlen v kapitole 6.4.

7.2 Řízení aplikace v plně manuálním režimu

Plně manuální řízení neboli *Debug Mode* je režim, při kterém veškeré chování na kolejišti řídí uživatel. Po spuštění aplikace dojde během inicializace jednotlivých proměnných a vytvoření instancí jednotlivých *User control* komponent i ke spuštění TCP serveru a následně k definování softwarových dorazů výhybek nainstalovaných na kolejišti. Zároveň dojde k definování a připojení eventů, které jsou vyvolány v jednotlivých komponentách a dochází k jejich zpracování v hlavním souboru *Windows form*, který je napojen na TCP server a umožňuje zasílání dat. Zároveň jsou v tomto hlavním souboru definovány akce vyvolané stisknutím jednotlivých tlačítek a stejně tak zobrazení správného uživatelského okna.



Obrázek 7-2: Obrazovka pro řízení jízdy lokomotiv v plně manuálním režimu

Uživatelské rozhraní bylo navrženo tak, aby splňovalo moderní vzhled a zároveň aby umožnilo co možná nejjednodušší případné rozšíření aplikace. Z tohoto důvodu byl zvolen postranní panel, který je neustále zobrazen a do kterého lze v případě potřeby kdykoliv přidat nová tlačítka s odkazem na nové komponenty. Pro *Debug mode* režim se v postranním menu

nachází několik tlačítek, kdy každé z nich vede na novou obrazovku, která je zobrazena uprostřed.

Tlačítko *Home* zobrazí domovskou obrazovku, na které je zobrazeno aktuální datum, čas a informace pro uživatele.

Locomotives je tlačítko, které slouží pro uvedení vybraných lokomotiv do pohybu. Jednotlivé lokomotivy jsou vybrány z konfiguračního souboru a zároveň je u nich testováno, zda již nebyly vybrány. Zároveň je řešena logika, že u pohybujícího vlaku nelze stisknout *combobox* s výběrem lokomotivy a stejně tak změnit její orientaci. Rychlost jednotlivých lokomotiv lze měnit i za jízdy, díky čemuž je manuální ovládání lokomotiv více realistické. Zároveň každá změna týkající se rozjetí, zastavení či změně rychlosti za jízdy vyvolá event, který je v hlavní formě zpracován a zaslán generátoru DCC signálu na změnu pohybu vlaku.

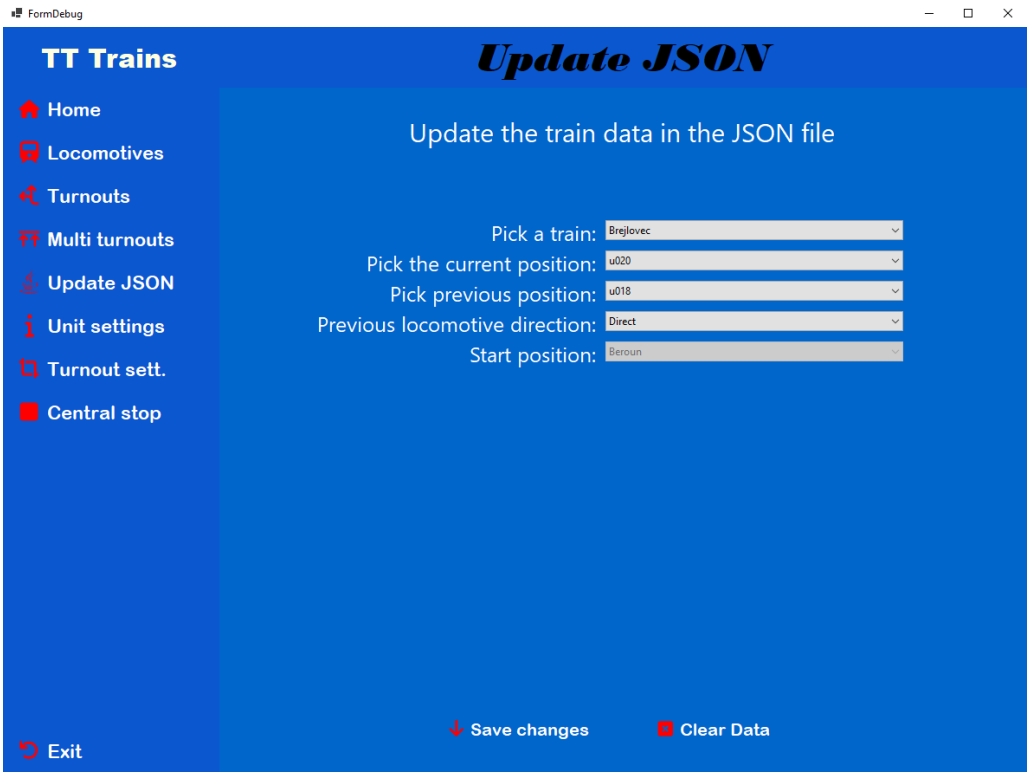
Další tlačítka v postranním menu jsou *Turnouts* a *Multi turnouts*. Obě tato tlačítka vedou na obrazovky, které slouží k nastavení výhybek, nicméně obě fungují na odlišném principu. První z těchto obrazovek nastavuje výhybky prostým vybráním ID jednotky, pozice výhybky a hodnoty, která se má nastavit na danou pozici, formou výběru hodnot z *comboboxů*. Druhá z těchto obrazovek umožňuje při znalosti jednotlivých výhybek jejich jednodušší nastavování, neboť výběr výhybek i hodnoty natočení jsou zadávány ve formě binárních čísel o délce jednoho bytu pro danou jednotku, čímž je umožněno nastavit všechny výhybky pro danou řídící jednotku naráz. Obě obrazovky jsou vybaveny mimo jiné tlačítky pro uložení dat či vymazání již navolených dat. Uložení dat opět vyvolá event, který je dále zpracován a data jsou zaslána jednotce výhybek. Nicméně pro nastavování výhybek je zapotřebí znát polohu výhybek, neboť při mylném nastavení výhybky může snadno dojít k vykolejení vlakové jednotky či ke kolizi s jinou vlakovou jednotkou.

Další z obrazovek, do které se lze dostat z postranního menu, je proklik přes tlačítko *Update JSON*. Tato obrazovka je potřebná před ukončením plně manuálního řízení kolejiště, neboť se na této obrazovce aktualizují data v databázi. Tato aplikace není napojená na řídící systém a tím nedochází k aktualizaci polohy za jízdy. I z tohoto důvodu, aby byl uživatel dostatečně informován, je při pokusu o zavření aplikace uživateli zobrazeno upozornění na nutnost aktualizace hodnot v databázi, jelikož při pokusu o jízdu může dojít ke kolizi. Řídící systém poté nezná správnou polohu vlaků a některé vlaky mají nulový odběr proudu, pokud se nedopouští žádného pohybu, což způsobí, že nebude detekována kolize ani při kontrole odběrů proudu. Tato obrazovka, stejně jako obrazovky pro výhybky, disponuje tlačítky pro uložení dat, čímž dojde k zápisu do databáze nebo k vymazání již navolených dat v okně aplikace.

Následují dvě tlačítka, která jsou zobrazena ve všech třech aplikacích a umožňují uživateli provést nastavení řídících jednotek kolejových úseků a výhybek. Jedná se o tlačítka *Unit settings* a *Turnout settings*. Na těchto obrazovkách může uživatel libovolně nastavit jednotky do požadovaného stavu, může je zapnout či vypnout, nastavit prodlevu odesílání změřeného odběru proudu nebo prodlevu před nastavením výhybek. Nicméně nejdůležitější nastavení se nachází v nastavení úsekových jednotek, neboť disponuje tlačítkem *Reset H-bridge*. Díky tomuto tlačítku lze vyresetovat můstek nacházející se na vývojové desce, protože k přetížení H-můstku může dojít v kterýkoliv okamžik a má to za následek výpadek napájení v úsecích, které daná jednotka napájí. Problém s řídící jednotkou je vždy uživateli hlášen a z odběrů proudů lze snadno vypozorovat, která z jednotek nefunguje správným způsobem.

Dalším z tlačítek v postranním menu pro *Debug mode* je tlačítko *Central Stop*, které vynutí okamžité zastavení všech lokomotiv na modelovém kolejišti.

Posledním z tlačítek je tlačítko *Exit*, které se nachází v levém spodním rohu. Cílem tohoto tlačítka je zavření aplikace pro plně manuální řízení kolejiště. Po jeho stisknutí je nejprve uživateli zobrazena zpráva vyžadující potvrzení ukončení aplikace, aby se zabránilo jejímu ukončení mylným stisknutím tlačítka. V případě potvrzení ukončení dojde okamžitě k zastavení všech vlaků na kolejišti, čímž je zajištěno, že vlaky nemohou jezdit po kolejišti po uzavření aplikace, dojde k ukončení TCP serveru a zároveň je uživateli zobrazeno okno s hlavním menu pro výběr režimu pro řízení modelového kolejiště



Obrázek 7-3: Obrazovka pro aktualizaci polohy vlaku v databázi

Ačkoliv aplikace pro plně manuální režim není napojena na řídící systém, uživatel je o většině provedených akcích informován v podobě *popup* notifikací, díky kterým vždy obdrží informaci o provedeném nastavení některé z jednotek či o vyslání příkazu lokomotivě. Zároveň je tím uživateli dána zpětná vazba o jeho krocích, kdy tyto notifikace slouží uživateli jako potvrzení o provedené akci.

V případě detekování chyby na kolejišti v podobě spadnutí jednotky či odpojení od TCP serveru je uživateli zobrazen *Message box*, prostřednictvím kterého je uživatel vždy informován o všech problémech na kolejišti, díky čemuž mu nebude umožněno pokračování v řízení, než danou zprávu potvrdí.

7.3 Řízení aplikace v manuálním režimu

Druhá aplikace, stejně tak jako předchozí, umožňuje manuální řízení kolejiště, nicméně v této aplikaci již nemá uživatel plnou kontrolu nad kolejištěm. Zároveň data pro pohyb lokomotiv nejsou zaslána rovnou generátoru DCC signálu přes TCP server, ale dochází pouze k úpravě dat pro daný vlak v JSONu včetně příznaku značícího, že vlak chce jet a možné rozjetí vlaku musí potvrdit řídící systém.

Řídící systém poté vyhodnocuje splnění podmínek, které jsou nezbytné pro bezpečný provoz po modelovém kolejišti a v případě jejich splnění je vyvolán event s informací o směru pohybu, ID vlaku a rychlosti vlaku, přičemž tato data odpovídají datům, která zadal uživatel pro pohyb daného vlaku. Společně s eventem na pohyb vlaku je též vyvolán v případě potřeby event na změnu výhybek, pokud mají být nastaveny a ve *Windows form* dojde nejprve k nastavení výhybek podle dat vyvolaných eventem a vzápětí k zaslání povelu DCC generátoru k pohybu dané lokomotivy.

Obdobně je to při detekci možné kolize či nemožnosti nastavit výhybky vlaku za jízdy z důvodu rezervace úseků již jiným vlakem. Řídící systém změní data v JSONu na „vlak chce jet“ a vyvolá tak event pro lokomotivu s daty k jejímu zastavení. Tento event je okamžitě zpracován a data jsou zaslána řídícímu generátoru.

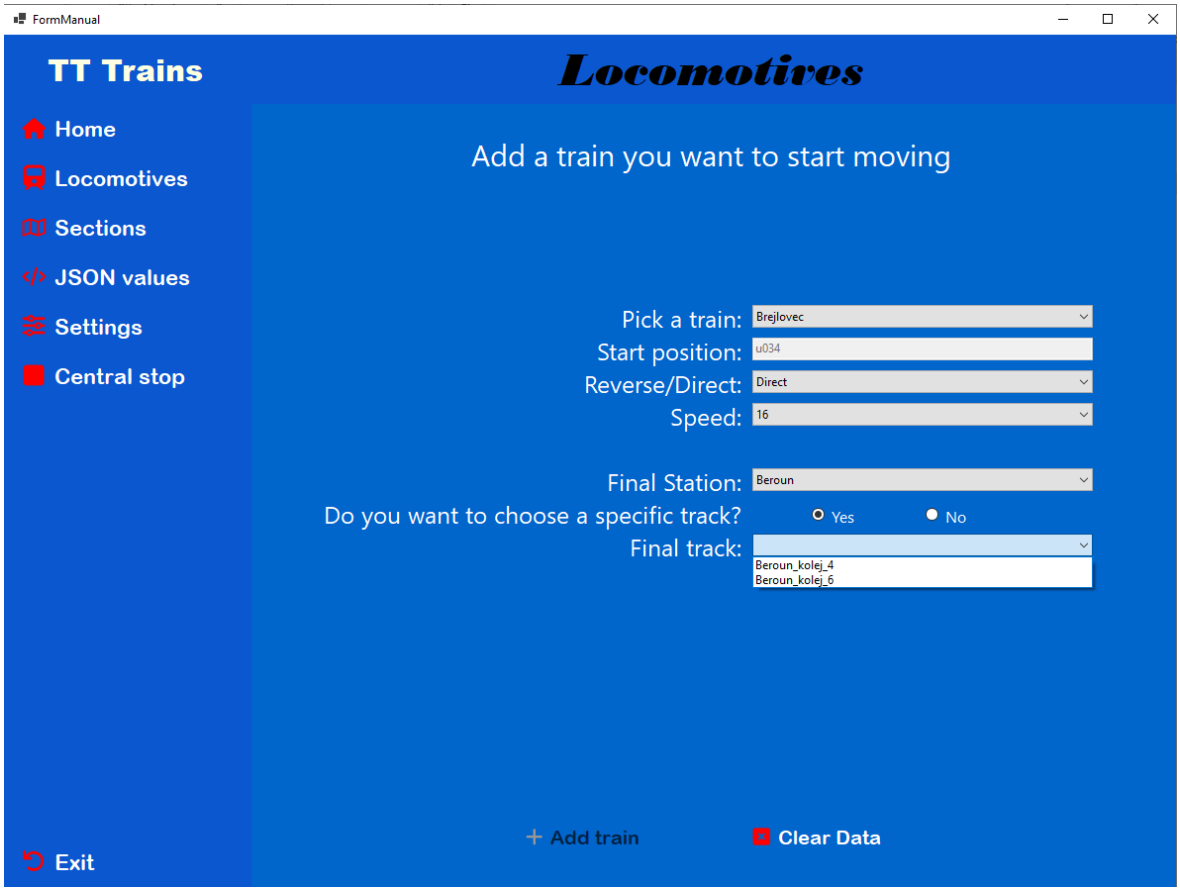
Po spuštění aplikace dochází k podobným inicializacím jako v plně manuálním režimu. Nejprve jsou opět vytvořeny instance jednotlivých *User control* komponent, spustí se TCP server a definují se softwarové dorazy. Mimo softwarových dorazů ovšem dochází k nastavení všech úsekových jednotek, aby odesílaly naměřené odběry proudu v periodě 250 ms, díky čemuž je zaručena neustálá stálost dat o odběrech proudů. Nicméně mimo inicializace a připojení eventů pro jednotlivé komponenty dochází i k vytvoření eventů pro řídící systém. Zároveň je pro správný chod aplikace a eventů od řídícího systému nezbytné vytvořenou instanci řídící logiky předat do řídící logiky, aby byla řídící logika schopna vracet eventy pro správnou instanci a aby tak tyto eventy mohly být zpracovány.

Manual mode aplikace je též tvořena tlačítky v levém postranním menu, které slouží pro ovládání aplikace. Menu je tvořeno 9 tlačítky, nicméně 2 tlačítka jsou po většinu času chodu aplikace skryta.

Prvním z tlačítek je *Home*, které zobrazí domovskou obrazovku, jako u aplikace pro plně manuální řízení, neboť se jedná o totožnou komponentu.

Druhým z tlačítek je tlačítko *Locomotives*, které slouží pro výběr vlaku, který se má pohybovat. V tomto případě ovšem není rychlost vybírána přes *trackbar*, ale pouze z *comboboxu*, jelikož s ohledem na logiku řízení není možné ovládat vlaky, které se pohybují nebo se chtějí pohybovat po kolejišti. I z tohoto důvodu je načten seznam vlaků z databáze, a nikoliv z konfiguračního souboru. Po vybrání konkrétního vlaku je uživateli zobrazena aktuální pozice vlaku a uživatel smí vybrat, zda má vlak jet vpřed či vzad a jeho rychlost. Další z hodnot je hodnota značící cílovou stanici. Hodnoty, které se uživateli objeví v poli na její výběr jsou vždy vyhledány a vyplněny s okamžikem určení směru pohybu vlaku a jsou vyhledávány na základě současné polohy vlaku, minulé polohy vlaku a orientace

vlaku. Díky tomu si uživatel může v případě potřeby vybrat požadovanou cílovou stanici a je mu tímto způsobem dodána zpětná vazba o tom, do kterých stanic může vlak ze současné polohy jet. V neposlední řadě je tato obrazovka tvořena *Radio buttony*, jejichž navolením si uživatel může zvolit požadovanou cílovou kolej v cílové stanici. Cílová kolej je vyhledávána opět z konfiguračního souboru, a to v okamžiku, kdy si uživatel vybere cílovou stanici. Cílová kolej je hledána dvěma způsoby v závislosti na poloze vlaku, a to buď z definovaných cest, pokud se vlak nachází u nádraží, nebo pomocí seznamu kolejí pro každou stanici. Pokud uživatel nezvolí cílovou stanici, řídící systém o vhodné cílové stanici rozhodne sám během jízdy. Zároveň je tato obrazovka tvořena tlačítky pro odeslání dat do řídícího systému, čímž dojde k aktualizaci dat v databázi či k vymazání aktuálně navolených dat na obrazovce.

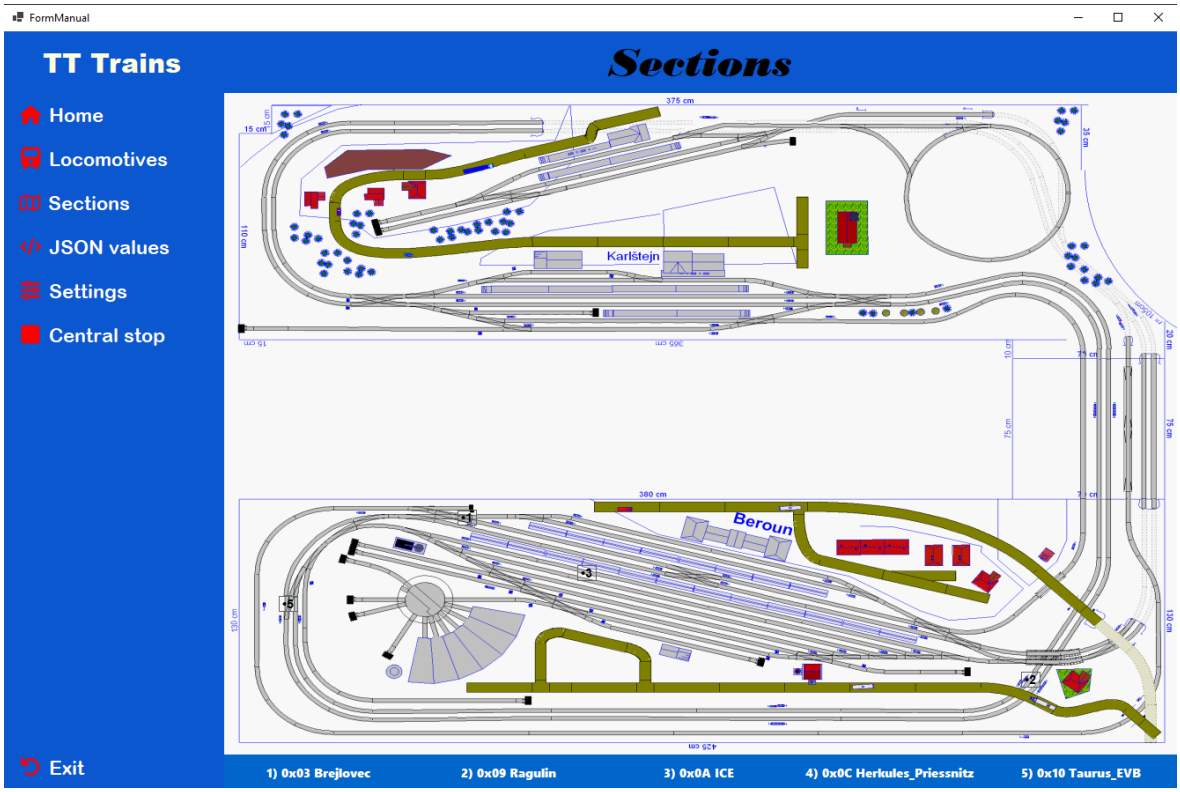


Obrázek 7-4: Obrazovka pro žádost o začátek pohybu vlaku po kolejišti

Dalšími dvěma tlačítky v postranním menu jsou *Sections* a *JSON values*. Obě tato tlačítka vedou na komponenty, které slouží k vizualizaci dat o pohybu vlaků po kolejišti, nicméně obě komponenty zpracovávají data zcela odlišným způsobem.

Položka *Sections* slouží k zobrazení mapy kolejiště, na které jsou vyobrazeny aktuální pozice vlaků, díky čemuž má uživatel přehled o poloze vlaků, aniž by je měl na očích. Tato data jsou aktualizována každou vteřinu a díky tomu má uživatel přesnou zpětnou vazbu o aktuální poloze vlaků po kolejišti. Zároveň jsou zde zobrazeny pouze vlaky, které se nachází na některém ze známých úseků modelového kolejiště. Pokud by uživatel tedy změnil databázi a aktuální poloha vlaku by neobsahovala úsek, který by byl znám v konfiguračním souboru, nebude daný vlak na mapě zobrazen.

Oproti tomu položka *JSON values* slouží k vyobrazení dat z databáze. Data, která jsou zobrazena v této tabulce, jsou pravidelně aktualizována každou vteřinu a tím je zajištěno, že uživatel obdrží vždy plně aktuální data z databáze. Zároveň má uživatel možnost sledovat, zda jsou data o poloze vlaku aktualizována správně.



Obrázek 7-5: Obrazovka znázorňující aktuální polohu vlaků

Dalším z tlačítek je *Settings*. Po stisknutí tohoto tlačítka se nezobrazí žádná nová komponenta, ale dojde k rozbalení dalších dvou tlačítek, kterými jsou nastavení úsekové jednotky a nastavení jednotky výhybek. Tato tlačítka jsou záměrně skryta, protože uživatel by správně neměl mít potřebu měnit nastavení jednotek, aby bylo zajištěno správné řízení modelového kolejiště pomocí řídícího systému.

V neposlední řadě se v postranním menu nachází opět tlačítko pro nouzové zastavení *Central Stop*, které slouží k okamžitému zastavení všech lokomotiv.

Posledním z tlačítek je tlačítko *Exit*, jehož cílem je ukončení aplikace pro manuální řízení kolejiště. Po stisknutí tohoto tlačítka a potvrzení požadavku na ukončení aplikace dojde nejprve k zastavení všech vlaků a následně dojde k vypnutí časovače řídícího systému, čímž je zajištěno zastavení řízení provozu modelového kolejiště. V neposlední řadě dojde na odpojení TCP klienta a uživatel je přesměrován do hlavního menu s výběrem režimu řízení modelového kolejiště.

Uživatel je i v této aplikaci informován o provedených akcích, rozjezdech či zastaveních lokomotiv a chybách v podobě *popup* notifikací. Tím je mu poskytnuta zpětná vazba o tom, co se děje v řídícím systému a zároveň to slouží k lepší vizualizaci aplikace. Mimo *popup* notifikací jsou uživateli zobrazeny též *Message boxy* v případě detekování chyby, ať už odpojením TCP serveru či detekce chyby na některé z řídících jednotek.

7.4 Řízení aplikace podle jízdního řádu

Poslední z vytvořených aplikací je *Timetable mode* neboli řízení modelového kolejiště pomocí jízdního řádu. V této aplikaci nemá uživatel žádnou kontrolu nad výběrem vlaků, které se budou pohybovat po modelovém kolejišti, jelikož výběr vlaků je prováděn z nahraného jízdního řádu. Stejně jako v předchozí aplikaci, i zde dochází k řízení pohybu vlaků po kolejišti řídícím systémem. Z tohoto důvodu dojde v čase odjezdu vlaku k aktualizaci dat v databázi a nastavení příznaku na požadavek k zahájení pohybu vlaku. Řídící systém poté sám vyhodnocuje, zda vlak může jet či nikoliv.

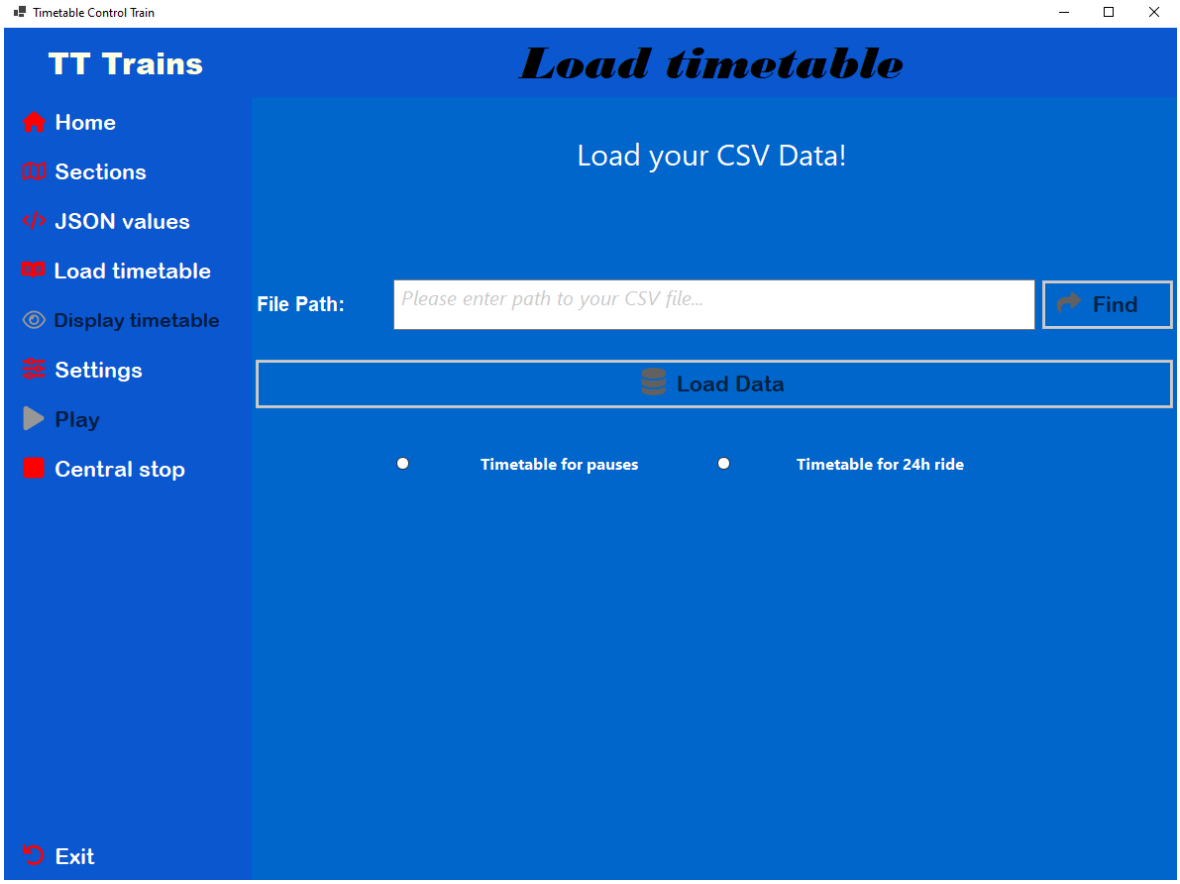
Po spuštění aplikace dochází k totožné inicializaci pro *Timetable mode* jako pro *Manual mode*, kdy nejprve jsou vytvořeny jednotlivé komponenty používané v aplikaci a je spuštěn TCP server. Následně dojde ke správnému nastavení úsekových jednotek a jednotek výhybek. V neposlední řadě jsou definovány eventy pro jednotlivé komponenty a eventy pro řídící systém včetně vytvoření jeho instance pro úspěšné zpracování eventů, které vytvoří.

Stejně jako předchozí aplikace, i tato aplikace je tvořena levým postranním panelem s jednotlivými tlačítky, pomocí kterých se lze pohybovat po modelovém kolejišti. Postranní menu je v tomto případě tvořeno 11 tlačítky, z nichž jsou 2 po většinu času skryta.

Prvním z tlačítek je opět tlačítko *Home*, které slouží k zobrazení domovské obrazovky s aktuálním datem a časem.

Druhé a třetí tlačítko jsou též shodná s aplikací pro manuální řízení kolejiště a jedná se o tlačítka *Sections* a *JSON values*. Tato tlačítka vedou na komponenty, které zobrazují aktuální polohu vlaků na mapě kolejiště či zobrazení aktuálních hodnot uložených v databázi.

Další tlačítko v postranním panelu je *Load Timetable*. Jak již název vypovídá, tato komponenta slouží k načtení jízdního řádu. Nejprve je vždy nutné navolit typ jízdního řádu, který se má načíst, jelikož jsou k dispozici 2 režimy. První režim z předem definovaných časů vytvoří časy odjezdů podle časů, ve kterých mají studenti pauzy a je šance, že může být v okolí kolejiště jisté množství pozorovatelů. Oproti tomu druhý režim načte jízdní řád s časy odjezdů bez pauz a vlaky budou neustále v provozu. Následně jsou na této obrazovce k dispozici dva způsoby načtení jízdního řádu. První spočívá ve stisknutí *Load Data*, po kterém uživatel obdrží možnost vybrat *csv* soubor ve svém zařízení. Druhá možnost spočívá v zadání přesné cesty k souboru a následném stisknutím tlačítka *Find*, po kterém dojde k načtení jízdního řádu ze souboru, který se nachází v zadané cestě.



Obrázek 7-6: Obrazovka pro načtení jízdního řádu

Další z tlačítek v postranním menu je *Display timetable*. Toto tlačítko vede na komponentu, na které je zobrazen jízdní řád a seznam odjezdu vlaků včetně časů odjezdu, jejich směru a stanic, mezi kterými vlak provede svoji cestu. Zároveň je v aplikaci implementována logika, která nepovoluje stisknutí tohoto tlačítka, pokud nebyl nahrán žádný jízdní řád a tím nejsou žádná data k zobrazení.

Dalším z tlačítek je *Settings*, které vykonává stejné akce jako v manuálním řízení kolejiště. Po jeho stisknutí jsou zobrazena další dvě tlačítka, která vedou na komponenty s nastavením úsekových jednotek a nastavení jednotek pro výhybky.

Následujícím tlačítkem je *Play/Pause*, které slouží k zapnutí, respektive vypnutí, časovače, který sleduje data jízdního řádu. Časovač každou vteřinu sleduje jízdní řád a v čase odjezdu se pokusí danou lokomotivu vypustit na danou trasu. V případě vypnutí časovače dojde k vypnutí kontroly jízdního řádu a žádné nové vlaky nebudou mít možnost začít cestu. Zároveň je toto tlačítko neaktivní, pokud není k dispozici jízdní řád.

Posledními z tlačítek jsou tlačítka *Central Stop* a *Exit*, jejichž význam není nikterak odlišný od využití v předchozích aplikacích. *Central Stop* vede k okamžitému zaslání příkazu k zastavení všech lokomotiv a *Exit* slouží k ukončení aplikace.

Kromě řídící logiky je v této aplikace implementována logika pro řízení možnosti odjezdu vlaku. Ta v době času odjezdu některého z vlaku nejprve testuje, zda zadaný vlak má shodné jméno s některým z vlaků v databázi a zdali daný vlak již nemá příkaz k pohybu. Následně je testováno, zda se vlak již nenachází v cílové stanice a má tedy vykonávat pohyb. V dalším kroku je testováno, zda vlak v zadaném směru pohybu vpřed či vzad bude vykonávat pohyb na trase z počáteční stanice do cílové stanice či jestli se nachází na otevřeném okruhu kolejiště, ve kterém v daném směru vykoná požadovanou cestu. Po splnění těchto podmínek jsou v databázi aktualizována data o požadovaném pohybu vlaku a řídící systém obstará řízení daného vlaku o zvolených parametrech dle dat z jízdního řádu.

I v této aplikaci je uživatel informován o provedených akcích v podobě *popup* notifikací, díky kterým je uživateli poskytnuta zpětná vazba o úspěšném či neúspěšném odeslání požadavku na rozjezd vlaku. Zároveň jsou uživateli též zobrazeny *Message boxy* v případě chyby aplikace, ať už jde o chybu při připojení k TCP serveru nebo o chybu některé z řídících jednotek kolejiště.

7.5 Budoucí rozšíření aplikace

Celý projekt byl vytvořen tak, aby jeho rozšíření bylo co možná nejsnadnější a nebylo zapotřebí předělávat celou aplikaci. I z tohoto důvodu dosahuje konfigurační soubor současné velikosti, jelikož obsahuje téměř všechna důležitá data, která jsou zapotřebí pro správný chod této aplikace.

Jako první je zapotřebí nastavit správnou cestu ke konfiguračnímu souboru a k databázi. Cestu k databázi je nutné pro každé zařízení upravit v souboru *JsonLogic* a zároveň je

potřeba zvolit správnou cestu k danému souboru na uživatelově zařízení. Cestu ke konfiguračnímu souboru lokomotiv je nutné upravit v souboru *Packet* ve třídě *LocomotiveInfo*. V neposlední řadě je pak nezbytné upravit správné cesty k souborům v programu *SearchLogic* a také v souboru *Packet* ve třídě *SectionInfo*.

7.5.1 Přidání nových úsekových jednotek na modelové kolejiště

Po přidání nových úsekových jednotek na modelové kolejiště je zapotřebí provést úpravu na dvou místech. První úprava je zapotřebí provést v balíku *TrainTTLibrary* v souboru *Packet*. Zde je zapotřebí provést patřičné úpravy v metodě *RecognizeUnit*. Tato metoda zpracovává data o odběrech proudu, pomocí kterých řídící systém ovládá celé modelové kolejiště. Z tohoto důvodu je zapotřebí upravit *switch-case* podmínku, pomocí které jsou jednotlivé úseky uloženy, kdy hodnota v case musí obsahovat ID řídící jednotky kolejových úseků a následně je zapotřebí vytvořit cyklus o správných hodnotách, jelikož jednotlivé úseky jsou seřazeny primárně podle ID jednotky a sekundárně podle kanálu, ke kterému je daný úsek k řídící jednotce připojen.

Dále je zapotřebí upravit konfigurační soubor kolejiště v sekci *layout*, ve které se nachází jednotlivé izolované úseky. Nově přidané úseky musí disponovat všemi potřebnými náležitostmi definovanými v kapitole 6.5.16.5. Zároveň je zapotřebí tyto nové úseky definovat v sekci *circuits*. V případě, že by se jednalo o cílové koleje, je nutné zároveň provést nadefinování těchto úseků v sekcích *circuitsToFinal*, vytvoření cest k těmto kolejím v *routes* a přidání těchto kolejí pod některou ze stanic v *stationTracks*. Zároveň je nezbytné přidat definici pro úseky, ve kterých má řídící systém provést vyhledávání cesty na nádraží v *criticalroutes*.

7.5.2 Přidání nových jednotek výhybek na modelové kolejiště

Po přidání nových řídících jednotek výhybek do modelového kolejiště je zapotřebí v konfiguračním souboru vytvořit nové cesty na, respektive z, nádraží v sekci *routes*, ve kterých jsou pro každou cestu nadefinovány výhybky, přes které vlak pojede. Tyto hodnoty jsou zde uvedeny v decimálním tvaru, nicméně po převedení na binární číslo lze snadno určit výběr správných výhybek a jejich poloh pro uskutečnění cesty do požadované polohy. Pro vytvoření nových cest je opět nutno vycházet z kapitoly 6.5.2.

Po vytvoření nových cest je nutné provést otestování výhybek v *Debug mode* aplikaci a zjistit, jaké softwarové dorazy je zapotřebí nastavit pro kterou výhybky, jelikož přesné natočení výhybky je individuální dle jejich manuálního uchycení ke kolejišti. Po zjištění správných softwarových dorazů je nutné tyto hodnoty uvést v konfiguračním souboru v *turnoutStopDefinitions*, čímž dojde automaticky po spuštění kterékoliv aplikace k jejich správnému nastavení.

7.5.3 Přidání nových vlakových jednotek na modelové kolejiště

Přidání nových vlakových jednotek na modelové kolejiště je také po mírných úpravách velmi jednoduché. První úpravu je nutné provést v konfiguračním souboru *locomotives*, ve kterém se nachází všechny vlakové jednotky, které se na modelovém kolejišti nachází. Bez jejich definování nebude možné žádné vlaky vyslat.

Druhou úpravu je nutné provést v databázi v programu *train_data.json*. Zde je potřeba vytvořit nový vlak, který bude mít shodné proměnné jako předchozí vlaky, přičemž každý vlak musí mít totožné jméno a ID jak v databázi, tak v konfiguračním souboru.

Posledním nutným krokem je přidání tohoto vlaku i do komponenty *User Controlu UCMap*, která slouží k vykreslování okamžité polohy vlaku na mapě. Zde je zapotřebí vytvořit mezi vlaky další položku a přidat jméno odpovídající jménu daného vlaku. Zároveň u zobrazovaní vlaků na mapě neprobíhá kontrola pořadí vlaků, a tudíž je nutno dodržet pořadí vlaku v databázi a v liště se seznamem vlaků.

Zhodnocení a závěr

Tato diplomová práce je zaměřena na návrh a vývoj desktopové aplikace, která obstarává řízení modelového kolejiště pomocí TCP komunikace a která je schopna řídit provoz na modelovém kolejišti a zároveň jej dokáže zabezpečit před kolizí.

Nejprve byla provedena analýza aktuálního stavu elektronických zařízení na kolejišti a aktuálního stavu softwaru. Během testování původního softwaru bylo detekováno několik neošetřených chyb, které vedly k samovolnému ukončení aplikace. Zároveň bylo vypozorováno obtížné rozšíření původní aplikace s ohledem na původní design. Při analýze elektronických zařízení na kolejišti bylo detekováno několik chyb ve firmwaru řídících jednotek, kvůli kterým nebylo zcela možné plně komunikovat s řídícími jednotkami nebo řídící jednotky nezasílaly požadované odběry proudu.

Dále je v této diplomové práci představen řídící systém, který monitoruje a řídí provoz na modelovém kolejišti. Tento systém má komplexní strukturu, která je na základě polohy, informací o pohybu vlaku a z proudových odběrů schopna řídit modelové kolejiště bezpečným způsobem, při kterém je vyloučeno riziko potenciální kolize. Řídící systém je napojen na nerelační databázi JSON, která slouží k uchování dat o vlakových jednotkách na kolejišti i po ukončení aplikací a pomocí které je možné provádět komplexnější řízení kolejiště. Řídící systém je schopen sám detekovat nesprávnou polohu vlaků i v případě, kdy došlo k zásahu do kolejiště člověkem a poloha vlaku tak byla změněna či neodpovídá poloze v databázi. Stejně tak umí bezpečně detekovat jiný vlak v následujících úsecích a zajistit okamžité zastavení vlaku.

V neposlední řadě jsou v této práci představeny tři aplikace, pomocí kterých lze řídit modelové kolejiště. První aplikace je použita pro manuální řízení, ve kterém uživatel obstarává komplexní řízení modelového kolejiště sám včetně akcí týkajících se správného nastavení výhybek. Zbylé dvě aplikace slouží pro řízení manuální a podle jízdního řádu, při kterých se o provoz na kolejišti stará výše zmíněný řídící systém sám a jsou mu pouze dodávány požadavky na pohyb vlaků. Zároveň bylo dbáno na to, aby aplikace byly i do budoucna jednoduše rozšiřitelné o nové komponenty, které by sloužily k zobrazení nových komponent či řízení nových částí kolejiště.

Cílem této diplomové práce bylo navrhnout vhodný způsob řízení modelového kolejiště včetně řídícího systému, který monitoruje kolejiště a slouží jako zabezpečovací prvek provozu na kolejišti. Funkčnost aplikací pro plně manuální řízení a pro manuální řízení kolejiště s řídícím systémem byly úspěšně otestovány a umožňují spolehlivě řídit modelové

kolejiště. Během testování bylo úspěšně otestováno i chování řídícího systému jako zabezpečovacího prvku před kolizí. Aplikaci pro řízení kolejiště pomocí jízdního řádu nebylo možné plně otestovat z důvodu nedostatečného počtu napájených úseků, díky kterým by bylo možné otestovat komplexnější provoz na kolejišti.

Mimo to nebylo možné provést otestování aplikace pro řízení kolejiště pomocí jízdního řádu z důvodu rušení, které vzniká na CAN sběrnici. Tento problém nebyl detekován při postupném vývoji tohoto softwaru, jelikož nové řídící jednotky byly k modelovému kolejišti instalovány postupně během celé doby vývoje této práce. Toto rušení bylo objeveno až v době dokončení této práce, kdy byly k modelovému kolejišti dodány poslední řídící jednotky. Toto rušení zahltí komunikaci po sběrnici CAN a neumožňuje spolehlivé zasílání příkazů řídícím jednotkám či generátoru. Jeho příčina je pravděpodobně způsobena chybou ve firmwaru řídících jednotek, nicméně jeho nápravu nebylo možné z časových důvodů zrealizovat.

Největší nedostatek této práce vidím v pevně definovaných cestách u nádraží skrze výhybky, kdy s ohledem na nedostatečný počet řídících jednotek nebylo zapotřebí implementovat komplexnější metodu pro vyhledávání těchto cest. Do budoucna by jistě bylo vhodné po připojení nových řídících jednotek aplikovat algoritmus prohledávání do hloubky, který by umožnil při větším množství jednotek komplexnější vyhledávání.

Literatura

[1] WEISSAR, Petr, ŽAHOUR, Jiří a URBAN, Ondřej. Vlaky TT. Projekty FEL. [Online] 2018. [Citace: 6. 4. 2023]. Dostupné z: https://projekty.fel.zcu.cz/index.php/Vlaky_TT

[2] ZVONAŘ, Filip. Generátor DCC signálu pro modelovou železnici. V Plzni, 2018. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta elektrotechnická. Vedoucí práce Ondřej Lufinka. Dostupné z: http://hdl.handle.net/11025/32999.

[3] BOULA, Luboš. *Řídicí jednotka přestavníků pro modelovou železnici.* V Plzni, 2020. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta elektrotechnická. Vedoucí práce Kamil Kosturik. Dostupné z: http://hdl.handle.net/11025/41714.

[4] ALBRECHT, Patrik. *Inovace elektroniky modelového kolejiště.* V Plzni, 2022. Diplomová práce. Západočeská univerzita v Plzni, Fakulta elektrotechnická. Vedoucí práce Kamil Kosturik. Dostupné z: http://hdl.handle.net/11025/48351.

[5] POLÁK, Karel. *Sběrnice CAN*. Elektrorevue. [Online] V Brně, 2003. VUT FEKT. [Citace: 6. 4. 2023.] Dostupné z: http://www.elektrorevue.cz/clanky/03021/index.html.

[6] KOSTURIK, Kamil. *Informační sběrnice* (přednáška). V Plzni, 2022. Západočeská univerzita v Plzni, Fakulta elektrotechnická.

[7] Aplikování sběrnice CAN. *Vývoj HW.* [Online] 2004. [Citace: 6. 4. 2023]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/rozhrani/aplikovani-sbernice-can.html.

[8] Mašinky Info. *Analog versus Digital.* [Online] 2015. [Citace: 6. 4. 2023]. Dostupné z: https://www.masinky.info/2011/01/analog-versus-digital.

[9] Advantages of DCC over Direct Current (Analog). *DCC Wiki – Open source.* DCC Tutorial Series. [Online] 2022. [Citace: 6. 4. 2023]. Dostupné z: https://dccwiki.com/Digital_Command_Control_Advantages_Over_Direct_Current.

[10] Digitální řízení modelové železnice – DCC. *Robodoupě.* [Online] 2016. [Citace: 6. 4. 2023]. Dostupné z: https://robodoupe.cz/2016/digitalni-rizeni-modelove-zeleznice-dcc/.

[11] National Model Railroad Association Standard. *Electrical Standards for Digital Command Control.* [Online] 9.1, 2021. [Citace: 6. 4. 2023]. Dostupné z: https://www.nmra.org/sites/default/files/standards/sandrp/pdf/s-9.1_electrical_standards_for_digital_command_control_2021.pdf.

[12] Digitální řízení modelové železnice – DCC, 2. část. *Robodoupě.* [Online] 2016. [Citace: 6. 4. 2023]. Dostupné z: https://robodoupe.cz/2016/digitalni-rizeni-modelove-zeleznice-dcc-2-cast.

[13] National Model Railroad Association Standard. *Extended Packet Formats For Digital Command Control, All Scales.* [Online] S-9.2.1, 2012. [Citace: 6. 4. 2023]. Dostupné z: https://www.nmra.org/sites/default/files/s-9.2.1_2012_07.pdf.

[14] National Model Railroad Association Standard. *Communications Standards For Digital Command Control, All Scales.* [Online] S-9.2, 2004. [Citace: 6. 4. 2023]. Dostupné z: https://www.nmra.org/sites/default/files/s-92-2004-07.pdf.

[15] LAPUNÍK, Vojtěch. *Řídicí SW pro modelové kolejiště.* V Plzni, 2019. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta Elektrotechnická. Vedoucí práce Petr Weissar. Dostupné z: http://hdl.handle.net/11025/37362.

Přílohy

Celý software je ke stažení zde: https://github.com/NejedloT/TrainsTT3/

Control Logic (package) – balík obsahující data potřebná pro řídící systém

- *JsonLogic.cs* – program pracující s databází (načítání, ukládání, částečné ukládání)
- *MainLogic.cs* – řídící systém pro modelové kolejiště
- *ProcessDataFromTCP.cs* – program, který zpracovává data z TCP komunikace, zpracovává chyby a ukládá hodnoty odběrů proudů
- *SearchLogic.cs* – program, který slouží k vyhledávání v konfiguračním souboru kolejiště
- *conf_kolejiste.xml* – konfigurační soubor kolejiště
- *train_data.json* – JSON databáze pro uchování dat o poloze a pohybu vlaků

TCPServerTrain (package) – balík obsahující program pro TCP server

- *Program.cs* – program na přijímání a odesílání zpráv TCP komunikace

TestDesignTT (package) – balík obsahující jednotlivé aplikace a jejich komponenty

- *FormMainMenu.cs* – hlavní menu aplikace, v které si lze vybrat ze tří aplikací
- *FormDebug.cs* – aplikace pro plně manuální řízení kolejiště
- *FormManual.cs* – aplikace pro manuální řízení kolejiště
- *FormTimetable.cs* – aplikace pro řízení kolejiště pomocí cizího řádu
- *Turnouts.cs* – aplikace pro přidávání dat pro přehození výhybek
- *DataTimetable.cs* – aplikace pro zpracování dat z načteného jízdního řádu
- *UCAddDebugTrain.cs* – komponenta (User Control) pro přidání, řízení a jízdu vlaku v plně manuálním řízení
- *UCAddManualTrain.cs* – komponenta pro přidání a jízdu vlak v manuálním řízení
- *UCDataLoad.cs* – komponenta pro načtení jízdního řádu
- *UCJsonDisplay.cs* – komponenta pro zobrazení dat z databáze
- *UCJsonEdit.cs* – komponenta pro aktualizování dat v databázi
- *UCTrainTimetable.cs* – komponenta pro zobrazení nahraného jízdního řádu
- *UCTurnouts.cs* – komponenta pro přehození výhybek v podobě „klikací“ obrazovky
- *UCTurnoutsMulti.cs* – komponenta pro přehození výhybek v bytové podobě
- *UCTurnoutsSettings.cs* – komponenta pro nastavení řídící jednotky výhybek
- *UCUnitSet.cs* – komponenta pro nastavení úsekové řídící jednotky
- *UCHome.cs* – komponenta pro zobrazení domovské obrazovky

TrainTTLibrary (package) – balík obsahující soubory pro vytvoření TCP serveru a paketů

- *Packet.cs* – soubor obsahující třídy pro vytvoření paketů
- *TCPBase.cs* – soubor obsahující předka TCP serveru a TCP klienta
- *TCPClient.cs* – soubor pro vytvoření TCP klienta
- *TCPServer.cs* – soubor pro vytvoření TCP serveru