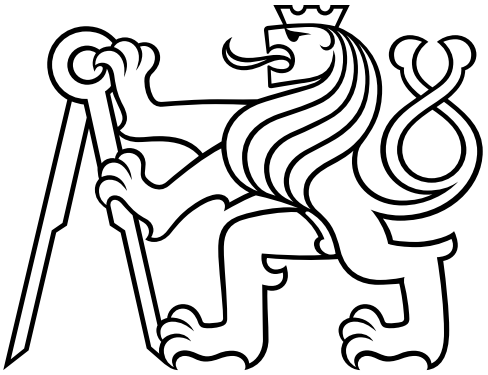
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE



Bc. Adam Eichler

NÁVRH ZPŮSOBU OVĚŘOVÁNÍ SHODY ZAŘÍZENÍ VŮČI STANDARDU VDV301

Diplomová práce

2021

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
děkan
Konviktská 20, 110 00 Praha 1

K620..... **Ústav dopravní telematiky**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Bc. Adam Eichler

Studijní program (obor/specializace) studenta:

navazující magisterský – IS – Inteligentní dopravní systémy

Název tématu (česky): **Návrh způsobu ověřování shody zařízení vůči standardu VDV301**

Název tématu (anglicky): Verification of Device Compliance with the VDV301 Recommendation

Zásady pro vypracování

Při zpracování diplomové práce se řiďte následujícími pokyny:

- navrhněte vhodný způsob provedení testů
- navrhněte a zrealizujte testovací prostředí v minimální konfiguraci
- ověřte funkčnost navrženého systému na 2 vybraných službách definovaných VDV301

Rozsah grafických prací: dle požadavků vedoucího práce

Rozsah průvodní zprávy: minimálně 55 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury:
Kabelová A., Dostálek L., Velký průvodce protokoly TCP/IP a systémem DNS, Albatros Media, 2012

Arcuri, A., RESTful API Automated Test Case Generation, IEEE Conference on QRS, Prague, 2017

Markacz, T., Testování integrací client-server, diplomová práce, ČVUT FEL, 2018

Vedoucí diplomové práce: **Ing. Milan Sliacky, Ph.D.**

Datum zadání diplomové práce: **30. června 2020**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání diplomové práce: **1. prosince 2021**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

L. S.

Ing. Zuzana Bělinová, Ph.D.
vedoucí
Ústavu dopravní telematiky

doc. Ing. Pavel Hrubeš, Ph.D.
děkan fakulty

Potvrzuji převzetí zadání diplomové práce.

Bc. Adam Eichler
jméno a podpis studenta

V Praze dne 11. srpna 2021

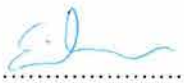
Poděkování

Tímto bych chtěl poděkovat hlavně svému vedoucímu Ing. Milanu Sliackému, Ph.D. za dlouholetou spolupráci v oblasti informačních a odbavovacích systému ve veřejné dopravě spočívající v rovnocenné výměně odborných informací a možnosti se aktivně zapojit v činnosti Zkušební laboratoře odbavovacích a informačních systémů na Fakultě dopravní ČVUT. Dále bych chtěl poděkovat svému současnému nadřízenému Ing. Tomáši Vršitému za projevenou důvěru při zavádění teoreticky nabitých znalostí do praxe Pražské integrované dopravy na Oddělení technického rozvoje Regionálního organizátora pražské integrované dopravy. (ROPID)

Čestné prohlášení

Prohlašuji, že jsem tuto písemnou studii bakalářské práce vypracoval samostatně, pouze za odborného vedení vedoucího práce Ing. Milana Sliackého, Ph.D. Dále prohlašuji, že veškeré podklady a zdroje, ze kterých jsem čerpal, jsou uvedeny v seznamu použité literatury v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací. Nemám závažný důvod proti užívání tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 30. 11. 2021



.....

Bc. Adam Eichler

Abstrakt

Autor: Adam Eichler

Název diplomvé práce: Návrh způsobu ověřování shody zařízení vůči standardu VDV301

Škola: České vysoké učení technické v Praze, Fakulta dopravní

Rok vydání: Praha 2021

Počet stran: 59

Cílem této práce je tvorba testovacího prostředí pro ověřování shody komponentů infromačního systému pro cestující s normou VDV301 pro potřeby Regionálního organizátora Pražské integrované dopravy.

Klíčová slova: OBU, OIS, MHD, VDV301, IBIS, IBIS-IP, HTTP, mDNS, DNS-SD, Qt, C++, ROPID, LCD, ethernet, ověřování

Abstract

Author : Adam Eichler

Name of master's thesis : Verification of Device Compliance with the VDV301 Recommendation

School: České vysoké učení technické v Praze, Fakulta dopravní

Year of Publication : Prague 2021

Pages: 59

The aim of this paper is to create a testing environment for checking the conformity of passenger information system components to the VDV301 recommendation used by ROPID (Regional Organiser of Prague Integrated Transport).

Keywords: OBU, Passenger information system, MHD, VDV301, IBIS, IBIS-IP, HTTP, mDNS, DNS-SD, Qt, C++, ROPID, ethernet

Obsah

1 Úvod

1.1 Předmluva

Informační technologie pronikají i do hromadné dopravy a jedna z oblastí, kde se výsledky rozvoje setkává i veřejnost, jsou informační systémy pro cestující ve veřejné dopravě. Pro zajištění poskytnutí správných informací o průběhu cesty (následující zastávky, číslo linky) cestujícímu pomocí zobrazovacích panelů je ovšem nutné technicky zajistit spolupráci jednotlivých kompoment až na úroveň komunikačního protokolu. Konkrétně pro potřeby Pražské integrované dopravy je nuté vytvořit způsob, jak zařízení otestovat ještě před vstupem do pravidelného provozu. Tato práce se zaměřuje na zařízení komunikující pomocí jednotného a otevřeného protokolu VDV301

1.2 Cíl práce

Cílem práce je vytvořit prostředí pro testování palubního počítače a LCD displeje pro cestující. Toto prostředí bude pomáhat při procesu certifikace zařízení do provozu v PID.

2 Vymezení rozsahu práce

Tato práce se zaměřuje na pouze částečné ověření shody s normou VDV301 a to jak manuálním, tak i automatickým způsobem.

3 Co je IBIS-IP

IBIS-IP je souhrn specifikací popisujících komunikaci a chování komponent informačního a odbavovacího systému. Jedná se o komunikační protokol fungující jako další vrstva nad TCP/IP. Protokol specifikuje norma VDV301, jejíž označení je často používáno místo názvu protokolu samotného. Protokol nahrazuje předchozí řešení s názvem IBIS definovaném normou VDV300. Toto zastaralé řešení kromě komunikačního protokolu deifinovalo i parametry specializované komunikační sběrnice, založené na sériové lince. Nové řešení IBIS-IP je tedy pouze protokolem, IBIS je zároveň název pro protokol i sběrnici.

IBIS-IP Vznikl za několika účely:

Unifikace V současném prostředí existuje mnoho různých dodavatelů komponentů OIS a tyto prvky často nejsou mezi dodavateli kompatibilní na úrovni HW či protokolu. Toto vedlo ke vzniku různých bilaterálních dohod mezi dodavateli řídících a podřízených zařízení, kdy jeden z účastníků poskytne komunikační protokol a druhý jej implementuje, ovšem za podmínky podpisu dohody o mlčenlivosti (NDA).

Otevřenost V současnosti jsou tyto komunikační protokoly obchodním tajemstvím a není možné získat dokumentaci a jednoduše vyrobit kompatibilní zařízení. Dokumentace k IBIS-IP je volně dostupná na internetu v němčině a angličtině včetně všech implementačních poznámek. [4]

Rozšiřitelnost Současné komunikační protokoly často narážejí na limity, co se týče objemu přenášených dat a budoucí rozšiřitelnosti. U rozhraní fungujících na principu sériového rozhraní jde často o následující omezení na straně HW:

1. poloduplexní spojení
2. striktní master-slave architektura
3. nedostatečný datový tok (IBIS, 1200 bps)

Dále existují omezení na straně protokolu:

1. pevně daný počet/pořadí řídicíh znaků
2. omezení znakové sady (IBIS - 7bitová abeceda)

Existují ovšem i protokoly založené na ethernetu, které jsou nějak omezené. Například jsou to takové, které využívají pro všechny přenosy UDP datagramy a v nich binárně

kódovaná data. Jelikož se fragmentace UDP datagramu do více IP datagramů silně nedoporučuje, je limitem velikosti obsahu velikost jednoho IP datagramu - UDP hlavička, tzn. 65 527 bytů. [9]. Toto například vede k omezení počtu zasílaných zastávek.

Potokol IBIS-IP využívá pro přenos XML formát a není tedy problém rozšířit specifikaci o další tagy. Toto je například využito v rozšíření pro Pražskou integrovanou dopravu, které je použito pro tuto práci.

Modernizace Mnoho výhod plyne už z využití ethernetu. Lze eliminovat veškerou specifickou kabeláž pro IBIS, případně RS485 a místo ní využít metalické vedení ethernetu různých kategorií, které je levnější, odolné vůči rušení a lze jej sdílet například s kamerovým systémem. IBIS-IP nijak neomezuje provoz ostatních služeb ve stejné síti a tím pádem není nutné jej nějak oddělovat.

3.1 Detekce zařízení

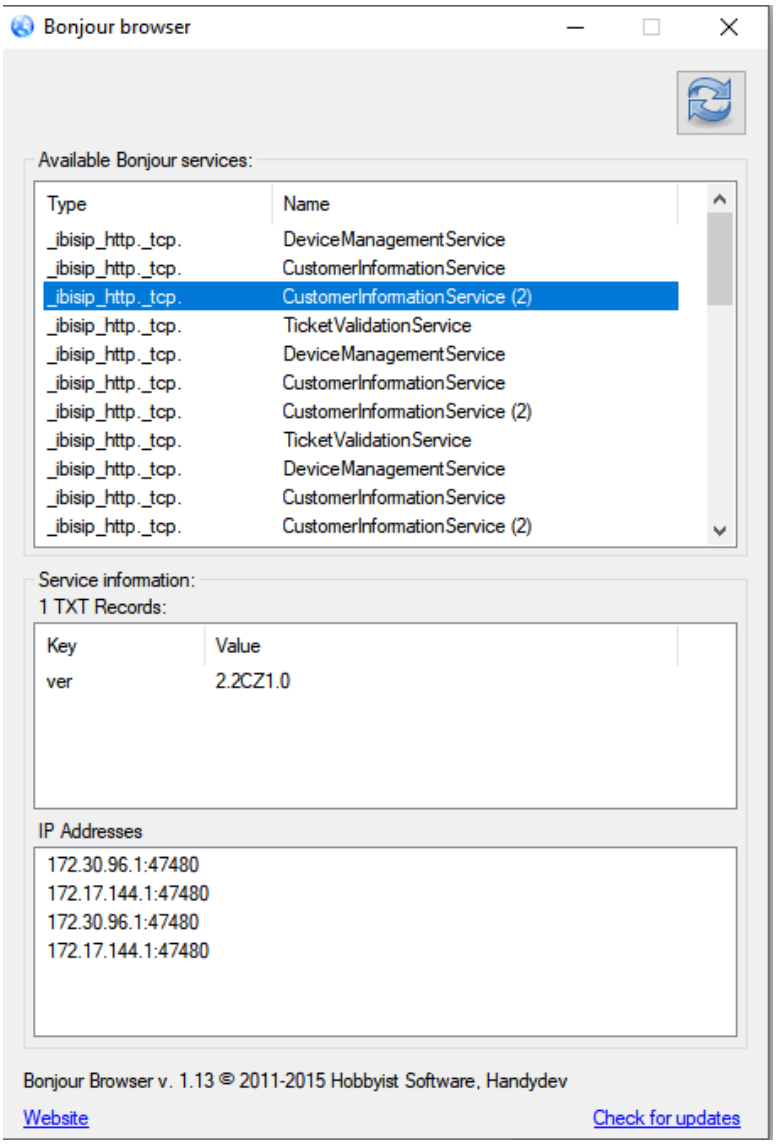
IBIS-IP využívá technologii DNS-SD (implementace jsou známé jako Bonjour a Avahi), která umožnuje publikovat poskytované služby zařízením do všech zařízení v síti.

Tato služba je často využívána mimo odvětví OIS, například při instalaci ovladačů síťových tiskáren, kdy instalátor dokáže automaticky zjistit IP adresu a port síťové tiskárny.

U IBIS-IP umožňuje DNS-SD zařízením, jako jsou zobrazovací panely, najít adresu palubního počítače a příhlásit se k odběru.

Naopak palubní počítač dokáže automaticky detekovat veškerá připojená zařízení, neboť všechna zařízení povinně publikují DeviceManagementService.

Bonjour Browser Pro kontrolu služeb publikovaných do DNS-SD lze na Windows využít freeware nástroj *Bonjour browser*. V tomto případě jsou IBIS-IP služby zobrazeny vícenásobně, pravděpodobně z důvodu chyby v programu *Bonjour browser*. Dále chybí ostatní služby dostupné v lokální síti, zřejmě vlivem firewallu na konkrétním počítač.



Obrázek 1: Bonjour browser, Windows (autor, 2021)

Avahi-browse V operačních systémech založených na Linuxu lze využít balík Avahi, a konkrétně příkaz *Avahi-browse*. Ten počet služeb zobrazuje správně. Zároveň jdou vidět i ostatní služby ostatních zařízení v síti:

- Tiskárna (CANON MG3550)
- Ikea zigbee Gateway (gw-44916029b4e7)
- síťové úložiště Synology (megaServer)

```
pi@fluiddA8:~ $ avahi-browse -a
+   eth0 IPv6 Canon MG3500 series                  _canon-bjnp1._tcp    local
+   eth0 IPv4 Canon MG3500 series                  _canon-bjnp1._tcp    local
+   eth0 IPv4 Canon MG3500 series                  UNIX Printer         local
+  wlan0 IPv6 Canon MG3500 series                  Internet Printer     local
+  wlan0 IPv4 Canon MG3500 series                  Internet Printer     local
+   eth0 IPv6 Canon MG3500 series                  Internet Printer     local
+   eth0 IPv4 Canon MG3500 series                  Internet Printer     local
+   eth0 IPv6 Canon MG3500 series                  _scanner._tcp        local
+   eth0 IPv6 Canon MG3500 series                  Web Site             local
+  wlan0 IPv6 gw-44916029b4e7                      _coap._udp           local
+  wlan0 IPv4 gw-44916029b4e7                      _coap._udp           local
+   eth0 IPv6 gw-44916029b4e7                      _coap._udp           local
+   eth0 IPv4 gw-44916029b4e7                      _coap._udp           local
+  wlan0 IPv6 TicketValidationService              _ibisip_http._tcp    local
+  wlan0 IPv6 CustomerInformationService (2)       _ibisip_http._tcp    local
+  wlan0 IPv6 CustomerInformationService           _ibisip_http._tcp    local
+  wlan0 IPv6 DeviceManagementService              _ibisip_http._tcp    local
+   eth0 IPv6 DeviceManagementService              _ibisip_http._tcp    local
+   eth0 IPv6 CustomerInformationService           _ibisip_http._tcp    local
+   eth0 IPv6 CustomerInformationService (2)       _ibisip_http._tcp    local
+   eth0 IPv6 TicketValidationService              _ibisip_http._tcp    local
+   eth0 IPv6 megaServer                           _device-info._tcp    local
+   eth0 IPv6 megaServer                           Web Site             local
+   eth0 IPv6 megaServer                           Microsoft Windows Network local
```

Obrázek 2: Avahi-browse, Raspberry Pi OS (autor, 2021)

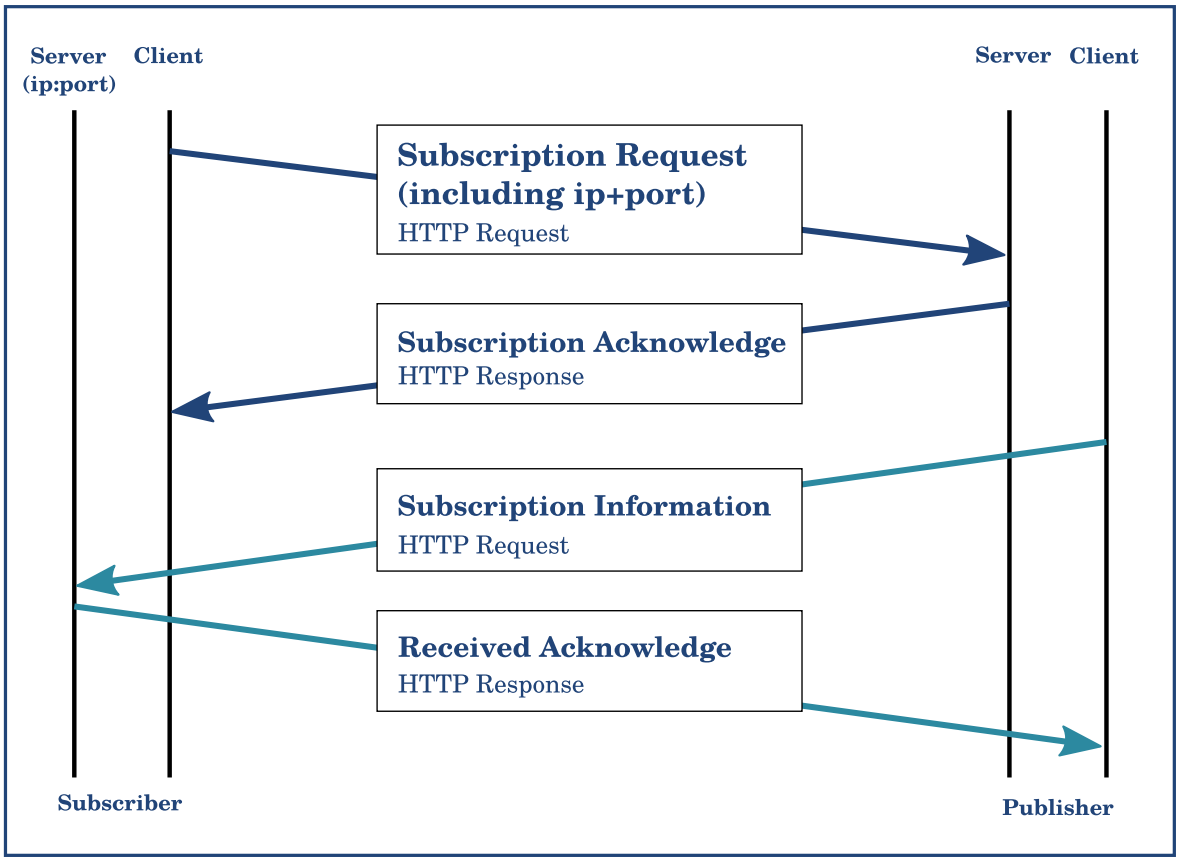
3.2 Publisher-Subscriber architektura

Celý systém je postavený na principu odběru informací od poskytovatele.

Odběratel (subscriber) odešle požadavek na odběr informací do poskytovatele (publisher). Ten si zapíše odběratele na seznam odběratelů konkrétní skupiny informací, který následně odesílá informace při jejich změně, případně periodicky.

Toto je zásadní rozdíl oproti master-slave architektuře, protože každé zařízení může být zároveň odběratelem i poskytovatelem, dokonce několika služeb. Například palubní počítač je poskytovatelem služby *CustomerInformationService* a zároveň odběratelem služby *DeviceManagementService*, kterou poskytuje zobrazovací panel.

3.2.1 Průbeh odběru



Obrázek 3: architektura Publisher - Subscriber (CDV, 2014)

3.3 Služby

Veškeré informace jsou poskytovány pomocí služeb. Různá zařízení můžou poskytovat/odebírat různé služby, které se liší obsahem.

Tyto služby se dělí na dvě skupiny dle typu paketů, jakými jsou odesílány.

3.3.1 UDP služby

UDP datagramy jsou využívány pro informace, které jsou odesílány periodicky a je důležitější kratší interval odesílání informací a není nutné potvrzení doručení.[9] Takové služby jsou zpravidla využívány pro menší objem odesílaných informací, které se musí vlézt do jednoho UDP paketu.

Testování těchto služeb není v této práci uvažováno.

GNSSLocationService V1.0 Tato služba slouží k odesílání informace o poloze, tedy hlavně souřadnic a typu GNSS signálu, který je právě přijímaný.

Z této služby může čerpat palubní počítač pro potřeby AVL i vnitřní zobrazovací LCD panel pro zobrazení polohy vozidla na mapě.

3.3.2 TCP služby

Tyto služby jsou zaměřené na přenos informací, u kterých je důležité doručení každé zprávy a přenos většího objemu informací.[9]

Toto umožnuje protokol TCP, který obsahuje mechanismy pro kontrolu doručení paketů. Dále lze dělit data do více TCP paketů, čehož se využívá například ve službě *CustomerInformationService*, která využívá pro přenos informací rozsáhlé XML soubory.

CustomerInformationService Tato služba poskytuje veškeré informace potřebné pro informování cestujících. Jedná se například o sled zastávek, stav otevření dveří, číslo linky.

TicketValidationService Důvodem vzniku této služby je poskytování zjednodušených informací pro validátory/označovače cetovních dokladů. Jedná se hlavně o aktuální tarifní pásmo projížděného tarifního bodu a o možnost vypnout označovač pri přepravní kontrole.

DeviceManagementService Tato služba slouží ke kontrole stavu zařízení a umožňuje i jednoduché ovládání periferií, například nucený restart zobrazovacího panelu na základě příkazu z palubního PC.

3.4 Rozšíření na verzi 2.2CZ1.0

Přesto, že je norma VDV301 popisující prokol IBIS-IP velice rozsáhlá, vznikla pro neměcké prostředí a tak bylo nutné vytvořit pro použití v Pražské integrované několik úprav.

Veškeré úpravy postupovaly dle požadavku dokumentu *Odbavovací a informační zařízení ve Vozidlech PID - Autobusy Srpen 2021*[3] a jeho starších verzí.

Jedním z příkladů doplnění jsou příznaky zastávek zobrazované na LCD zobrazovačích jako piktogramy. V původní německé variantě normy není k dispozici ani základní příznak zastávky na zastavení. Tato funkcionalita nebyla vyžadována, neboť v německy mluvících zemích jsou zpravidla v režimu na znamení všechny zastávky a ty, které na znamení nejsou, tedy nebylo nutné odlišit při zobrazení pro cestující.

Dalším specifickým požadavkem pro PID je rozlišování názvů zastávek podle pozice zobrazovacího panelu podle jeho polohy ve vozidle. Bylo to tedy nutné přidat nové elementy pro každý panel zvlášť.

```
<Destination>
        <DestinationFrontName>
                <Value>NÁDR. KLÁNOVICE -</Value>
                <Language>cz</Language>
        </DestinationFrontName>
        <DestinationFrontName>
                <Value>SEVER ~</Value>
                <Language>cz</Language>
        </DestinationFrontName>
```

```
        <DestinationSideName>
                <Value>N. Klánovice-sever ~</Value>
                <Language>cz</Language>
        </DestinationSideName>
        <DestinationRearName>
                <Value/>
                <Language>cz</Language>
        </DestinationRearName>
        <DestinationLcdName>
                <Value>Nádr.Klánovice-sever</Value>
                <Language>cz</Language>
        </DestinationLcdName>
        <DestinationInnerName>
                <Value>Nádr.Klánovice-sever</Value>
                <Language>cz</Language>
        </DestinationInnerName>
</Destination>
```

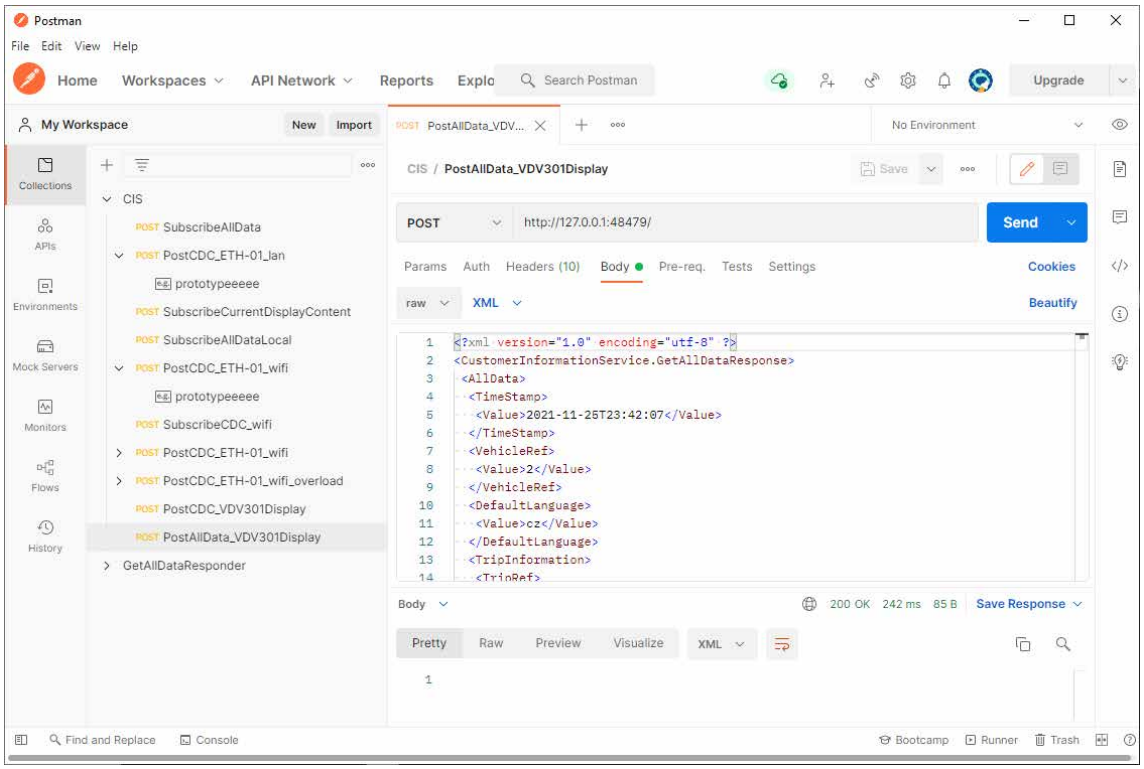
4 Možnosti testování

Vzhledem k tomu, že je komunikační protokol IBIS-IP postaven na běžných webových technologiích, je vytvoření testovacího rozhraní z principu možné a lze tak učinit pomocí volně dostupných zdrojů bez potřeby využívat nějakého autorsky/obchodně chráněného know-how. Vzhledem k plánovanému postupu certifikace - tzn. testování spolupráce testovaného zařízení vůči referenčnímu zařízení, které je kompletně simuluje skutečnou protistranu, jedná se o tzv. mocking.[10]

4.1 Postman

Jednou ze zvažovaných možností je freeware program *Postman*, určený hlavně pro testování HTTP a REST API. Je k dispozici v desktopové verzi a webové verzi. Webová verze vyžaduje připojení k internetu a proto není vhodná pro offline testování v lokální síti.

Hlavní funkcí programu je možnost odesílat libovolné HTTP požadavky. Lze tedy jednoduše zvolit cíl, kam má být požadavek odeslán, hlavičku požadavku a obsah těla.



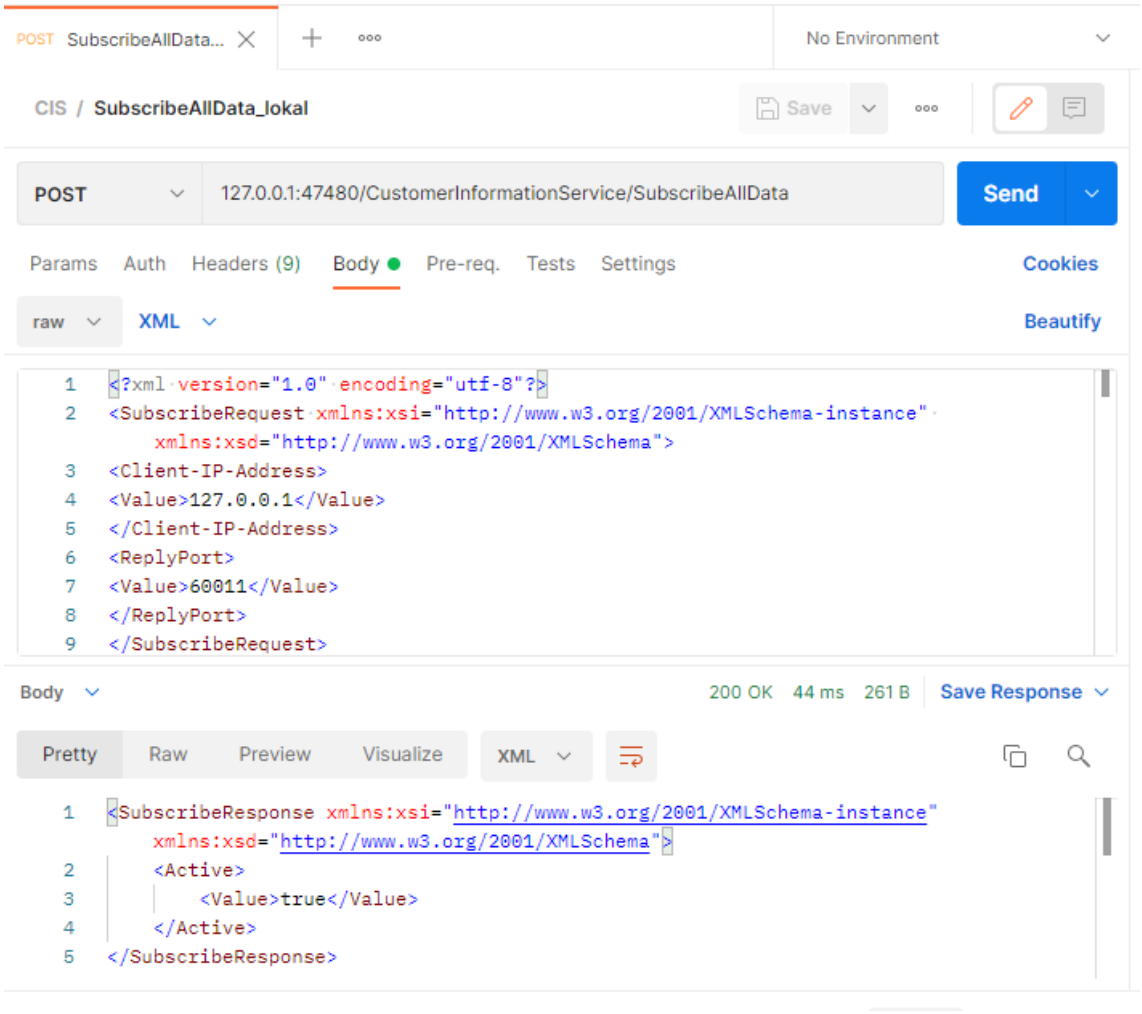
Obrázek 4: Rozhraní programu Postman (autor, 2021)

Tímto postupem lze ověřit pouze část funkcionality IBIS-IP zařízení, neboť je nutné manuálně nastavit IP adresy testovaných zařízení a není tedy možné ověřit funkčnost DNS-SD procesů.

Přesto, že program Postman není zvolen jako výsledné řešení testování zařízení, výrazně pomohl při tvorbě programu *VDV301tester*, hlavně při ověřování funkcionality HTTP serveru a HTTP klienta.

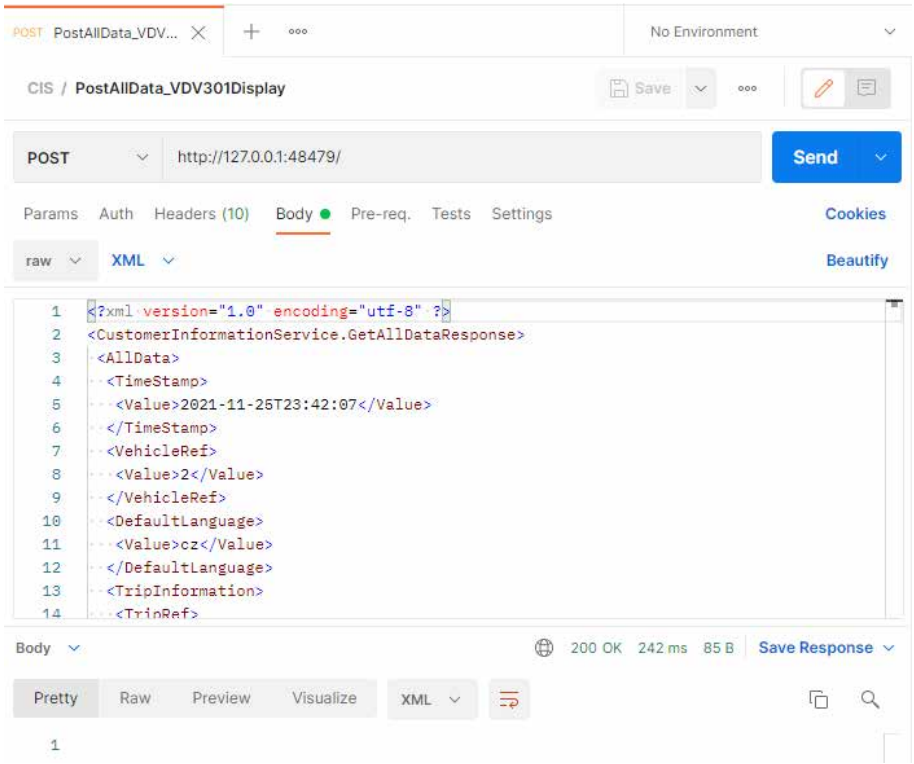
V programu lze uložit několik různých šablon pro requesty. Jako nejužitečnější se ukázaly tyto dva:

Přihlášení k odběru Tato část simuluje situaci, kdy se zobrazovací panel(Postman) přihlašuje k odběru dat ze služby CustomerInformationService běžící na palubním počítači (testované zařízení). V obsahu requestu zobrazovací panel oznámí, na kterou adresu a port má palubní počítač data zasílat a ze které XML struktury mají byt data generována. V odpovědi (dolní polovina obrázku) palubní počítač oznámí, zda se odběr povedl, nebo ne.



Obrázek 5: Příkaz pro zahájení odběru (autor, 2021)

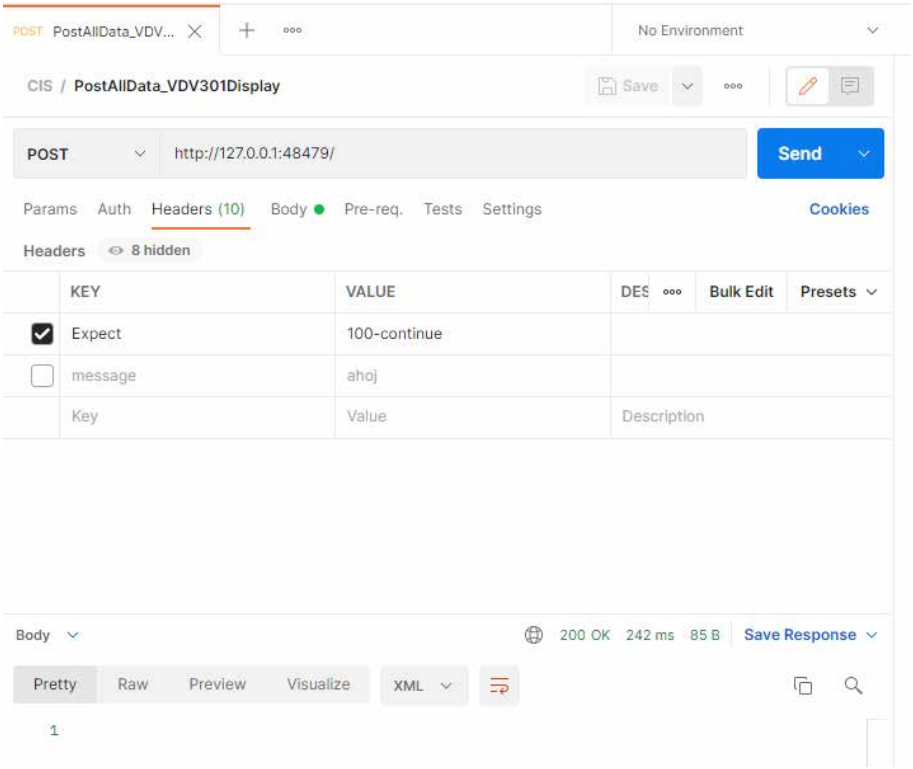
Odeslání dat Druhá část requestu simuluje situaci, kdy palubní počítač (Postman) odesílá již konkrétní data na zobrazovací panel (testované zařízení).



Obrázek 6: Odeslání konkrétních dat na panel (autor, 2021)

Zde je tělo odpovědi prázdné, odpověď pouze v hlavičce obsahuje HTTP stav *200 OK*.

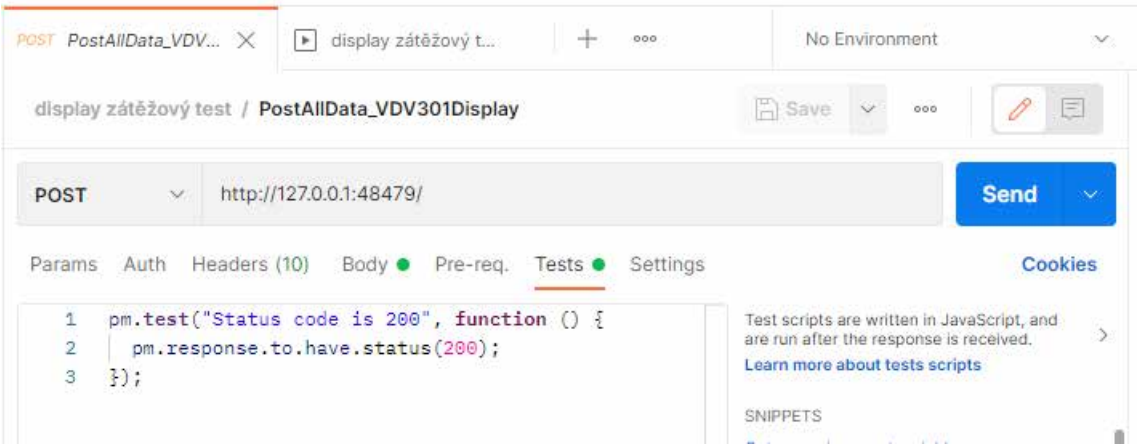
Postman umožňuje kromě těla změnit i hlavičku požadavku. Pro oba requesty byl nastaven pouze parametr *Expect: Continue*. Tento parametr zajistí, že se klient nejdřív zeptá serveru, zda je připraven přijmout velký balík dat.



Obrázek 7: Nastavení hlavičky pro oba požadavky (autor, 2021)

4.2 Zátěžové testy

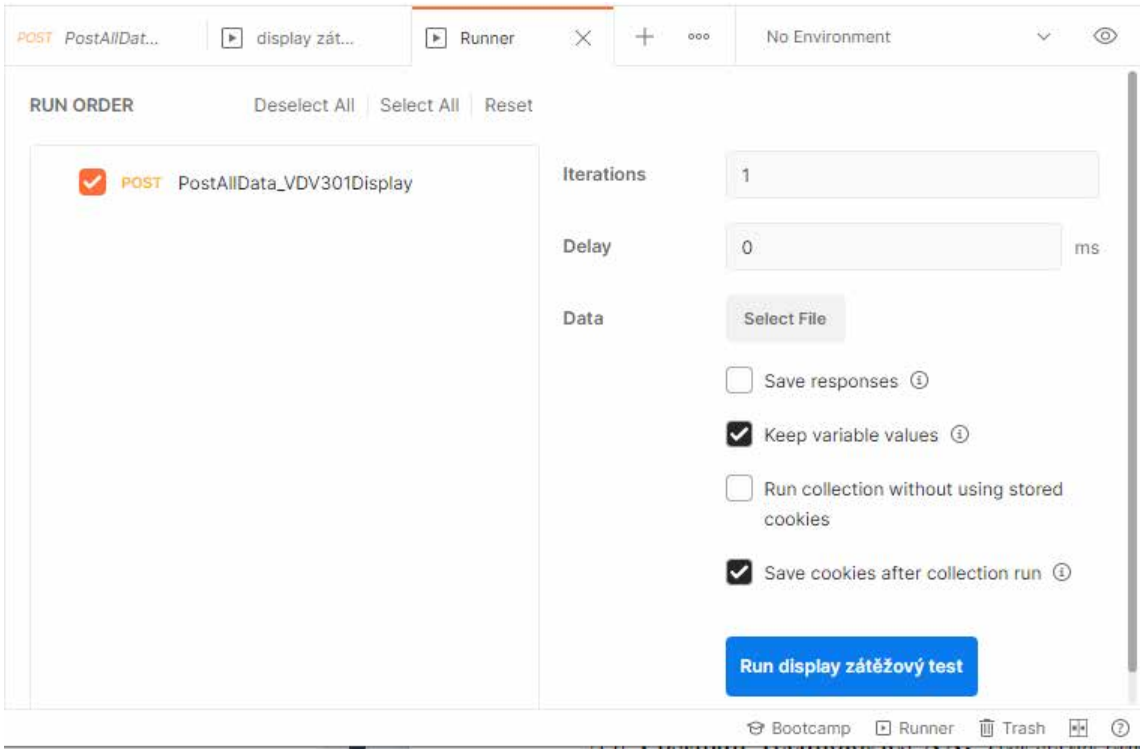
Kromě správně nastavených procesů pro odesílání dat je na místě testovat i spolehlivost HTTP serveru zařízení. Jedním ze způsobů, jak toto docílit, je odesílání většího počtu HTTP požadavků sledování, kolik požadavků se vrátilo se správným výsledkem (HTTP stav 200 OK). Nástroj Postman přímo obsahuje skriptovací rozhraní využívající jazyk Javascript. Přímo jedním z příkladů uvedených v dokumentaci[14] je ověřování odpovědi na požadavek.



Obrázek 8: Zadání testovacího skriptu do programu Postman (autor, 2021)

Po zadání testovacího skriptu k jednomu z požadavků je třeba spustit funkci programu **Collection Runner**, která provádí postupně všechny příkazy ve sbírce testů (Collection).

Výhodou je možnost nastavení počtu opakování testu, které program provede a dokonce s možností nastavení časového rozestupu jednotlivých testů.



Obrázek 9: Nastavení počtu iterací testu v programu Postman (autor, 2021)

Při takovýchto základních testech je rychlost implementace základního testu bezkonkurenční ve srovnání s funkčně srovnatelnou implementací vlastního programu.

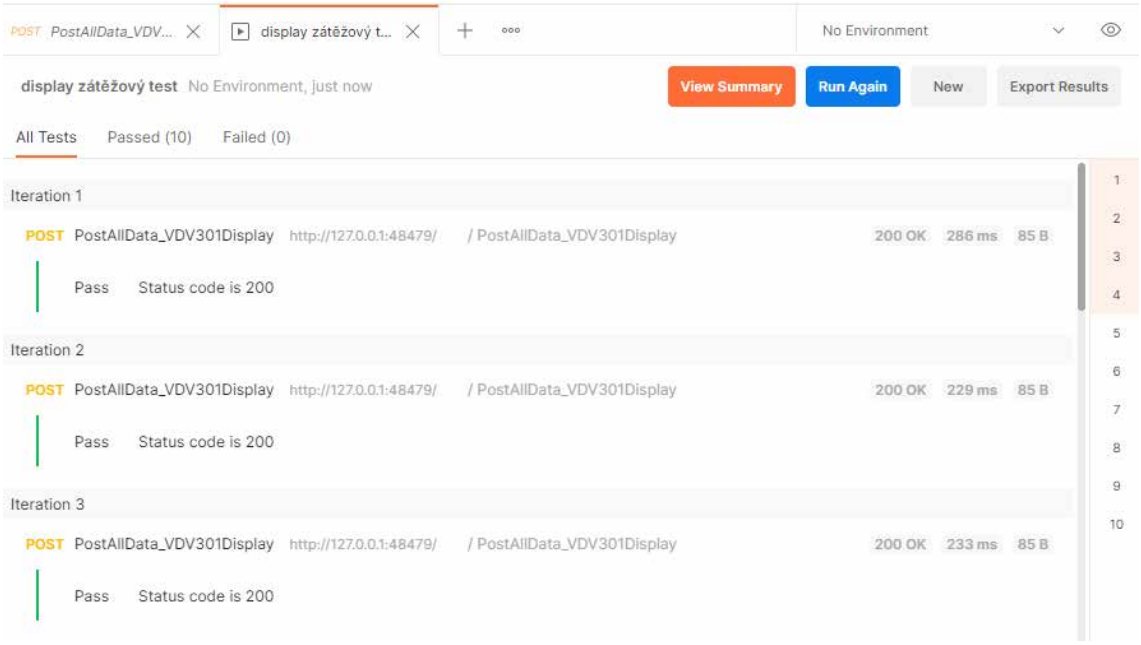
Pro složitější testy se již implementátor musí podbrobněji seznámit se syntaxí jazyka JavaScript a s dokumentací předdefinovaných funkcí, které pro skriptování Postman nabízí.

Průběh testu:

Obrázek 10: Průběh v programu Postman (autor, 2021)

Po testu program zobrazí výsledky. Součástí testu je i informace, jak dlouho (v mili-

sekundách) trvalo serveru odeslat odpověď.



Obrázek 11: Výsledky testu v programu Postman (autor, 2021)

Tyto testy byly prováděny v programu Postman for Windows, Version 9.2.0, win32 10.0.19044 / x64.

4.3 Vlastní program

Vlastním programem je v tomto případě myšlena situace, kdy pro danou potřebu vzniká od základů program nový, který je naprogramován na zakázku přímo pro daný účel.

4.3.1 Důvody pro volbu

Výhodou vlatního programu je možnost vytvořit program na míru z hlediska funkčního i uživatelského.

Lze zkombinovat více různých technologií do jedné - existující testovací programy často pracují jen s jednou. Například databáze a XML, HTTP server a DNS-SD.

Nevýhodou je časová/finanční náročnost takového projektu. V případě nemožnosti realizovat problém již hotovým programem se může jednat o nevyhnutelnou variantu.

5 Testovací scénáře

Všechny testy jsou koncipované jako samostatné a pokud selže jakákoliv dílčí část/fáze testu, je výsledek testu NEPROŠEL a další dílčí části už se neprovádí.

5.1 Testování počítače

- Vstupní XML
- Očekávaný výstup po odběru
- Displej hledá pal. PC
- Displej se hlásí k odběru
- Displej obnoví odběr/neobnoví odběr
- Automatické srovnání XML
- Různé verze služby
- Přetížení requesty
- Nastavení času
- Schválně podvržený request
- Různé hlavičky

5.2 Testování displeje

Pro testování zobrazovacích zařízení jsou navrhnuty následující scénáře:

5.3 Manuální testy

Při těchto testech je nutná součinnost operátora, který kontroluje správnost vizuální reprezentace vstupních dat pohledem na zobrazovací plochu zařízení.

5.3.1 Test přepínání stavů

Při tomto testu je sledována reakce na polohové stavy:

- BeforeStop
- AtStop
- AfterStop
- BetweenStop

Dle grafického manuálu PID [3] je možné vizuálně zkontrolovat pouze stav *AtStop* (šedé podbarvení řádku) a správné přepínání zastávek v seznamu následujících zastávek.

Připojení zařízení Operátor připojí zařízení k systému.

Kontrola přihlášení Operátor otevře seznam odběratelů a zkontroluje, zda na seznamu přibylo nové zařízení.

Volba linkospoje Operátor zvolí vhodný linkospoj dle následujících kritérií:

- dostatečné množství zastávek (aspoň 10)
- minimálně dvěma zastávkami, které byly v backoffice označeny jako nácestné

Simulace jízdy Operátor pomocí programu mění polohu na linkospoji a sleduje správné podbarvování řádku s aktuální/příští zastávkou.

Kontrola nácestných zastávek Při průběhu jízdy operátor kontroluje, zda z běžícího řádku nácestných zastávek mizí již projeté zastávky.

Konečná zastávka Operátor kontroluje, zda při příjezdu do zastávky dojde k zobrazení obrazovky *konečná zastávka*.

Obrázek 12: Očekávaný výstup u obrazovky Konečná zastávka [3]

5.3.2 Test příznaků

Připojení zařízení Stejně jako předchozí test.

Kontrola přihlášení k odběru Stejně jako předchozí test.

Výběr linkospoje Podobně jako u předchozího testu, ovšem zde jsou jiná kritéria pro výběr linkospoje:

- obsahuje zastávky s příznaky (minimálně jedna z nich je i nácestnou zastávkou)
  - přestup metro A
  - přestup metro B
  - přestup metro C
  - přestup metro D
  - přestup přívoz
  - přestup lanová dráha
  - zastávka na znamení
- obsahuje minimálně jednu zastávku s kombinací příznaků
- obsahuje cíl s příznakem

Kontrola zobrazení U tohoto testu operátor kontroluje existenci piktogramů pro všechny druhy příznaků. Dále kontroluje, zda lze piktogramy zobrazit na následujících pozicích:

- současná/příští zastávka
- následující zastávky
- nácestné zastávky
- cíl

5.3.3 Test zobrazení přestupů

Následující test ověří, zda dochází ke korektnímu zpracování přestupů z MPVnet/Golemio.

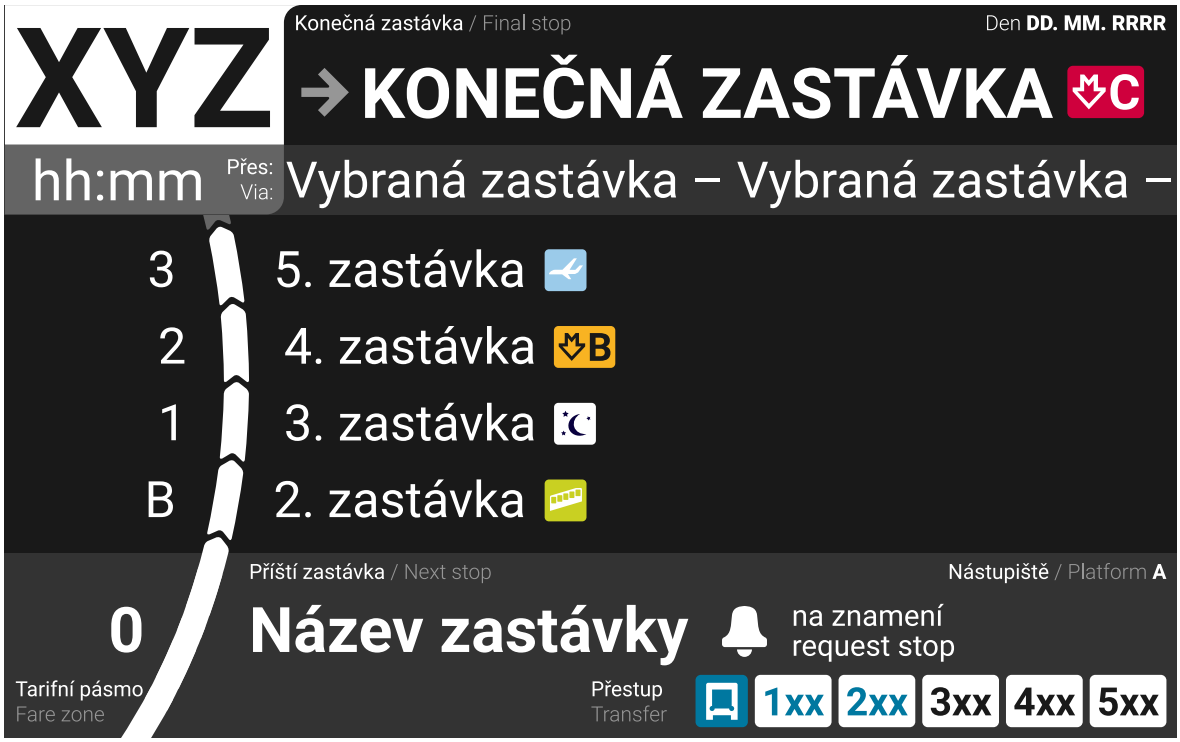
Připojení zařízení

Volba linkospoje Pro relevantní zobrazení je nutné zvolit linkospoj, který obsahuje aspoň jeden přestupní uzel, ideálně pojížděný více druhy linek dle dopravního prostředku (autobus, tramvaj, metro).

Test je nutné provádět dopoledních až odpoledních hodinách, kdy je hustota spojů dostatečná.

Operátor zároveň zajistí, že testovací program má přístup k internetu.

Volba správného stavu Přesun na zastávku s mnoha přestupními vazbami (ideálně tramvaj, autobus, metro).

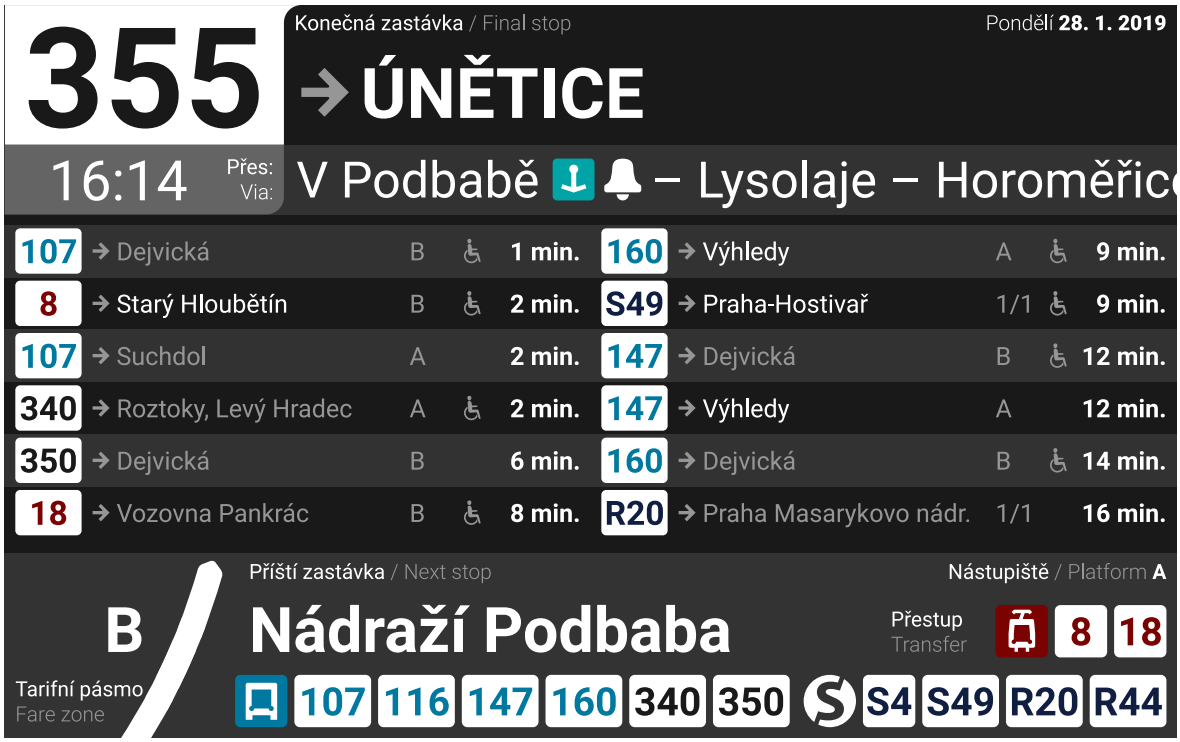


Obrázek 13: Očekávaný vystup při zobrazení příznaků zastávek [3]

Kontrola Operátor zkontroluje zobrazení přestupů vůči vygenerovanému XML. Vygenerované XML zobrazí pomocí webového prohlížeče, do kterého zadá adresu

`http://127.0.0.1:47480/CustomerInformationService/GetAllData`.

Zařízení musí zobrazit správný počet přestupů. Všechny, pokud je počet menší než maximum a maximum, pokud je skutečný počet větší. Zařízení musí správně vyhodnotit i zobrazení odjezdů a příznak nízkopodlažnosti.



Obrázek 14: Očekávaný vystup při zobrazení přestupů [3]

5.4 Automatické testy

5.4.1 Test přihlášení k odběru

Spuštění služby Prvním krokem je nastartování služby *CustomerInformationService* (CIS) ve VDV301tester.exe, což spočítá hlavně v publikování služby CIS do DNS-SD.

Čekání na přihlášení k odběru Spustí se odpočet času, po který VDV301tester čeká, až se nějaké zařízení přihlásí k odběru. Pro test je nutné, aby bylo připojené pouze jedno zařízení. Při testu je zalogována adresa a port odběratele.

Pokud vyprší časovač dříve, než se nějaké zařízení přihlásí k odběru, je výsledkem *FAIL*.

Reakce na přijatá data VDV301tester odešle odběrateli data a očekává pozitivní odpověď. Vše je opět kontrolováno časovačem.

5.5 Návrh testu služby DeviceManagementService

Služba DeviceManagementService slouží ke správě zařízení. V případě této služby má smysl provádět detekci více připojených zařízení přes DNS-SD a sledovat, zda dochází ke správnému pojmenování zařízení. Další možností je nucený restart zařízení přes tuto službu a kontrola, zda zařízení opravdu nabootovalo a za jaký čas.

Kontrolu, zda všechny zařízení na síti tuto službu poskytují, je v současnosti možné

otestovat manuálně pomocí programu VDV301Display, který zobrazuje veškeré IBIS-IP služby, které v síti detekoval.

6 Příprava na realizaci

6.1 Správa verzí

Pro práci byl zvolen verzovací systém Git a to z několika důvodů.

1. je využíván sdružením VDV
2. široká podpora komunity uživatelů
3. je multiplatformní
4. lze obsluhovat pomocí příkazové řádky

Verzovací systém pomáhá organizovat celý proces vývoje software. Sleduje jednotlivé změny ve zdrojovém kódu a umožňuje nahrávat kód na síťová úložiště (např. GitHub).

Výhodou takového řešení je možnost se vrátit k předchozí funkční verzi.

Zdrojové kódy programů, které jsou výsledkem této práce, jsou k dispozici právě na GitHubu:

VDV301tester: `https://github.com/frutabruta/VDV301tester`

VDV301display: `https://github.com/frutabruta/VDV301display`

6.2 Volba operačního systému

Drtivá většina vývoje proběhla na notebooku Lenovo ThinPad L390 Yoga s operačním systémem Windows 10 firmy Microsoft a to hlavně za účelem zajištění kompatibility se služebními počítači organizace ROPID.

Ověření funkčnosti proběhlo i na jednodeskovém počítači Raspbbery Pi 3 (architektura ARM, operační systém Raspbian Buster) a na notebooku Lenovo ThinkPad X220 s operačním systémem Ubuntu 20.04 LTS). Zdrojový kód lze tedy zkompilovat beze změn pro různé operační systémy i architektury procesoru.

6.3 Volba vývojového prostředí

Pro tvorbu tvorbu programu bylo využito prostředí QT creator s verzí Qt 5.15.2 a jazykem C++. Tato kombinace se osvědčila už při tvorbě bakalářské práce. [1]

Kombinace plně splňuje následující požadavky:

- Vývojové prostředí musí být multiplatformní
- Musí obsahovat nástroje pro tvorbu grafického rozhraní

- Dostupnost knihoven
    1. HTTP server
    2. Bonjour publisher/browser
- spolupráce s verzovacím nástrojem Git

Požadavek na multiplatformní prostředí plyne ze snahy ověřit funkčnost navrhovaného řešení i na slabším HW, přibližně odpovídající výkonově reálným dodávaným zařízením. Pro toto ověření byl využit jednodeskový počítač Raspberry Pi 3.

Kvůli využití knihovny *qthttpserver* bylo nutné využít aktuální verzi Qt, tedy 5.15. Toto způsobilo problémy při kompilaci pro jednodeskové počítače Raspberry Pi 3 a 4. Na těchto počítačích běží operační systém Raspberry Pi OS (dříve Raspbian), který umožňuje instalaci nových předkompilovaných balíčků z repozitářů pomocí balíčkovacícho systému *APT* (Advanced Packaging Tool)

6.4 Instalace prostředí

Windows Pro Windows je instalace jednoduchá. Je nutné stáhnout instalační soubor OpenSource verze Qt. V průběhu instalace je možné vybrat konkrétní verzi Qt, kompilátoru atd. Pro Windows není nutné Qt kompilovat.

Linux Pro operační systém Linux je situace složitější. Kvůli rozmanitosti linuxových distribucí není v silách *The Qt Company* zajistit kompatibilitu předkompilovaných verzí, a tak jsou vydávány pouze zdrojové kódy.

U některých distribucí je možné nainstalovat již zkompilované Qt pomocí repozitářů. Jedná se často o starší verze.

V distribuci Ubuntu je to 5.12.8, v distribuci RaspberryPi OS je to dokonce ještě starší verze 5.11.3.

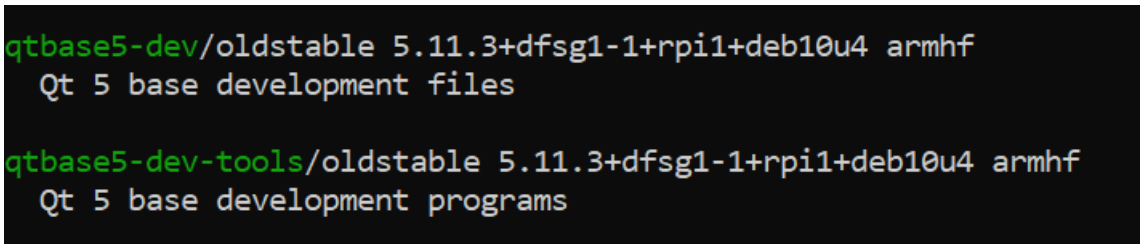
```
qtbase5-dev/focal 5.12.8+dfsg-0ubuntu1 amd64
  Qt 5 base development files

qtbase5-dev-tools/focal 5.12.8+dfsg-0ubuntu1 amd64
  Qt 5 base development programs
```

Obrázek 15: Výpis repozitáře systému APT v Ubuntu 20.04 (autor, 2021)

6.4.1 Kompilace pro Raspberry Pi 3

Vzhledem ke stáří verze Qt v repozitáři Raspberry Pi OS (knihovna qthttpserver vyžaduje aspoň verzi 5.12.0 [15]), je běžně nutné Qt zkompilovat ze zdrojových kódů, ovšem



Obrázek 16: Výpis repozitáře systému APT v Raspberry Pi OS 10 (autor, 2021)

existují i předkompilované balíčky vytvořené komunitou, například verze Qt5.15.2 LTS, kterou předkompiloval Koen De Vleeschauwer a zveřejnil na svém GitHubu i s postupem kompilace. [7]

6.4.2 Instalace qthttpserver

QtHttpServer není součástí standardní instalace Qt frameworku a proto je nutné jej doinstalovat zvlášť. Nejjednodušší způsob je využití nástroje git

1. Instalace git
2. spuštění příkazu

   ```
   git clone https://github.com/qt-labs/qthttpserver
   ```

   Spuštění Qt konzole
3. příkaz v konzoli:

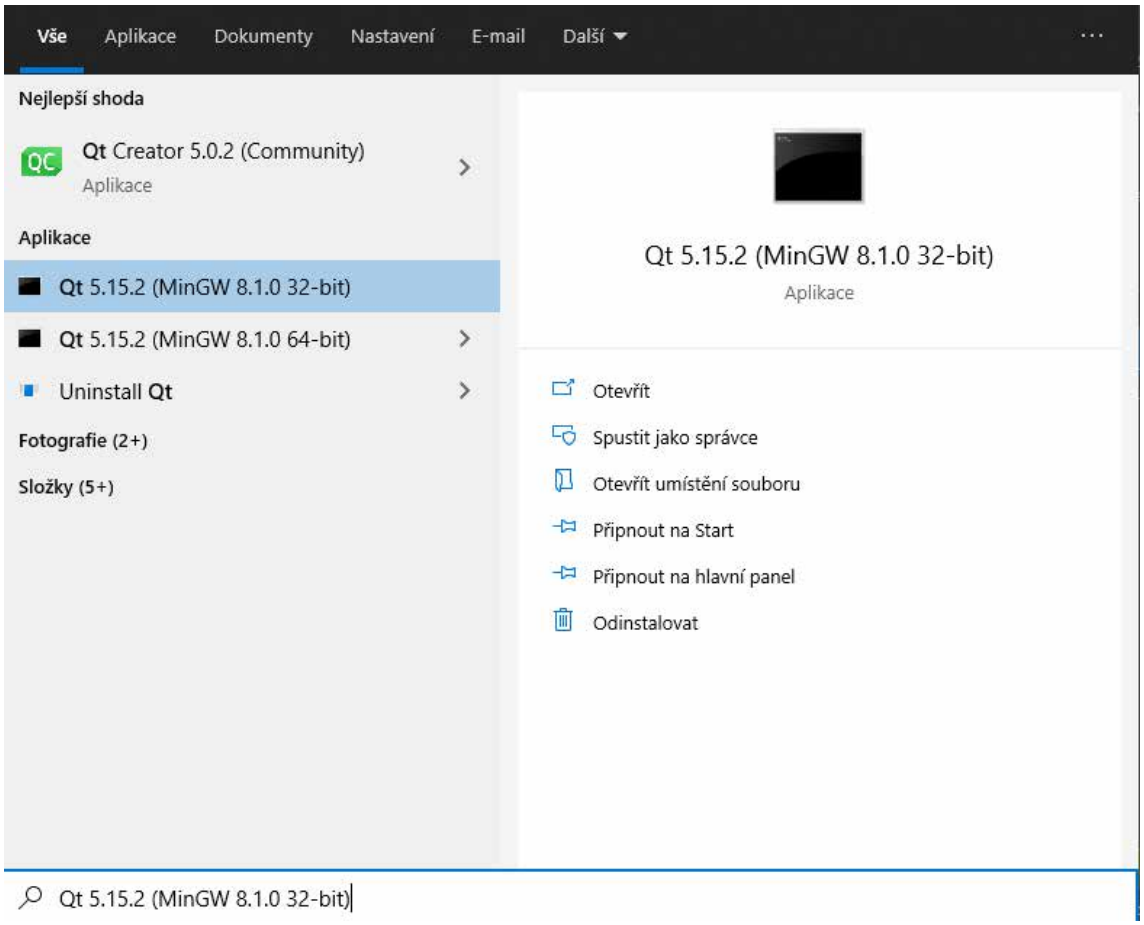
   ```
   qthttpserver
   ```

   cd
4. příkaz v konzoli:

   ```
   qmake
   ```
5. příkaz v konzoli:

   ```
   mingw32-make
   ```
6. příkaz v konzoli:

   ```
   mingw32-make  install
   ```



Obrázek 17: Postup otvírání Qt konzole (autor, 2021)

6.4.3 Windeployqt

Aby bylo možné spustit zkompilované programy, je nutné spustit v Qt konzoli příkaz

```
windeployqt Vdv301.exe
```

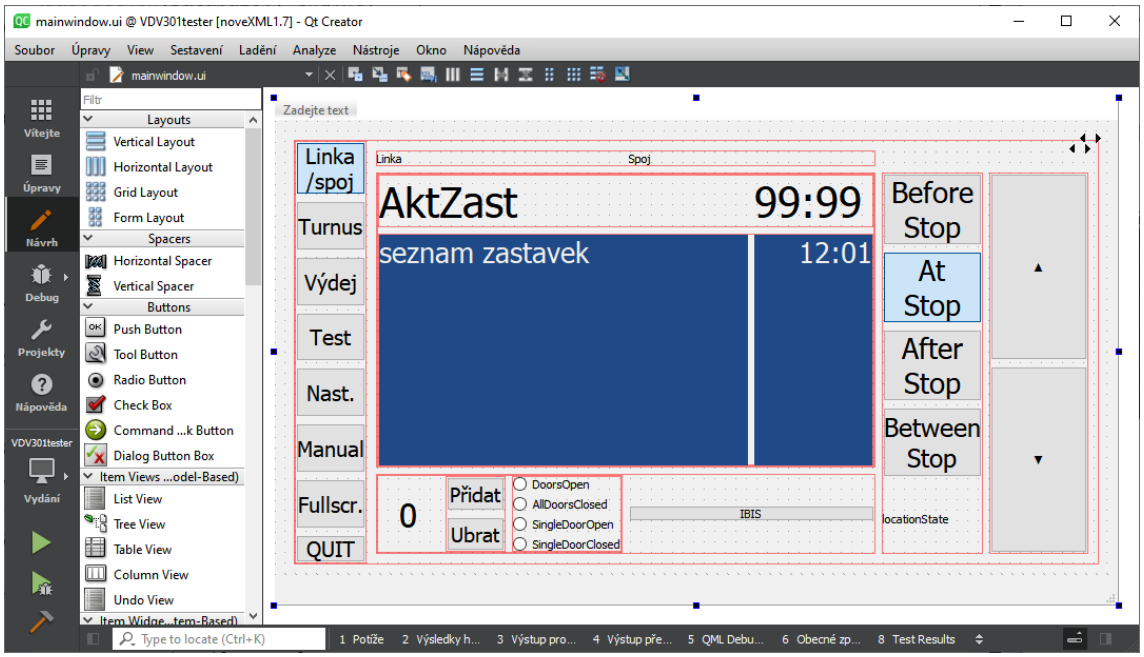
, který do složky k .exe nakopíruje potřebné soubory Qt knihoven.

Poté není nutné ani instalovat celé Qt SDK do cílového počítače. Výsledný program veškeré soubory uchovává ve své složce a lze jej spouštět bez instalace, například z USB úložiště.

6.5 Signály a sloty

Specifikem celého Qt frameworku je práce se signály a sloty. Toto umožňuje zpracovávat několik různých vláken kódu naráz a spolupracovat knihovnám mezi sebou.

To funguje tak, že *třída A* disponuje slotem a *třída B* vyvolává signál v při určitých situacích. I přesto, že jsou instance obou tříd definované na stejné úrovni kódu v hlavní třídě *MainWindow*, může událost v *(třídě B)* vyvolat spuštění kódu ve *třídě A*.



Obrázek 18: Nástroj pro navrhování grafického rozhraní (autor, 2021)

7 Realizace VDV301tester

7.1 Funkce

Pro komplexní testování byla zvolena tvorba vlastního programu místo složitého skriptování hotového řešení. Hotová řešení pro testování HTTP API navíc neumožňují automatickou tvorbu těla HTTP požadavku dle zvolených vstupních dat.

Vznikl tedy program VDV301tester.exe, který ve spojitosti s pomocným programem VDV301display.exe tvoří testovací prostředí.

Výsledný program *VDV301tester.exe* splňuje následující funkcionality:

- Import jízdního řádu z XML souboru do SQLite databáze
- Výběr konkrétního spoje
- Simulace jízdy - rozhraní podobné palubnímu počítači
- zobrazení časů odjezdu a zastávek na znamení
- ovládání zobrazovacích panelů pomocí IBIS (VDV300)
- ovládání zobrazovacích panelů pomocí IBIS-IP (VDV301)
  - publikování služby
  - generování XML dat do služeb
  - odesílání dat odběratelům
  - manuální zastavování/spouštění služeb

  - kontrola obsahu generovaných XML dat přes HTTP Get
- hlášení zastávek z dodaných MP3 souborů
- spouštění automatických testů

7.2 Implementované funkce

8 Uživatelské rozhraní

Uživatelské rozhraní je přizpůsobeno na plochu 800×480 px, což je rozlišení dotykového displeje 7"displeje *Raspberry Pi Touch Display*, který byl využit při vývoji a jako demonstrátor Rozhraní je uzpůsobeno tak, aby připomínalo skutečný palubní počítač.

Obrázek 19: program VDV301tester běžící na Raspberry Pi 3 s dotykovým diplejem (Autor, 2021)

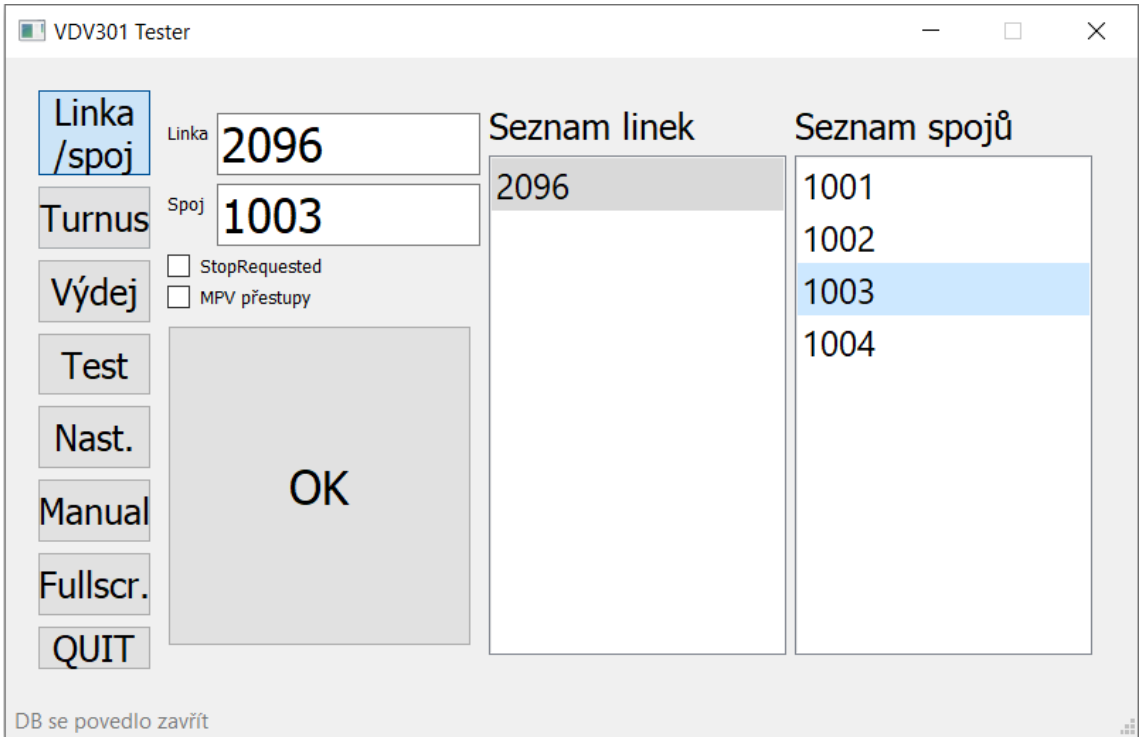
Linka/spoj Pro téměř všechny funkce programu je třeba zvolit linkospoj z dat. Výběr je prováděn výběrem v seznamu, což je varianta vhodná pro ovládání prsty, na rozdíl od varianty zapisování čísel přímo do políček, což navíc vyžaduje klávesnici.

Další výhodou je okamžitý přehled, jaké linky a spoje jsou k dispozici. Z výběru jsou odfiltrovány neveřejné spoje a přejezdy.

Zatržítko MPV přestupy umožnuje zvolit, zda se automaticky v průběhu simulované jízdy budou ze systému MPVnet stahovat přestupy.

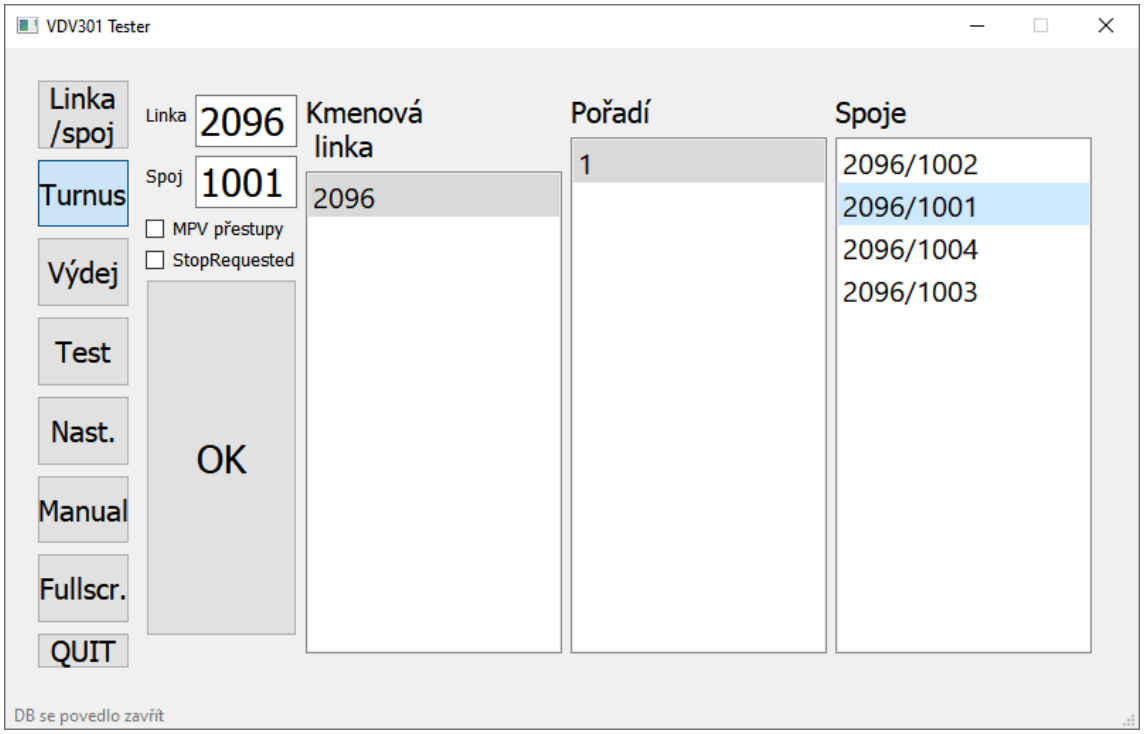
Turnus Alternativní možností je výběr spoje dle kmenové linky a čísla pořadí (tzn. oběhu/turnusu). Zatím není implementováno automatické přepínání po dojetí linkospoje.

Obrázek 20: Základní obrazovka skutečného zařízení Mikroelektronika OCC3 (Autor, 2021)



Obrázek 21: Okno s výběrem linkospoje (autor, 2021)

Toto ovšem bude muset být v budoucnu implementováno kvůli testování návazných spojů.



Obrázek 22: Okno s výběrem spoje dle oběhu (autor, 2021)

Výdej Nejdůležitějším oknem z celého programu je okno výdeje. Zobrazí se po stisknutí tlačítka OK při výběru existujícího spoje na předchozích obrazovkách. Obrazovce dominuje zobrazovač následujících zastávek s časy odjezdů z dané zastávky. Tento zobrazovač se dynamicky mění spolu s posouváním zastávky. Nejbližší zastávka je ta uvedena v bílém řádku, který je obsazen pouze při stavu *v zastávce*:*AtStop*. Simulovat jízdu lze pomocí šipek nahoru a dolu. Tyto způsobují postupné přesouvání mezi stavy. Tyto přesuny mezi stavy vyvolávají různé události, například příjezd do zastávky (*BeforeStop*>*AtStop*) aktivuje vyhlášení zastávky.

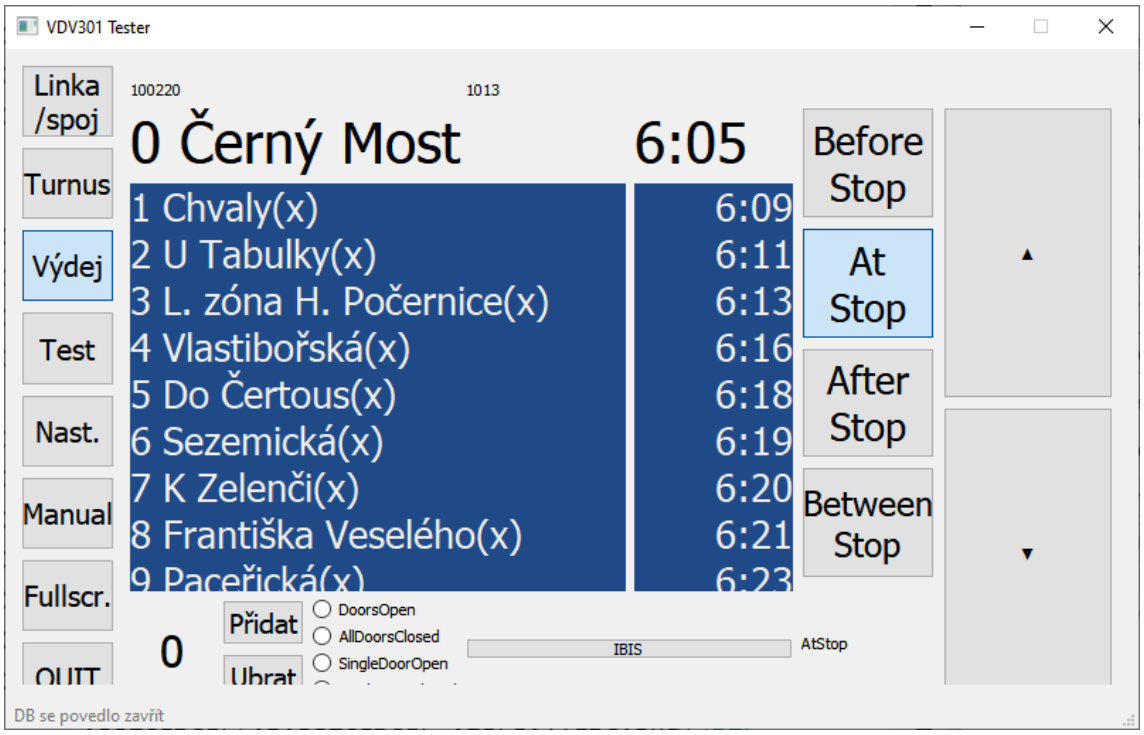
Stavy polohy lze měnit přímo i tlačítky, jejichž názvy odpovídají těmto stavům, avšak v případě takové volby nedochází k posunu mezi zastávkami.

Naopak pro skákání mezi zastávkami bez změny polohového stavu lze využít tlačítka **Přidat** a **Ubrat**.

Další podstatné tlačítko je **IBIS**. Toto tlačítko spustí odeslání aktualizovaných údaju o cestě do sériového portu pro ovládání starších zařízení fungujících dle normy VDV300 po protokolu IBIS.

Aktualizace těchto dat probíhá manuálním stiskem tlačítka, neboť při odesílání dat dochází k výraznému zpomalení programu. Nepodařilo se zjistit, zda je chyba na straně knihovny QtSerial, použitého čipu USB-RS232 převodníku, či špatné implementace knihovny v kódu.

Naopak aktualizace a odeslání dat do panelů fungujících přes VDV301 (IBIS-IP) probíhá automaticky při změně polohového stavu/indexu zastávky a žádné zpomalení programu samotného nezpůsobuje.



Obrázek 23: Okno výdeje programu VDV301tester (autor, 2021)

Nast. vede do nastavení. Zde je možné vybrat sériový port pro ovládání starších IBIS periferií a složku s hlášením.

Nejdůležitější částí je ovšem sekce vstupní data, kde probíhá práce s databází a výběr vstupního souboru následujícím postupem:

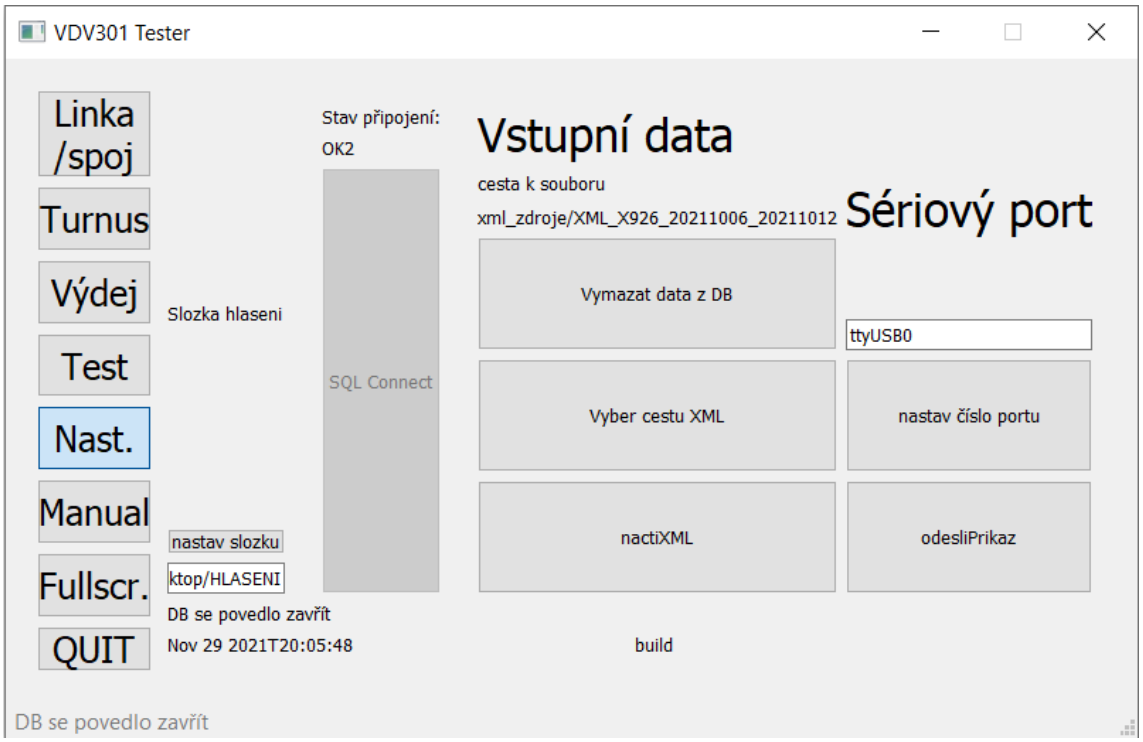
1. **Vymazat data z DB** vyprázdní data ze všech tabulek SQlite databáze a přitom zachová databázovou strukturu.
2. **Vyber cestu XML** otevře okno pro výběr složky s XML souborem.
3. **nactiXML** spustí import XML souboru do databáze.

Průběh procesu je indikován zobrazením hlášky *DB se povedlo zavřít.*

Tento postup je nutné provádět pouze při změně vstupních dat. Údaje v databázi zůstanou i po restartu programu.

Naopak složku s hlášením je nutné nastavovat po každém restartu. Implementace konfiguračního souboru, který by toto řešil, není dokončena.

Manual slouží pro správu odběratelů a odesílání předvoleného XML na panely.



Obrázek 24: Okno nastavení (autor, 2021)

Do seznamu odběratelů se automaticky přidávají všichni odběratelé, a to ihned po odběru. Odběratele lze do seznamu manuálně přidat i odebrat.

Možnost manuálního odeslání dat slouží pro případy, kdy je provedena změna v protokolu, která ještě nebyla zapracována do zdrojového kódu programu VDV301tester, ale periferie tuto změnu už podporují.

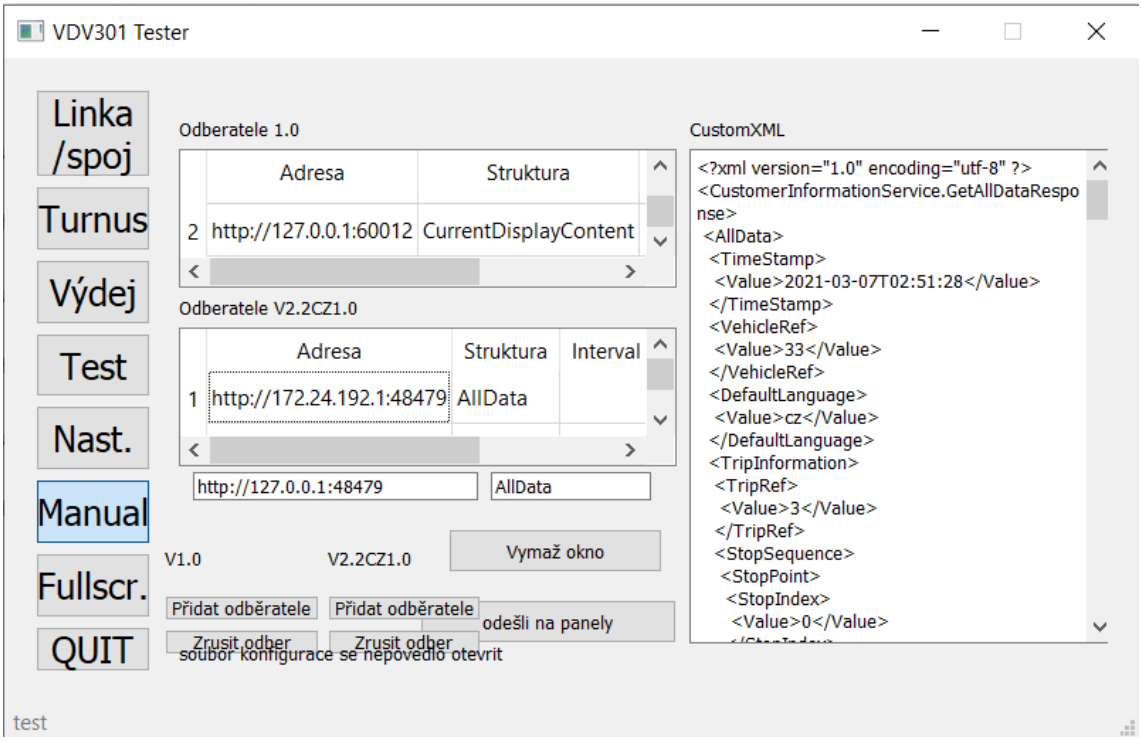
Druhá možnost využití je při výuce, kdy studenti mají možnost zkoušet manuální zásahy do připraveného XML souboru a sledovat změny na připojeném LCD.

Test přepne celé rozhraní do testovacího režimu, vrátit zpět lze tlačítkem **PalPC**. Na záložce **Služby** je možné manuálně vypínat a zapínat služby jako takové. Tlačítka **Fullscr.** a **QUIT** mají stejné funkce jako v režimu palubního počítače, a to režim celé obrazovky a ukončení programu.

Průběh testu obsahuje okno pro automatické testy. V první řadě je nutné vybrat test ke spuštění z několika tlačítek nad seznamem fází. Tím dojde k naplnění seznamu fází testu. Poté je třeba test zahájit tlačítkem **Spustit test**. Testy se ukončují automaticky. V případě potřeby test ukončit předčasně je k dispozici tlačítko Zastavit test.

První je **Vzorový test**. Ten slouží pouze pro otestování interního časovače programu a pouze simuluje průběh testu - fáze se posouvají automaticky po uplynutí časového limitu. Tlačítka **Test Bonjour** a **XX** jsou připravena pro budoucí rozvoj.

Jediným skutečně implementovaných automatickým testem je **Test Odberu**, který provádí test na připojené periferii. Aktuálně je implementován pouze pro verzi 2.2CZ1.0.



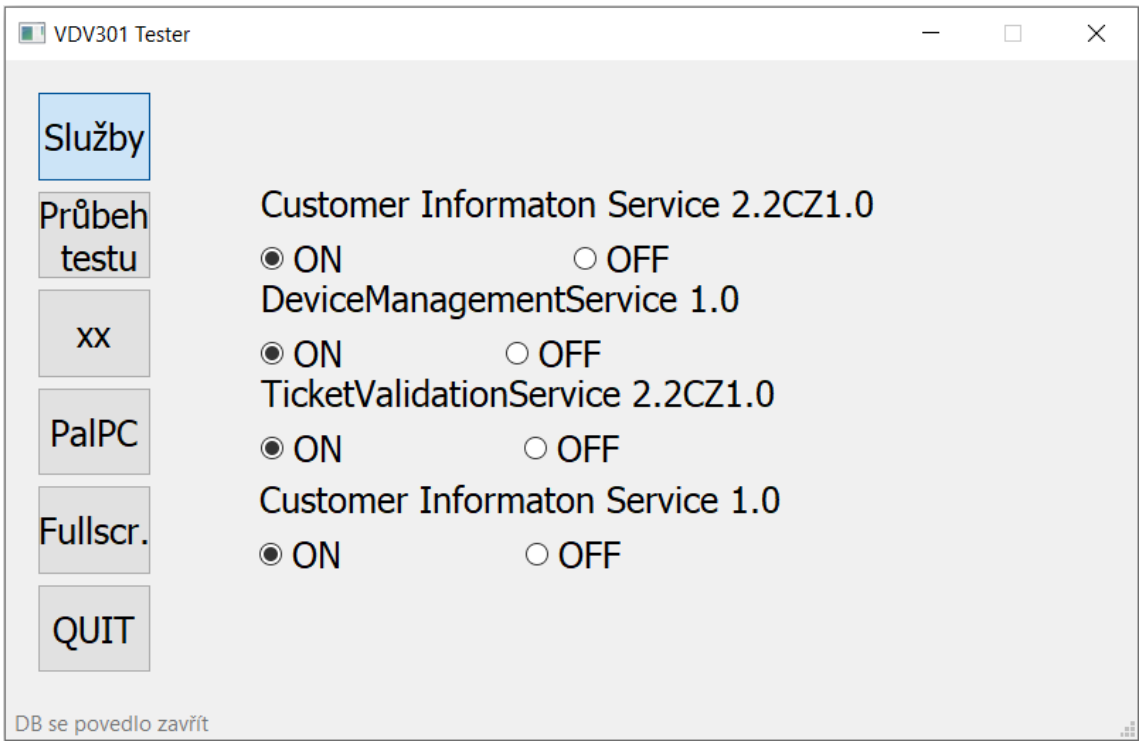
Obrázek 25: Okno manuálního odesílání dat a správy odběrů (autor, 2021)

Má několik fází:

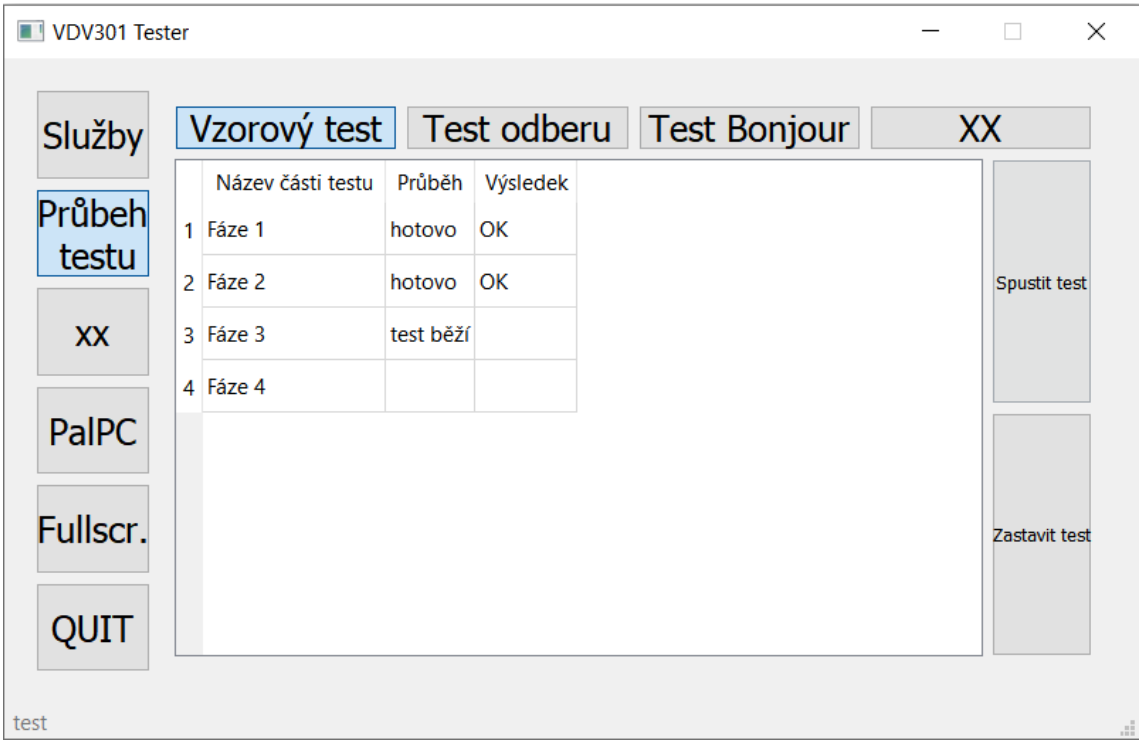
1. **První odběr po spuštění**: před začátkem testu je třeba odpojit testované zařízení z počítačové sítě. Poté je možné spustit test. Program očekává, že se zařízení přihlásí k odběru. Pokud ano, vypíše IP zaříení, které se přihlásilo.
2. **Odezva na odeslaná data**: Program zastaví časovač služby *Customer informationService* a odešle data do panelu. Zkoumá, zda z panelu přijde odpověď na HTTP Post požadavek. Očekává *200 OK*.
3. **Odběr znova po 120 s bez dat**: V této fázi program čeká, zda se po době 120 s bez odeslání dat testované zařízení opět přihlásí k odběru. Timeout testu je s rezervou nastaven na 200 s.
4. **Response**: Probíhá stejně jako fáze 2.

Každý test má nastavený vlastní časový limit, do kterého musí fáze vrátit úspěšný výsledek.

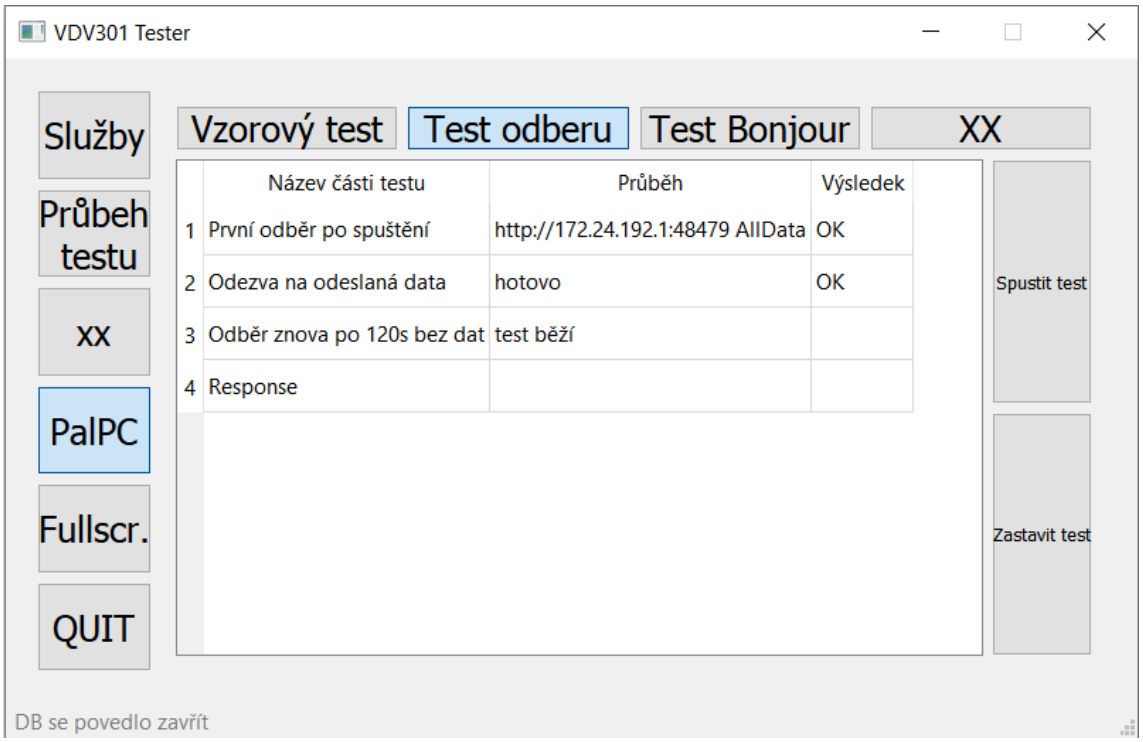
Test indikuje svůj průběh tak, aby bylo jasné, ve které fázi se právě nachází. Obdobně program indikuje selhání v testu. Vzhledem k návaznosti testů se při prvním selhání některé z fází test ukončí.



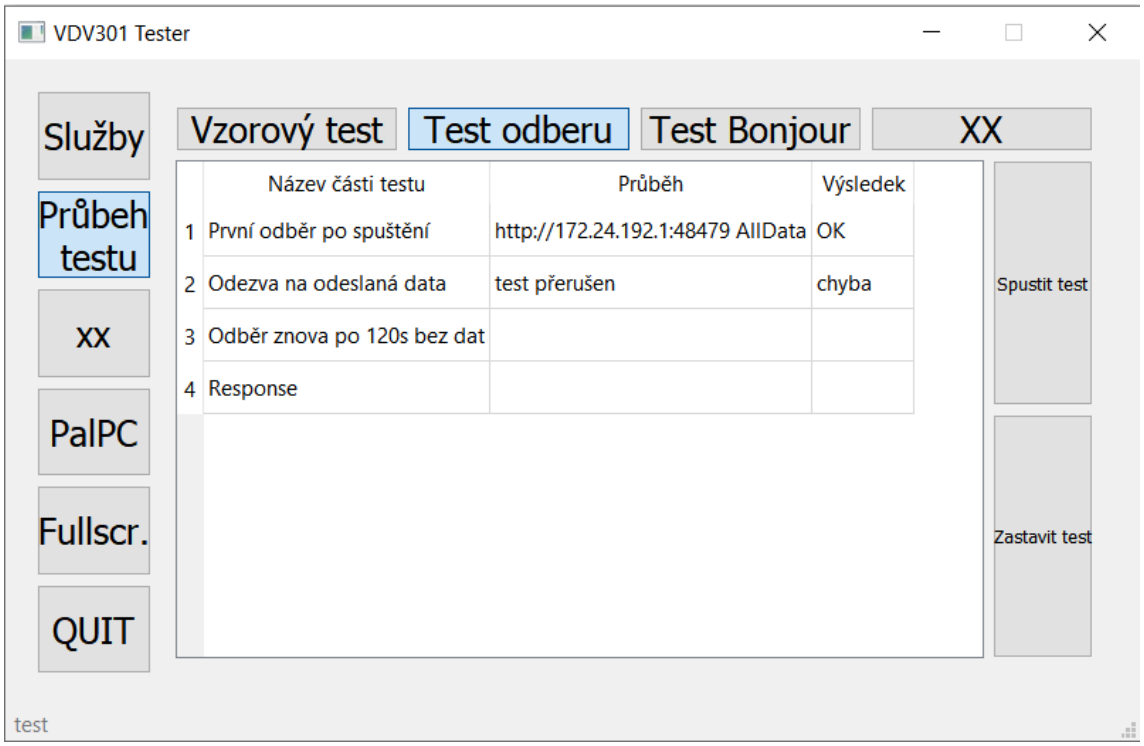
Obrázek 26: Možnost manuální deaktivace služeb (autor, 2021)



Obrázek 27: Vzorový test - simulace (autor, 2021)



Obrázek 28: Průběh testu (autor, 2021)



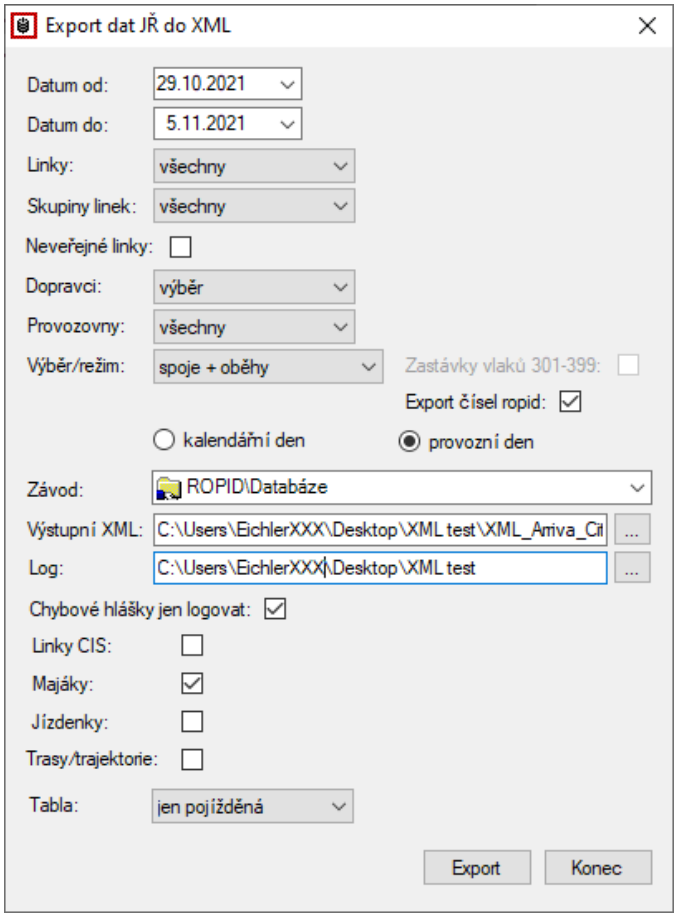
Obrázek 29: Indikace selhání testu (autor, 2021)

8.1 Datový vstup

8.1.1 ropidXML

Program je upraven pro příjem vstupních dat ve formátu XML vytvořených programem *ASW JŘ* firmy CHAPS.

Jedná se o mnohem pokročilejší alternativu formátu JDF. XML soubor obsahuje jak veřejně dostupná data, jako jízdní řády, tak i oběhy, které jsou interním tajemstvím dopravce. Proto jsou přiložená data záměrně neaktuální a ořezaná.



Obrázek 30: Okno exportu XML dat programu ASW JŘ (autor, 2021)

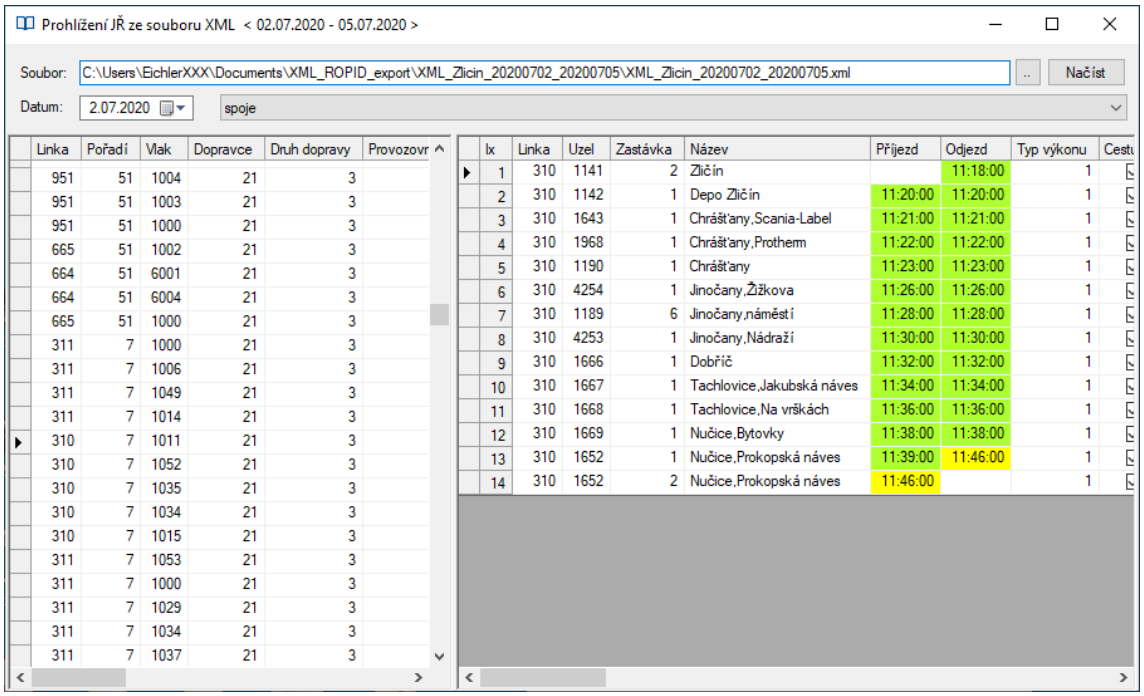
Vygenerovaná data lze ihned zkontrolovat pomocí vestavěného programu Prohlížeč XML.

Z funkcionalit obsažených ve vstupních datech byly implemetovány tyto:

- různé názvy zastávek podle typu zobrazovacího panelu
- čas odjezdu dle jízdního řádu
- návazné spoje (částečná implementace)
- výběr spoje dle turnusu
- výběr spoje dle čísla linky a spoje
- náležitost zastávek do tarifních bodů

Naopak tyto funkcionality byly vynechány:

- kalendář jízd



Obrázek 31: Kontrola dat v modulu Prohlížeč XML programu ASW JŘ(autor, 2021)

- majáky preference
- změna tarifního systému

8.2 Databázová struktura

Databázová struktura byla vytvořena tak, aby respektovala strukturu vstupních XML dat. Nebylo tedy možné využít databázovou strukturu z bakalářské práce[1], která nerespektovala rozdělení zastávek na uzly a stanoviště. Vzhledem k podobnosti databázové struktury a vstupního souboru, jsou příklady uvedené ze zdrojových XML dat. Rozdíly struktur XML a DB jsou uvedeny.

Specifickým prvkem téměř každého elementu je *kalendář jízd*. Jedná se o binární masku platnosti daných elementů a obsahuje tolik prvků, pro kolik dní jsou XML data vygenerována.

Například může existovat více variant stejného spoje lišících se časy odjezdu, pokud každý den z intervalu platnosti celé dávky platí pouze jedna varianta (číslo 1 na n-té pozici v atributu *kj*, kde n je pořadí dne v intervalu platnosti)

```
1 kj="111000010"
```

Kalendář jízd v programu není integrován a ze všech variant jsou zobrazovány ty, které platí první den z intervalu platnosti.

8.3 Jednotlivé tabulky

Názvy jednotlivých tabulek jsou zvoleny shodně s názvy elementů v XML ROPID formátu z ASW JŘ pro větší přehlednost při progamování. Samotné elementy XML mají krátké názvy pro aspoň částečné snížení datové náročnosti formátu.

Ve výsledná databáze databáze obsahuje drtivou většinu dat obsažených v XML souboru. Obsah XML souboru byl importován v největší možné míře pro zjednodušení postupné implementace tak, aby nemusela být prováděna kontrola provedeného importu. .

Popis tabulek je seřazen dle logické návaznosti při implementaci.

d Dopravci Seznam dopravců, pro které je dávka vygenerována. ASW JŘ umožňuje vygenerovat dávku pro pro více dopravců naráz.

Naplněno, neimplementováno.

```
<d c="79" n="Lutan␣s.r.o." kj="1111111" ncis="Lutan␣s.r.o." ico="
   06292755" dic="CZ06292755" ul="Benátecká␣Vrutice␣16" me="Milovice
   " psc="289␣23" tel="+420␣778␣448␣210" teld="+420␣778␣448␣210"
   teli="+420␣778␣448␣210" em="lutan@email.cz" />
```

p (Provozovny) Seznam provozoven je využíván později při konstrukci oběhů. Vozidlo může patřit jen do jedné provozovny, tím pádem celý oběh musí být v režii jen jedné provozovny.

ids (Integrovaný Dopravní Systém) Tato tabulka definuje všechny IDS, do kterých linka spadá. Využíváno u mezikrajských linek, např. PID-IDOL, PID-IREDO. Využíváno u zpracování tarifních pásem.

dd (Druh Dopravy) Jedná se o číselník dopravních prostředků. Např. autobus, tramvaj, loď.

tv (Typy Vozů) Tento číselník se zaměřuje na silniční vozidla a rozděluje je z hlediska kapacity a nízkopodlažnosti. Každý oběh má přiřazený typ vozidla, kterým má být obsluhován. Např. Kb - kloubový, Sd - standardní, MdN - nízkopodlažní midibus

```
<tv c="31" z="Sd" n="Standard" dd="3" />
<tv c="32" z="Kb" n="Kloubový" dd="3" />
<tv c="33" z="SdN" n="Standard␣(NP)" dd="3" np="true" />
<tv c="34" z="KbN" n="Kloubový␣(NP)" dd="3" np="true" />
<tv c="39" z="Sd@N" n="Standard␣plus␣(NP)" dd="3" np="true" pl="true"
   />
<tv c="42" z="MdN" n="Midibus␣(NP)" dd="3" np="true" pl="true" />
<tv c="45" z="Md" n="Midibus" dd="3" />
<tv c="49" z="Md@" n="Midibus␣plus" dd="3" />
<tv c="50" z="Md@N" n="Midibus␣plus␣(NP)" dd="3" np="true" />
```

k (Kategorie linek) Číselník kategorie linek pro IDOS a MPVnet. Může být využito pro zabarvení čísla linky na LCD pro cestující.

Příklady:

```
<k c="3" n="autobus" />
<k c="4" n="autobus␣regionální" />
<k c="9" n="školní␣linka" />
<k c="11" n="smluvní" />
<k c="16" n="autobus␣regionální␣noční" />
```

Tento číselník není implementován, budoucí implementace pro simulátor bude nutná pro správnou volbu dopravního prostředku pro generování dat.

z (Zastávky) Seznam zastávek reprezentuje jednotlivé označníky. Každý označník je reprezentován číslem uzlu a pořadovým číslem daného stanoviště. Kombinace těchto dvou čísel se využívá jako složený cizí klíč v ostatních tabulkách.

Dále obsahuje např. informace o tarifním pásmu a GPS souřadnice daného sloupku.

V této tabulce jsou uvedeny i obecné příznaky zastávek. Nicméně tyto příznaky nerespektují časová omezení, a proto program využívá přestupy z elementu x (zastavení).

Tento element obsahuje i CIS číslo, které je využito pro stahování informací o přestupu.

```
<z u="897" z="1" kj="11111111" n="Černý␣Most" n2="Černý␣Most" n3="Černý␣Most" n4="Černý␣Most" n5="Černý␣Most" n6="Černý␣Most" n7="Černý␣Most" n8="Černý␣Most" pop="výstup,Černý␣Most␣(směr␣Horní␣Počernice)" cis="47090" ois="897" co="554782" no="Praha" cco="9" nco="Praha␣9" spz="AB" ids="1" tp="P,0" sx="-731434.38" sy="-1042135.83" lat="50.1091156" lng="14.5771322" sta="V1" m="1" bbn="true" xB="true" kidos="301003" st="CZ" />
```

t (Tabla) Položka tabla všechny varianty názvu sloupku. Například název pro tisk na JŘ, pro zobrazení na palubním PC, pro zobrazení na čelním panelu.

```
<t u="897" z="1" kj="11111111" ois="897" cis="47090" nza="Černý␣Most" ri="Černý␣Most" ji="Černý␣Most" vtm="Černý␣Most␣=M=" vtn="Černý␣Most␣=M=" btm="Černý␣Most␣[B]" btn="Černý␣Most␣[B]" ctm="ČERNÝ␣MOST␣&lt;" ctn="`ČERNÝ␣MOST␣&lt;" lcdm="Černý␣Most" lcdn="Černý␣Most" hl="přestup␣na␣metro␣B" n="Černý␣Most" nf="Černý␣Most" />
```

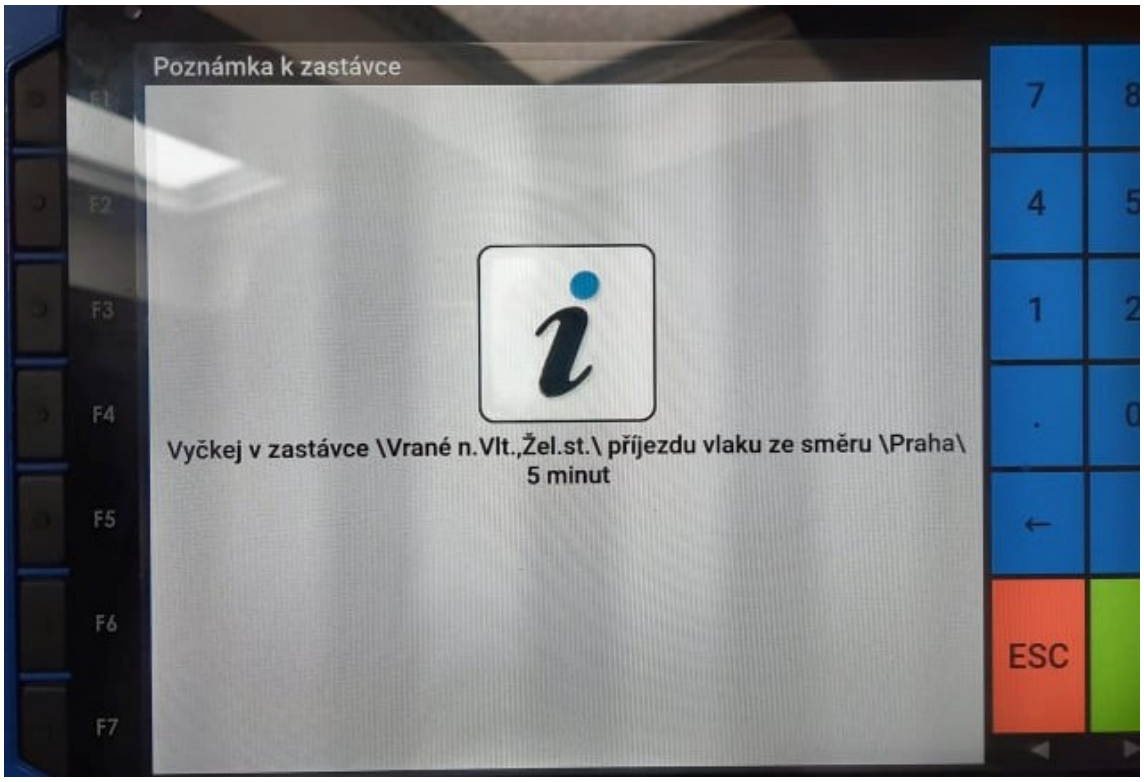
l (Linky) Seznam linek obsahuje parametry linky (denní, noční), názvy linky (celý název, licenční číslo, alias pro OIS).

Dle zvyklostí se pro informační systém využívá alias a pokud není vyplněný, využívá se poslední trojčíslí licenčního čísla.

```
<l c="204" d="18" kj="11111111" lc="100204" tl="A" n="Stavební␣zóna␣Horní␣Počernice␣-␣Sídliště␣Petrovice" kup="1" ids="true" kli="3" cids="1" />
```

po (POznámky) Seznam poznámek obsahuje informace o návaznostech, které se v průběhu jízdy zobrazují řidiči.

Vytvořeno, nenaplněno. Bude implementováno v budoucnosti.



Obrázek 32: Zobrazení poznámky na reálném zařízení Mikroelektronika OCC3 (autor, 2021)

```
<po c="14" t="V␣zast.␣Dolní␣Počernice␣vyčkej␣opožděného␣příjezdu␣linky
   ␣109␣z␣centra␣max.␣2␣min." zkr1="/B/" zkr2="/B/" zkr3="/B/" ois="
   true" n="true" tn="m" u="97" z="2" u2="97" z2="1" nl="109" anl="
   109" cd="120" usm="801" zsm="1" dd="3" />
```

s (Spoje) Seznam spojů obsahuje příslušnost k lince a obsahuje vlastnosti spoje. Mezi spoje jsou započítány i manipulační přejezdy.

Spoje jsou identifikovány unikátním pořadovým číslem v dávce, které tvoří primární klíč. Pro zobrazení se ovšem využívá čtyřmístné číslo, které je unikátní pouze v rámci jedné linky.

Běžný postup z praxe byl při generování jízdních řádů očíslovat spoje podle času odjezdu. Po odstranění spoje došlo k přečíslování všech spojů časově následujících. To vytvořilo problémy v návazných systémech (např. MPVnet) a vnitřních procesech dopravce.

U systému čtyřmístného čísla dojde k přegenerování pouze těch spojů, které nově vznikly.

x (zastavení) Zastavení tvoří pomyslný jeden čas odjezdu v jízdním řádu. Tento element spojuje spoj, sloupek a čas. Zároveň obsahuje příznaky zastávky. Například odpolední spoj hlásí přestup na metro, noční ne.

Při importu do databáze dochází k lehké změně a to k takové, že v XML souboru jsou

zastavení **x** uvedena jako vnořené prvky elementu spoje **b**, což plně odpovídá filozofii formátu XML.

```
<s s="2" id="296148" zvd="50" l="2096" p="1" sm="false" dd="3" pr="
    108" d="79" tv="41" kj="1110011" ty="1" ch="1" ids="true" vy="
    true" sp1="true" c="1002">
    <x u="2977" z="1" o="26460" ty="1" zn="true" xVla="true" icls="5">
      <v />
    </x>
    <x u="2976" z="1" p="26760" o="26760" ty="1" xVla="true" icls="4">
      <v m="6300" />
      <v p="5" m="6300" />
    </x>
    <x u="9166" z="2" p="27120" o="27120" ty="1" icls="3">
      <v m="4740" />
      <v p="5" m="4740" />
    </x>
    <x u="9167" z="1" p="27240" ty="1" icls="2">
      <v m="510" />
      <v p="4" m="510" />
    </x>
 </s>
```

Pro převod do databázové struktury, kde není možné vytvářet vnořené elementy, jsou zastavení **x** uvedena ve zvláštní tabulce a se spojem jsou propojena cizím klíčem **s_id**.

Za lehce nešťastné lze považovat uvedení příznaků jako atributů XML. Vzhledem z možnosti vzniku potřeby dalších příznaků by bylo vhodnější tyto příznaky uvádět jako vnořené elementy a nebylo by nutné uvádět u každé zastávky všechny atributy, když je hodnota většiny z nich nastavena na**false**.

Jak je uvedeno výše, kromě tabulky zastávek jsou příznaky uvedeny i zde, ovšem nemusí je jednat o kopii 1 : 1. V elementu x se příznaky vztahují ke konkrétnímu zastavení. Například pro školní linky neplatí zastávky na znamení, pro noční linky není hlášen přestup na metro.

o (Oběhy) Oběhy obsahují hlavičku oběhu a seznam spojů, které do daného spoje spadají. Příslušnost je zde znázorněna nekonvenčně a jinak než u zastavení.

Vzhledem k naprosto nevhodné reprezentaci navazujících spojů v XML je jejich reprezentace vyřešena jinak.

V XML struktuře je seznam spojů uveden v atributu prvku **o** jako vektor čísel. Toto odporuje veškerým principům formátu xml. Dále jsou navazující spoje uvedeny ve zvláštních elementech **ds**. Zde se projevuje duplicitní uložení informací, neboť je každý dlouhý spoj podmnožinou oběhu a pořadí následujících spojů je neměnné.

```
<o l="402" p="62" kj="0000001" sp="564␣565␣483␣484␣485␣997␣998␣999" tv
    ="49" td="3">
        <ds sp="565␣483␣484" />
        <ds sp="485␣997" />
        <ds sp="998␣999" />
</o>
```

Řešením pro uložení spojů u oběhů je nová tabulka **sp_po**. Ta reprezentuje seznam seřazených spojů. U spojů, které jsou uvedeny v elementu **ds** (kromě posledního v

každém dlouhém spoji) je uveden příznak **pokrac**. Tím je zachována informace o tom, jak se palubní počítač má zachovat při přechodu mezi spoji.

Název sloupce	Popis sloupce
l	linka
p	pořadí
kj	kalendář jízd
s	id spoje
ord	pořadí na oběhu
pokrac	příznak navazujícího spoje

Tabulka 1: Struktura tabulky **sp_po**

m (Majitelé označníků) Seznam majitelů označníků. Naplněno, neimplementováno.

r (pReference) Číselník úrovní preferencí Vytvořeno, nenaplněno.

sp_po (SPoj - POřadí) . Tato tabulka jako jediná není přímo založena na elementu z XML, ale vznikla na základě nutnosti zachovat informaci o přiřazení spoje do pořadí. Zároveň integruje informaci o tom, zda spoj následující za aktuálním spojem je navazující (dlouhý spoj - ds). V takovém případě dochází ke speciálnímu chování:

- Nevyhlášení průjezdné konečné
- Zobrazení "dále jako"na panelech
- Zobrazení zastávek následujícího spoje na LCD ihned za zastávkami současného spoje

ty (TYp výkonu) Číselník typů výkonu spoje. Nenaplněno.

ch (CHarakteru výkonu) Číselník charakterů výkonu. Vytvořeno, nenaplněno.

8.4 Databázový engine

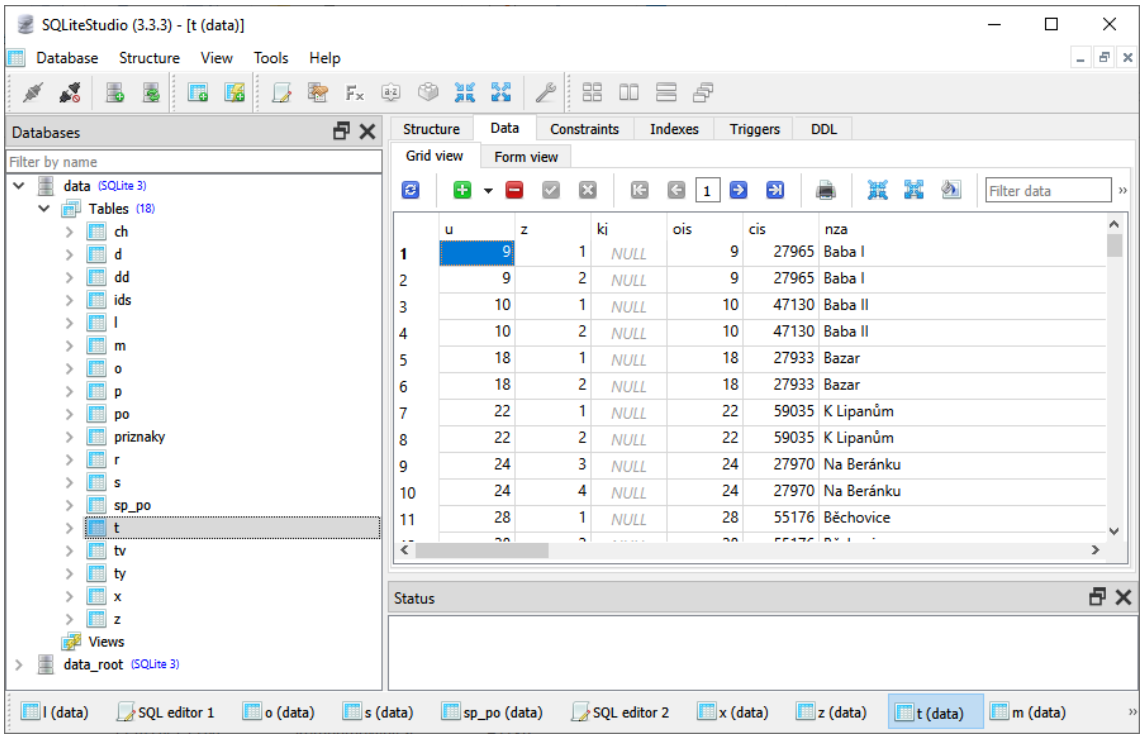
Pro ukládání dat přímo pro potřeby programu je využit databázový engine SQLite, který místo databázového serveru využívá jeden soubor se samotnou databází. Toto vede ke zjednodušení instalace programu na nová zařízení.

Při instalaci s klasickým databázovým serverem (v BP[1] byl využit MySQL) je nutné tento databázový server zvlášť nainstalovat (MySQL, MariaDB) zároveň s prostředím pro správu (PHPMyAdmin). toto prostředí pro svou funkci vyžaduje navíc webový server (Apache) a interpretem jazyka PHP. Všechny tyto prvky lze nainstalovat naráz

pomocí předpřipravených balíků (např. WAMP pro Windows). Poté je nutné naimportovat zálohu databázové struktury.

Při použití SQLite přístupu stačí distribuovat pouze databázový soubor SQLite a o práci s ním se stará přímo VDV301tester.

Pro manuální zásahy je možné využít freeware programy *DB Browser for SQLite* nebo *SQLiteStudio*. Konkrétně druhý jmenovaný byl použit pro vytvoření databázové struktury.



Obrázek 33: Prohlížení vytvořené databáze v programu SQLiteStudio (autor, 2021)

Další výhodou SQLite je dostupnost knihovny/ovladače přímo v použitém vývojovém balíku Qt. Ovladače (dle Qt terminologie SQL Database Drivers) pro přístup do MySQL databází je třeba zvlášť stáhnout, zkompilovat a nakonfigurovat. [6]

8.5 VDV301tester knihovny

8.5.1 Publisher

newhttpserver.cpp Tato třída zakládá na knihovně *qthttpserver* a zpracovává veškeré HTTP požadavky, která do programu dorazí.

httpsluzba.cpp Tvoří mateřskou třídu a obsahuje tedy všechny sdílené funkce pro IBIS-IP služby. Spravuje mechanismy odběru (přidávání/odstranění odběratele) a obsluhuje pravidelné obvolávání odběratelů po vypršení časovače.

customerinformationservice.h Dědí od *httpsluzba.cpp* a obsahuje funkce specifické pro tuto konkrétní službu. Tato třída dokáže zároveň obsluhovat zařízení vyžadující protokol IBIP-IP verze 1.0 a 2.2CZ1.0.

ticketvalidationservice.cpp Tato třída obsahuje funkce specifické pro TicketValidationService. Jedná se hlavně o naplnění správnými daty pro odeslání do zařízení tuto službu vyžadující (hlavně označovače cestovních dokladů, validátory).

8.5.2 VDV301testy

vdv301testy.cpp Mateřská třída obsahující části kódu shodné pro všechny testy. Obsahuje například výstup pro vykreslení do tabulky fází a spravuje stavy probíhajícího testu.

polozkatestu.cpp Datová struktura obsahující informace o aktuálně probíhající fázi testu.

testdemo.cpp Vzorový test, ověřuje funkčnost třídy Vdv301testy

testodberuserver.cpp Ostrý test odběratele služby CustomerInformationService. Používá signály pro propojení s instancí třídy CustomerInformationService.h jménem customerInformationService2.2CZ1.0. Je napsána tak univerzálně, že pro použití s jinou službou stačí přepsat ve třídě MainWindow propojení slotů a signálů na jinou službu:

```
1     connect(&customerInformationService2_2CZ1_0,&
          CustomerInformationService::signalVypisSubscriberu,&
          testOdberuServer,&TestOdberuServer::slotAktualizaceSubscriberu
          );
2     connect(&testOdberuServer,&TestOdberuServer::
          signalVymazSeznamOdberatelu,&
          customerInformationService2_2CZ1_0,&CustomerInformationService
          ::slotVymazSubscribery);
3     connect(&testOdberuServer,&TestOdberuServer::signalNastartujSluzbu
          ,&customerInformationService2_2CZ1_0,&
          CustomerInformationService::slotStart);
4     connect(&testOdberuServer,&TestOdberuServer::signalZastavCisTimer
          ,&customerInformationService2_2CZ1_0,&
          CustomerInformationService::slotZastavCasovac);
5     connect(&testOdberuServer,&TestOdberuServer::
          signalOdesliDataDoPanelu,&customerInformationService2_2CZ1_0,&
          CustomerInformationService::slotTedOdesliNaPanely);
6     connect(&customerInformationService2_2CZ1_0,&
          CustomerInformationService::signalVypisSubscriberu,&
          testOdberuServer,&TestOdberuServer::slotAktualizaceSubscriberu
          );
7     connect(&customerInformationService2_2CZ1_0,&HttpSluzba::
          signalOdpovedNaPost,&testOdberuServer,&TestOdberuServer::
          slotVypisOdpovedServeru);
```

testdetekcebonjour.cpp Tato třída je založena pro budoucí test správného publikování IBIS-IP do DNS-SD.

8.5.3 qtzeroconf

qtzeroconf je knihovna importovaná jako vnořený projekt a tvoří rozhraní mezi DNS-SD a Qt. Vyžaduje pro svou funkci Bonjour na Windows (nyní jako Apple Print Services) a na Linuxu vyžaduje balík *avahi-client*. Tato knihovna je použita jak ve Vdv301publisher, tak ve Vdv301tester. [12]

8.5.4 VDV301struktury

cestaudaje.cpp Tato datová struktura ukládá veškeré proměnné týkající se stavu systému z hlediska informačního systému.

linka.cpp Ukládá dočasně veškeré parametry linky, jako například její typ (příznaky linky), číslo, alias.

obeh.cpp Struktura popisující oběh jako seznam následných spojů.

pasmo.cpp Struktura popisující název pásma a IDS, do kterého spadá.

spoj.cpp Struktura popisující spoj jako posloupnost zastávek/zastavení.

zastavka.cpp Struktura popisující zastávku jako takovou, hlavně její příznaky a název.

zastavkacil.cpp Tato struktura popisuje Zastávku z pohledu IBIS-IP a řídí se filozofií, že každá zastávka má svůj cíl, svůj seznam nácestných zastávek a své číslo linky. Toto je nutné například kvůli polookružním linkám, u kterých se mění název spoje v průběhu cesty.

8.5.5 Ostatní

hlasic.cpp Tato třída obsluhuje hlášení zastávek. Na základě vstupních informací skládá MP3 soubory do fronty a tyto pak přehrává. Pro přehrávání využívá knihovnu *QMediaPlayer*, která byla zvolena kvůli integraci v Qt balíku a kvůli své kompatibilitě napříč operačními systémy.

Bohužel má své nedostatky, například při přehrávání souborů z fronty s různým datovým tokem dochází ke krátké nezanedbatelné pomlce.

Řešením by mohlo být využití externích knihoven, například částí přehrávače VLC. Toto vy ovšem vyžadovalo instalování těchto částí do všech počítačů, na kterých má VDV301tester běžet.

ibisovladani.cpp Tato třída se stará o obsluhu periferií fungujících po sběrnici/protokolu IBIS (VDV300). Využívá knhovnu *QSerialPort* pro odesílání příkazů na sériový port a správné fungování vyžaduje speciální HW převodník RS232->IBIS, případně USB->IBIS. Bohužel se využití sériového portu ukazuje jako problematické a odesílání dat probíhá se značným zpožděním. Tento problém nebyl dále řešen, neboť tato funkcionalita byla spíše experimentální.

konfigurace.cpp Tato třída slouží jako příprava pro konfigurační soubor ve formátu XML (zápis/čtení/parsování), který by uchovával stavové informace mezi restarty programu. Hlavním účelem je ukládat nastavení složek pro data, například nastavenou cestu složky s hlášením.

Třída je v pokročílém stádiu rozpracování.

logfile.cpp Tato třída opět pracuje se souborem a ukládá stavové informace do souboru *logy/log.txt* a chybové hlášky programu společně s časovou značkou. Zatím se do logu zapisují hlavně chyby práce s databází a importu XML souboru.

Vzorek logu:

```
    st pro 1 15:10:48 2021 program spuštěn
st pro 1 15:11:33 2021 je databáze otevřena
st pro 1 15:11:33 2021 připojení se povedlo
st pro 1 15:11:33 2021 je databáze otevřena
st pro 1 15:11:33 2021 DB se povedlo zavřít
st pro 1 15:11:33 2021 je databáze otevřena
st pro 1 15:11:33 2021 DB se povedlo zavřít
st pro 1 15:11:53 2021 je databáze otevřena
st pro 1 15:11:53 2021 připojení se povedlo
st pro 1 15:11:53 2021 DB se povedlo zavřít
st pro 1 15:11:54 2021 je databáze otevřena
st pro 1 15:11:54 2021 DB se povedlo zavřít
```

main.cpp Tato třída je v běžných C++ programech tou nejdůležitější třídou, která propojuje ostatní třídy a kde probíhá definování globálních proměnných. V Qt frameworku jen spouští hlavní okno programu (MainWindow) a všechny další úkony probíhají tam. [2]

```
#include "mainwindow.h"
#include <QApplication>
#include <QDebug>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

mainwindow.cpp Nejdůležitější třída propojující všechny ostatní třídy. Stará se o práci prvky uživatelského rozhraní, jako jsou tlačítka, textová pole, popisky. Všechny

proměnné, které jsou považovány za globální, jsou umístěné do této třídy. Také jsou zde definovány instance ostatních tříd. I přes snahu o co největší převedení proměnných a funkcí do zvláštních tříd se jedná o třídu s největším počtem řádků (1274 pro .cpp soubor).

V této třídě také probíhá propojování slotů a signálů z ostatních tříd.

O důležitosti celé třídy a rozsahu vypovídá hlavičkový soubor:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QVector>
#include <QListWidget>
#include <QtSerialPort/QSerialPort>
#include <QNetworkAccessManager>
#include <QMainWindow>

#include "VDV301struktury/zastavka.h"
#include "VDV301struktury/linka.h"
#include "VDV301struktury/cestaudaje.h"
#include "VDV301struktury/zastavkacil.h"

#include "VDV301publisher/httpsluzba.h"
#include "VDV301publisher/customerinformationservice.h"
#include "VDV301publisher/ticketvalidationservice.h"

#include "VDV301testy/vdv301testy.h"
#include "VDV301testy/testodberuserver.h"
#include "VDV301testy/testdemo.h"

#include "sqlpraceropid.h"
#include "xmlmpvparser.h"
#include "xmlropidparser.h"
#include "ibisovladani.h"
#include "hlasic.h"
#include "konfigurace.h"
#include "logfile.h"



namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

    //konstanty
    QString umisteniProgramu=QCoreApplication::applicationDirPath();

    //datove struktury
    CestaUdaje stavSystemu;
    QVector <Linka> seznamLinek;
```

```
51      QVector <Spoj> seznamSpoju;
52      QVector <Obeh> seznamObehu;
53
54      bool platnostSpoje=1;
55
56      //SQLprace mojesql;
57      SqlPraceRopid mojesql;
58
59      //instance knihoven
60      XmlMpvParser xmlMpvParser;
61      XmlRopidParser xmlRopidParser;
62      IbisOvladani ibisOvladani;
63      Hlasic hlasic;
64      Logfile logfile;
65
66      //VDV301testy
67      TestDemo vzorovyTest;
68      TestOdberuServer testOdberuServer;
69      int testIndex=0;
70
71      QFile log;
72
73      //udalosti
74      void xmlHromadnyUpdate();
75      int priPrijezdu();
76      int priOdjezdu();
77      void vsechnyConnecty();
78      void testNaplnOkno(int index);
79
80      //IBIS-IP lsuzby
81      HttpSluzba deviceManagementService1_0;
82      CustomerInformationService customerInformationService1_0;
83      CustomerInformationService customerInformationService2_2CZ1_0;
84      TicketValidationService ticketValidationService2_3CZ1_0;
85
86      //konfiguracni soubor
87      Konfigurace konfigurace;
88
89      //vyberove dialogy
90      void NaplnVyberLinky(QVector<Linka> docasnySeznamLinek);
91      void NaplnVyberSpoje(QVector<Spoj> docasnySeznamSpoju);
92      void NaplnKmenoveLinky(QVector<Linka> docasnySeznamLinek);
93
94      void testStart(int index);
95      void testStop(int index);
96
97      QString otevriSouborXmlDialog();
98      void nastavLabelCestyXml();
99  private:
100     Ui::MainWindow *ui;
101     //void replyFinished(QNetworkReply *);
102     void AktualizaceDispleje();
103     void startDatabaze();
104     void inicializacePoli();
105     void vypisSubscribery(QVector<Subscriber> adresy);
106
107     void vypisSubscribery2(QVector<Subscriber> adresy);
108     void NaplnVyberPoradi(QVector<Obeh> docasnySeznamObehu);
```

```
109     void NaplnVyberTurnusSpoje(QVector<Spoj> docasnySeznamSpoju);
110     void zastavSluzby();
111     void toggleFullscreen();
112     void nastartujVsechnySluzby();
113
114
115 public slots:
116     void vypisSqlVysledek(QString vstup);
117     void testyVykresliCasti(QVector<PolozkaTestu> &seznamPolozek);
118 private slots:
119     //tlacitka
120     int on_prikaztlacitko_clicked();
121     void on_sipkaNahoru_clicked();
122     void on_sipkaDolu_clicked();
123     void on_ubratTlacitko_clicked();
124     void on_pripojeniTlacitko_clicked();
125     void on_pridatTlacitko_clicked();
126     void on_quitTlacitko_clicked();
127
128     void on_BeforeStop_clicked();
129     void on_AtStop_2_clicked();
130     void on_AfterStop_clicked();
131     void on_BetweenStop_clicked();
132
133     void on_tlacitkoNactiXMLropid_clicked();
134     void on_tlacitkoNastaveni_clicked();
135     void on_tlacitkoTruncate_clicked();
136     void on_tlacitkoOdesliPrikaz_clicked();
137     void on_tlacitkoNastavPort_clicked();
138     void on_tlacitkoIBIS_clicked();
139     void on_tlacitkoZpetVydej_clicked();
140     void on_tlacitkoLinkospoj_clicked();
141     void on_tlacitkoSmazOkno_clicked();
142     void on_tlacitkoManual_clicked();
143     void on_tlacitkoOdesliXml_clicked();
144
145     void on_tlacitkoAddsubscriber_clicked();
146     void on_tlacitkoRemoveSubscriber_clicked();
147     void on_tlacitkoHlaseniSlozka_clicked();
148     void on_tlacitkoAddsubscriber_2_clicked();
149     void on_tlacitkoRemoveSubscriber_2_clicked();
150     int on_prikazTlacitkoTurnus_clicked();
151     void on_quitTlacitko_2_clicked();
152     void on_tlacitkoPalubniPc_clicked();
153     void on_tlacitkoTestRozhrani_clicked();
154     void on_tlacitkoFullscreen_clicked();
155     void on_tlacitkoFullscreen2_clicked();
156     void on_tlacitkoTurnus_clicked();
157
158     //tlacitkaTestu
159     void on_tlacitkoPrubehTestu_clicked();
160     void on_tlacitko_StartTest_clicked();
161     void on_TlacitkoStopTest_clicked();
162     void on_pushButton_test1_clicked();
163     void on_pushButton_test2_clicked();
164     void on_pushButton_test3_clicked();
165     void on_pushButton_test4_clicked();
166     void on_tlacitkoSluzby_clicked();
```

```
167
168     //checkbox
169     void on_prestupyCheckbox_stateChanged(int arg1);
170     void on_checkBox_stateChanged(int arg1);
171
172     //radio buttons
173     void radio1(bool stav);
174     void radio2(bool stav);
175     void radio3(bool stav);
176      void radio4(bool stav);
177
178     //zmeny seznamu
179     void on_listSpoje_currentItemChanged(QListWidgetItem *current,
           QListWidgetItem *previous);
180     void on_listLinek_currentItemChanged(QListWidgetItem *current,
           QListWidgetItem *previous);
181     void on_listKmenovychLinek_currentItemChanged(QListWidgetItem *
           current, QListWidgetItem *previous);
182     void on_listPoradi_currentItemChanged(QListWidgetItem *current,
           QListWidgetItem *previous);
183     void on_listTurnusSpoje_currentItemChanged(QListWidgetItem *
           current, QListWidgetItem *previous);
184
185     //vlatni signaly
186     void MpvNetReady();
187     void vypisDiagnostika(QString vstup);
188
189
190
191
192     void on_tlacitkoXmlVyberCestu_clicked();
193 };
194
195 #endif // MAINWINDOW_H
```

prestupmpv.cpp Datová struktura ukládající jednotlivé přestupy a informace o nich.

xmlropidparser.cpp Tato třída se stará o celý proces importu vstupních dat od smazání databáze, přes otevření XML souboru až do importu nových dat do databáze.

sqlpraceropid.cpp Velice důležitá třída pracující se SQLite databází na úrovni čtení. Načítá data z databáze do dočasných datových struktur, často uložených do polí s proměnnou veélkou typu QVector.

Tato třída dotazy generuje i spouští, např. tento dotaz pro stáhnutí celé posloupnosti zastáve na základě vybraného linkospoje.

```
1     SELECT DISTINCT   z.n, z.tp, z.cis, z.ois, t.ri,t.hl, t.ctn, t.btn
          , t.lcdn, t.vtn, t.ctm, t.btm, t.lcdm, t.vtm, l.c, l.lc, l.tl,
           l.aois,l.noc, x.o, x.t, x.na, x.zn, x.xA, x.xB, x.xC, x.xD, x
          .xVla, x.xLet, x.xLod, x.xorder, x.zsol, s.ns, s.c
2     FROM x
3     LEFT JOIN s ON x.s_id=s.s
4     LEFT JOIN z ON x.u = z.u AND x.z=z.z
```

```
    LEFT JOIN l ON s.l=l.c
    LEFT JOIN t ON t.u=x.u AND t.z=x.z
    WHERE l.c=2096 AND s.c=1003 AND  s.kj LIKE '1\%'
    ORDER BY x.xorder
```

subscriber.cpp Datová struktura pro ukládání jednotlivých odběratelů. Její součástí je IP adresa a port, kam má palubní počítač zasílat data a struktura, kterou . Pokud zařízení odebírá z více struktur (například AllData a CurrentDisplayContent služby CustomerInformationSystem), je v seznamu uvedeno jako dva odběratelé.

xmlgenerator.cpp Tato třída slouží ke generování XML struktur IBIS-IP služeb. Funguje modulárně a tedy jeden typ vnořeného tagu vždy generuje jedna konkrétní funkce. Tato třída je v současnosti společná pro všechny služby a v budoucnu by se generování XML struktur mělo přesunout do třídy HttpSluzba a tříd z ní odvozených.

xmlmpvparser.cpp Tato služba se pomocí knihovny *QNetworkAccessManager* dotazuje na server MPVnet, který poskytuje jako odpověď XML soubor. Tento soubor pak třída zpracovává do seznamu prvků *prestupmpv*.

Chybející třída Pro plnou simulaci skutečného panelu není použita knihovna VDV301Publisher a tak není možné poskytovat službu DeviceManagementService. Toto bude vyřešeno v budoucích verzích.

9 Realizace VDV301display

Doplňkový program *VDV301display.exe* podporuje tyto funkce:

- zobrazení služeb publikovaných do DNS-SD
- přihlášení k odběru služby CustomerInformationService verze 2.2CZ1.0
- obnovení odběru po 120 s bez odeslání dat
- simulace LED panelů
- simulace LCD panelu

Program VDV301Display.exe slouží k vizualizaci dat odesílaných z palubního počítače a zároveň jako simulátor podřízeného zařízení.

Zobrazovaná data přijímá ze struktury *AllData* služby *CustomerInformationService*. Tato struktura obsahuje veškerá potřebná data jak pro LCD, tak pro LED panely.

V praxi se očekává, že LED zobrazovací panely budou využívat strukturu *CurrentDisplayContent*, která je pouze podmnožinou *AllData* a bude tedy vyžadovat slabší hardware v zobrazovacích panelech, hlavně z důvodu parsování menšího XML souboru.

LCD simulace Pro simulování LCD displeje byla zvolena metoda zobrazování SVG souboru, u kterého dochází nahrazováním textů za skutečné názvy zastávek.

Zobrazování piktogramů příznaků zastávek není v současné verzi implementováno.

Obrázek 34: Simulace TFT-LCD displeje v VDV301display.exe (autor, 2021)

Animace při přechodu mezi zastávkami je implementována na úrovni SVG souboru, ovšem použitá knihovna QtSVG není schopna tyto animace zobrazit.

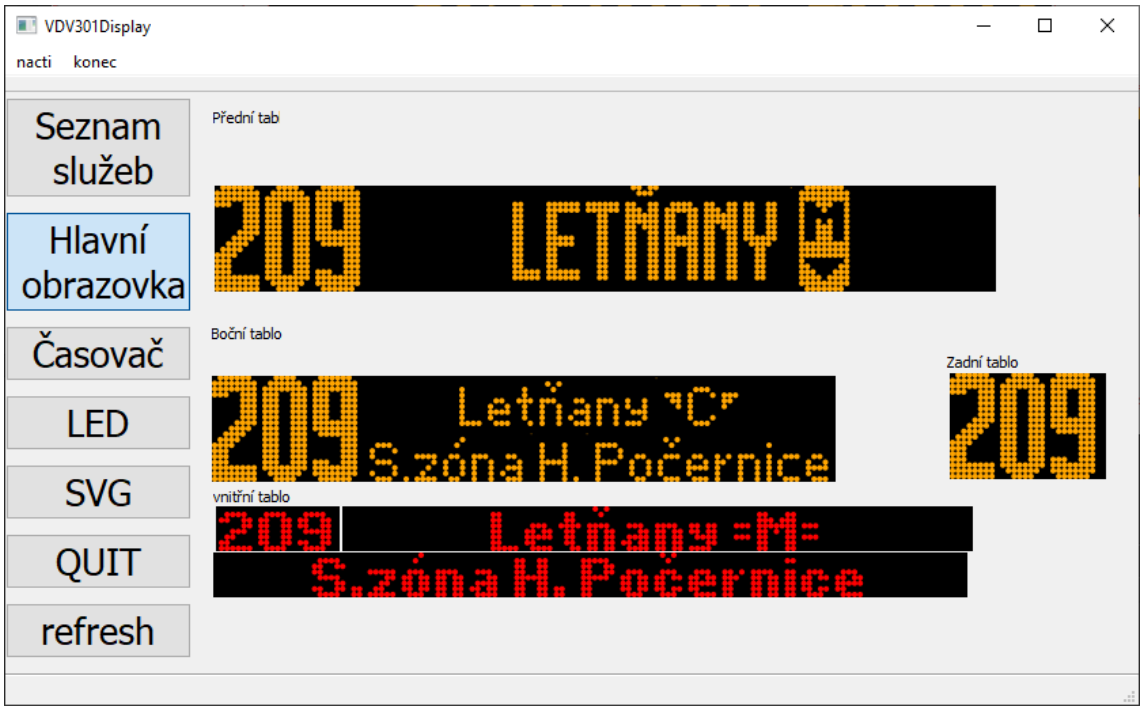
Pro příští verzi je plánován přechod na QtWebEngine, což je vykreslovací rozhraní založené na webovém prohlížeči Chromium. [5] Tohoto zatím nebylo využito, částečně kvůli HW náročnosti. Současná varianta využívající QtSvg funguje i na slabém HW, jako je Raspberry Pi Zero W, pro který již kvůli architektuře procesoru nelze knihovna *QtWebEngine* zkompilovat.

Funkčnost SVG animace byla ověřena pomocí internetového prohlížeče Microsoft Edge, který s Chromiem sdílí vykreslovací jádro.

LED simulace Program disponuje i simulací LED panelů. Tato slouží hlavně pro vizualizace rozložení názvů zastávek na konkrétním typu panelu. V backoffice *ASW JŘ* lze upravit název zastávky pro každý typ / polohu panelu (přední, boční, vnitřní) zvlášť a pokud současný text přetéká přes okraj, lze zvolit vhodnější zkratku názvu.

Pro vykreslení panelů jsou využity .ttf fonty, které vytvořil Bc. Michal Štursa ve webové službě FontStruct pro zjednodušení tvorby dokumentace Standardů kvality PID.[3] a poskytl tyto fonty k použití v této práci.

Simulované panely svými pixelovými rozměry (počet LED) odpovídají řadě zobrazovacích panelů BUSE BS210, které ze všech aktuálně provozovaných typů panelů mají nejměnší počet diod a proto je vhodné dimenzovat názvy zastávek právě na ně jako



Obrázek 35: Simulace LCD v Microsoft Edge, průběh překreslování animace (autor, 2021)

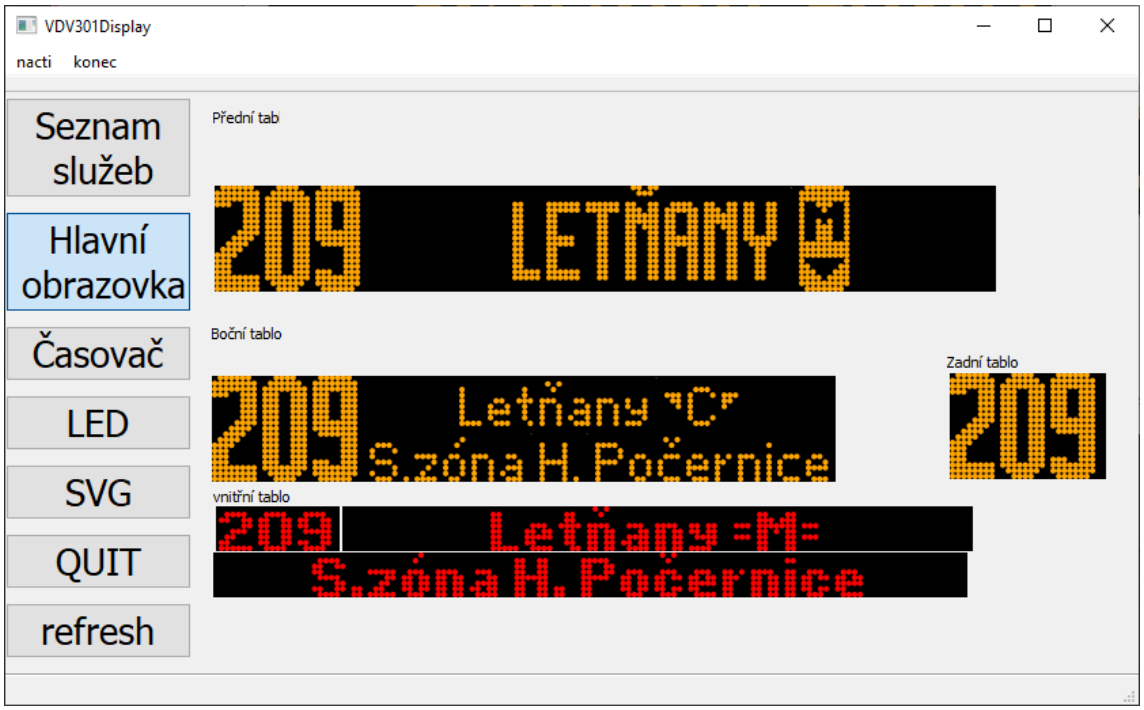
	Výška	Šířka
Přední	19	140
Boční	19	112
Zadní	19	28
Vnitřní	16	135

Tabulka 2: Virtuální rozměry simulovaných panelů BUSE BS210 (v bodech)

nejhorší scénář.

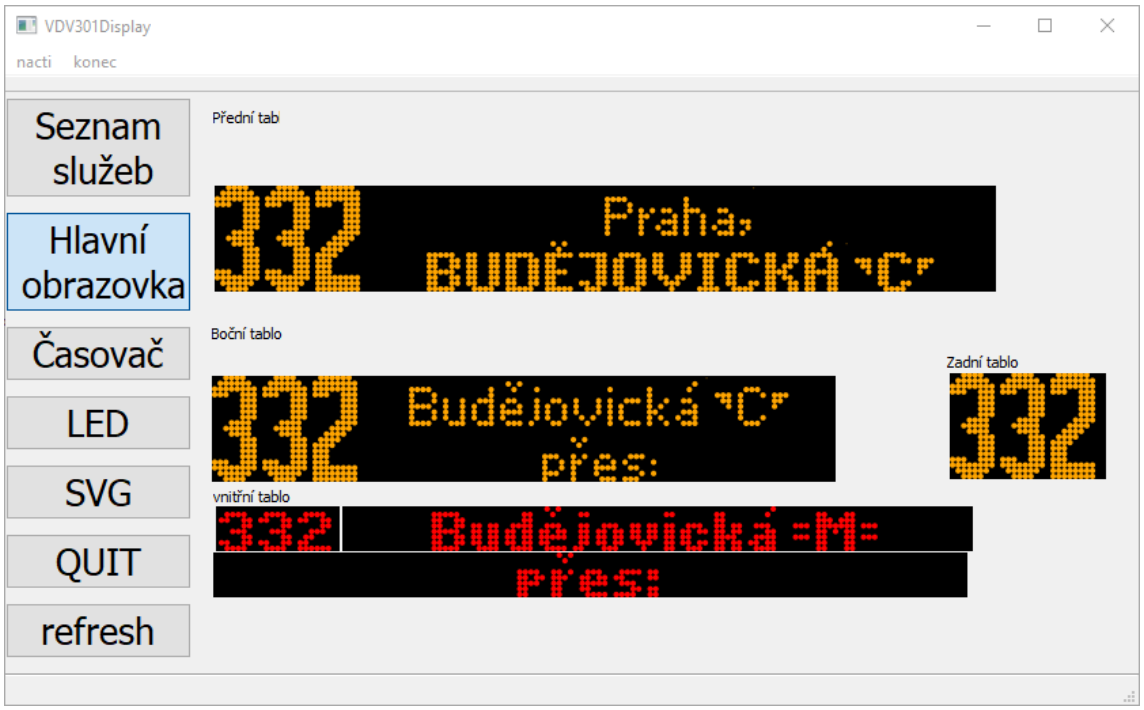
Modifikace normy VDV301, konkrétně 2.2CZ1.0 umožňuje zobrazování dvouřádkových cílů.

Toto je realizováno přidáním dvou tagů *DestinationFrontName*.



Obrázek 36: Simulace LED displejů v VDV301display.exe (autor, 2021)

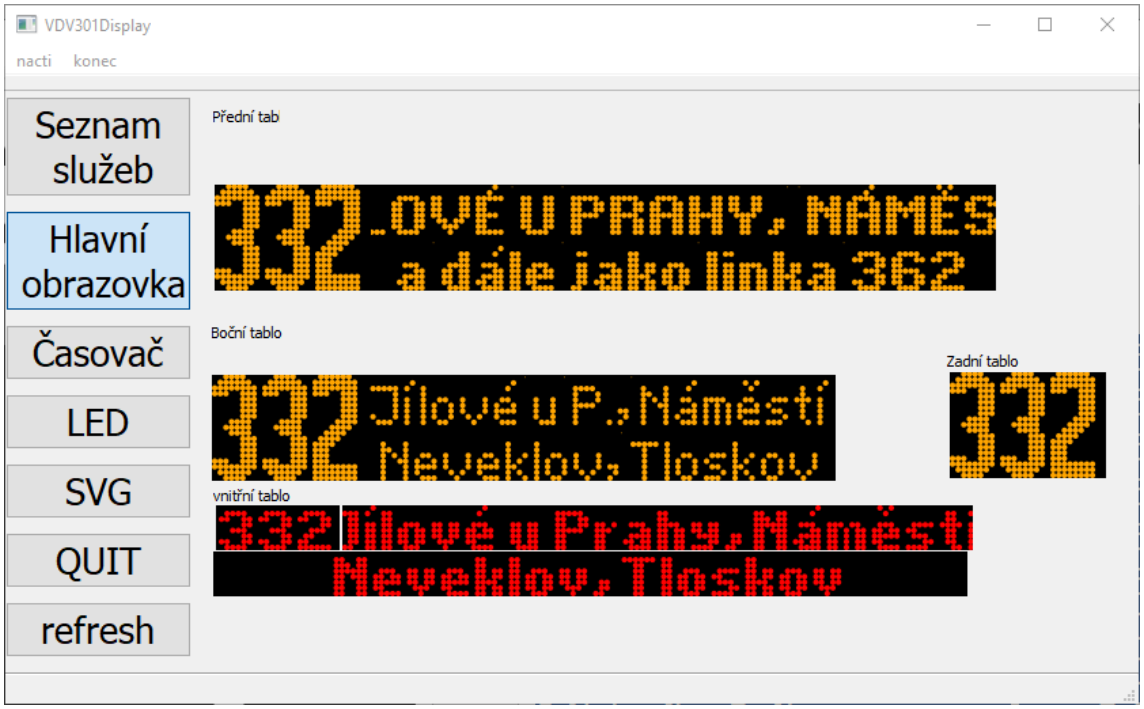
Obrázek 37: Stojan se skutečnými zobrazovacími panely (autor, 2021)



Obrázek 38: Simulace LED displejů v VDV301display.exe, dvouřádkový text (autor, 2021)

```
<Destination>
        <DestinationProperty>UndergroundC</DestinationProperty>
        <DestinationFrontName>
                <Value>`</Value>
                <Language>cz</Language>
        </DestinationFrontName>
        <DestinationFrontName>
                <Value>BUDĚJOVICKÁ [C]</Value>
                <Language>cz</Language>
        </DestinationFrontName>
        <DestinationSideName>
                <Value>Budějovická [C]</Value>
                <Language>cz</Language>
        </DestinationSideName>
        <DestinationRearName>
                <Value/>
                <Language>cz</Language>
        </DestinationRearName>
        <DestinationLcdName>
                <Value>Budějovická</Value>
                <Language>cz</Language>
        </DestinationLcdName>
        <DestinationInnerName>
                <Value>Budějovická =M=</Value>
                <Language>cz</Language>
        </DestinationInnerName>
</Destination>
```

Komunikační protokol stejným postupem umožňuje zobrazit i návazné spoje. V tomto případě je text cíle příliš dlouhý a vyžaduje optimalizaci.



Obrázek 39: Simulace LED displejů v VDV301display.exe, návazný spoj (autor, 2021)

```
<Destination>
```

```
        <DestinationFrontName>
                <Value>JÍLOVÉ U PRAHY,|NÁMĚSTÍ</Value>
                <Language>cz</Language>
        </DestinationFrontName>
        <DestinationFrontName>
                <Value>a dále jako linka 362</Value>
                <Language>cz</Language>
        </DestinationFrontName>
        <DestinationSideName>
                <Value>Jílové u P.,Náměstí</Value>
                <Language>cz</Language>
        </DestinationSideName>
        <DestinationRearName>
                <Value/>
                <Language>cz</Language>
        </DestinationRearName>
        <DestinationLcdName>
                <Value>Jílové u Prahy,Náměstí</Value>
                <Language>cz</Language>
        </DestinationLcdName>
        <DestinationInnerName>
                <Value>Jílové u Prahy,Náměstí</Value>
                <Language>cz</Language>
        </DestinationInnerName>
</Destination>
```

9.1 Zobrazení IBIS-IP služeb

VDV301tester dokáže detekovat a vypsat všechny IBIS-IP služby v lokální síti a zobrazit jejich IP adresu a port, na kterém přijímají požadavky.

9.2 VDV301display knihovny

9.2.1 VDV301subscriber

ibisipsubscriber.cpp Tato třída se stará o kompletní povinnosti IBIS-IP odběratele. Prochází DNS-SD záznamy a hledá služby, ze kterých může odebírat a přihlašuje se k odběru pomocí HTTP klienta. Obsahuje zároveň HTTP server pro příjem dat z POST requestů. Obsahuje nastavitelný časovač, který se spouští s přijetím

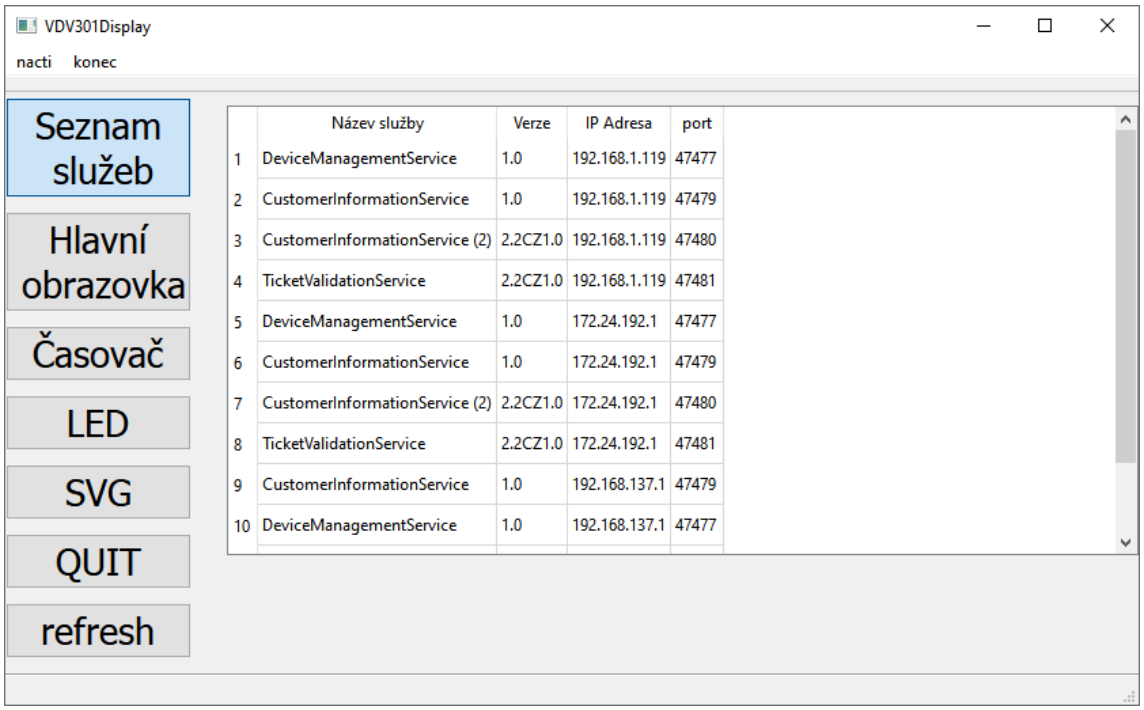
newhttpserver.cpp shodná s VDV301tester, vyžadována funkcí

9.3 VDV301struktury

shodné s VDV301tester.

9.4 Ostatní

main.cpp IDentická jako u VDV301tester.



Obrázek 40: Simulace LED displejů v VDV301display.exe, procházení IBIS-IP služeb (autor, 2021)

mainwindow.cpp Tato třída zde plní podobnou funkci jako v programu VDV301tester. Zde se stará o vykreslování LED panelů, což je zjednodušeno využitím speciálních fontů. Tyto fonty lze uložit do složky s programem a není tedy třeba je instalovat do systému.

Specialitou programu je rozdělení okna virtuálního zobrazovacího panelu na více prvků a změna mezi jednořádkovám a víceřádkovým textem probíhá skrýváním/zobrazováním jednotlivých elementů.

svgvykreslovani.cpp Tato třída se stará o nahrazování textů v SVG souboru. Vzhledem k principu práce s SVG souborem a jeho zobrazením je nutné otevřít SVG soubor, načíst jej do paměti, upravit (nahradit texty) a následně tato SVG data uložit zpět do souboru a teprve ze souboru je načíst.

xmlparser.cpp Tato funkce konvertuje data přijatá v POST requestu do vektorů tříd ze složky Vdv301struktury. Provádí přesně opačné operace než funkce xmlgenerator.

9.5 MPVnet přestupy

Data o přestupech jsou v provozu získávána ze systému MPVnet pomocí neveřejného XML API systému MPVnet a stejně je tomu i v testovacím programu.[1]

Pro nové instalace je předpokládáno využití datové platformy Golemio. Tato slouží pro sběr a poskytování Smart City dat hlavního města Prahy. Mězi těmito daty jsou i infor-

mace o poloze vozidel a virtuální zastávkové panely. Tyto informace jsou poskytovány přes veřejné JSON API.[8]

Pro použití obou datových zdrojů je třeba znát číslo zastávky. Pro MPVnet je to číslo odpovídající systému CIS JŘ (Celostátní informační systém o jízdních řádech)., pro Golemio lze využít i GTFS id a ASW_ID. [11] Tato čísla lze získat z číselníků zastávek z portálu `opendata.praha.eu`[13].

Strojově čitelný vzorový výstup z neveřejného API systému MPVnet (10 odjezdů):

```
<TBL cas="2021-11-29T14:47:54" ver="1.0.7999.19153" text="Podmínky použití: pid.cz/zis-podminky">
<t id="51888" stan="52,A,B,C,D,E,F,G,H,H30,H31,H39,H40,H41,H42,I,J,K,L,M1,M2" zast="Kobylisy" ciszast="51888">
<o stan="H" lin="177" alias="177" spoj="26" smer="Chodov" odj="2021-11-29T14:44:00+01:00" sled="true" zpoz="3" np="true" nad="false" dd="3" smer_c="56714"/>
<o stan="A" lin="10" alias="10" spoj="151" smer="Sídliště Řepy" odj="2021-11-29T14:47:00+01:00" sled="true" zpoz="0" np="false" nad="false" dd="2" smer_c="50697"/>
<o stan="A" lin="17" alias="17" spoj="312" smer="Sídliště Modřany" odj="2021-11-29T14:47:00+01:00" sled="true" zpoz="0" np="true" nad="false" blik="true" dd="2" smer_c="55248"/>
<o stan="C" lin="200" alias="200" spoj="27" smer="Sídliště Bohnice" odj="2021-11-29T14:48:00+01:00" sled="true" zpoz="0" np="true" nad="false" dd="3" smer_c="58860"/>
<o stan="I" lin="145" alias="145" spoj="31" smer="Kobylisy" odj="2021-11-29T14:48:00+01:00" sled="true" zpoz="0" np="true" nad="false" dd="3" smer_c="51888"/>
<o stan="H" lin="152" alias="152" spoj="16" smer="Českomoravská" odj="2021-11-29T14:48:00+01:00" sled="true" zpoz="1" np="true" nad="false" dd="3" smer_c="47178"/>
<o stan="H" lin="102" alias="102" spoj="63" smer="Šimůnkova" odj="2021-11-29T14:48:00+01:00" sled="true" zpoz="1" np="true" nad="false" dd="3" smer_c="59309"/>
<o stan="C" lin="177" alias="177" spoj="54" smer="Poliklinika Mazurská" odj="2021-11-29T14:49:00+01:00" sled="false" zpoz="0" np="true" nad="false" dd="3" smer_c="59234"/>
<o stan="M1" lin="C" alias="C" spoj="13" smer="Letňany" odj="2021-11-29T14:49:00+01:00" sled="false" zpoz="0" np="false" nad="false" dd="1" smer_c="47122"/>
<o stan="M2" lin="C" alias="C" spoj="476" smer="Háje" odj="2021-11-29T14:49:00+01:00" sled="false" zpoz="0" np="false" nad="false" dd="1" smer_c="55083"/>
</t>
</TBL>
```

Vzorek dat ze systému Golemio

(vzhledem k prostorové náročnosti formátu JSON je vypsáno pouze jedno stanoviště a jeden odjezd)

```
{
  "stops": [
    {
      "level_id": "U675L2",
      "location_type": 0,
      "parent_station": "U675S1",
      "platform_code": "1",
```

```
      "stop_id": "U675Z101P",
      "stop_lat": 50.12408,
      "stop_lon": 14.45428,
      "stop_name": "Kobylisy",
      "wheelchair_boarding": 1,
      "zone_id": "P",
      "asw_id": {
        "node": 675,
        "stop": 101
      }
    }
  ],
  "departures": [
    {
      "arrival_timestamp": {
        "predicted": "2021-11-29T13:51:22.000Z",
        "scheduled": "2021-11-29T13:51:00.000Z"
      },
      "delay": {
        "is_available": true,
        "minutes": 0,
        "seconds": 22
      },
      "departure_timestamp": {
        "predicted": "2021-11-29T13:51:22.000Z",
        "scheduled": "2021-11-29T13:51:00.000Z"
      },
      "last_stop": {
        "id": "U267Z2P",
        "name": "Služská"
      },
      "route": {
        "short_name": "177",
        "type": 3,
        "is_night": false,
        "is_regional": false,
        "is_substitute_transport": false
      },
      "stop": {
        "id": "U675Z8P",
        "platform_code": "H"
      },
      "trip": {
        "direction": {},
        "headsign": "Chodov",
        "id": "177_1492_210924",
        "is_at_stop": false,
        "is_canceled": false,
        "is_wheelchair_accessible": true,
        "short_name": {}
      }
    }
  ],
  "infotexts": []
}
```

9.5.1 Konfigurace sítě

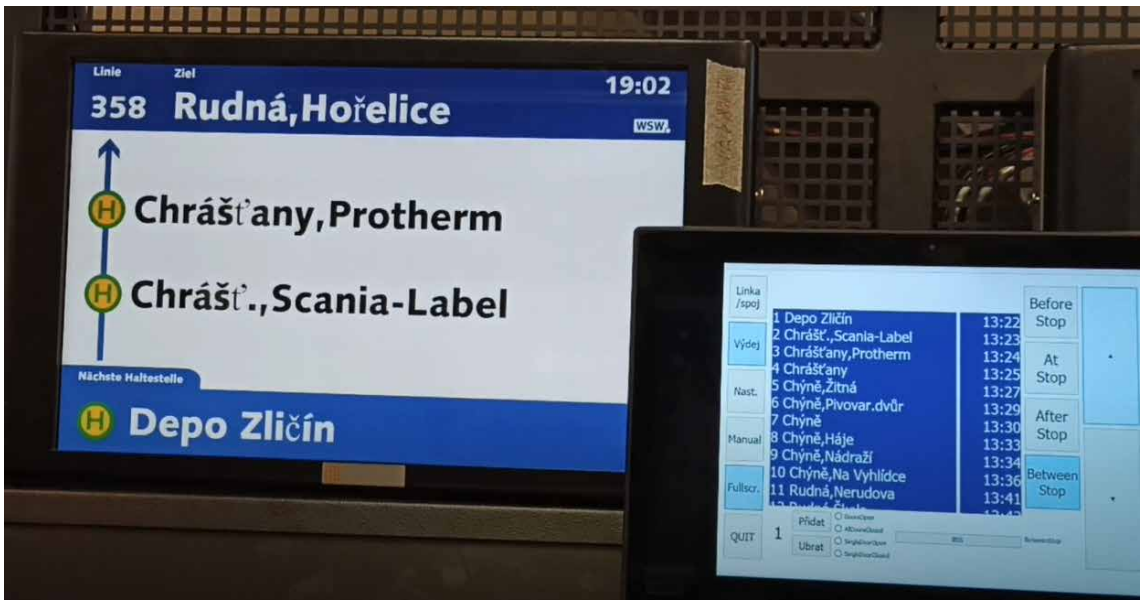
Nespornou výhodou celého systému VDV301 je možnost využití automatické konfigurace sítě pomocí DHCP, případně tzv. „link local“ adresy. Není tedy nutné se zabývat manuálním nastavováním statických IP adres. HTTP server lze nakonfigurovat i tak, že jsou porty pro komunikaci voleny dynamicky, avšak pro účely testů (např. výjimky ve firewallu) jsou pevně nastaveny na tyto hodnoty:

Název služby	verze	VDV301tester vysílání	VDV301displej příjem
CustomerInformationService	1.0	47479	x
CustomerInformationService	2.2CZ1.0	47480	48479
DeviceManagementService	1.0	47477	x
TicketValidationService	2.2CZ1.0	47481	x

Tabulka 3: Konfigurace portů

10 Ověření na reálném zařízení

V rámci této diplomové práce byla správnost chování testovací sestavy validována vůči skutečnému zařízení Bustec BT719 v konfiguraci pro německé město Wuppertal, komunikující po protokolu VDV301 verze 1.0 s doplněním piktogramů přestupů.



Obrázek 41: Ověření funkčnosti s reálným panelem - hlavní obrazovka (autor, 2021)

Obrázek 42: Ověření funkčnosti s reálným panelem - obrazovka přestupů (autor, 2021)

11 Závěr

Tato diplomová práce opět prokázala, že implementace protokolu IBIS-IP (dle normy VDV301) je realizovatelná pomocí běžně dostupných nástrojů a knihoven.

Totéž platí o testování zařízení pracujících dle těchto specifikací a základní testy lze dokonce provádět pomocí již hotových nástrojů.

Celý komunikační protokol má velmi dobrou dokumentaci a teoreticky vyžaduje minimální využití znalostí mimo základy programování s využitím základních síťových technologií.

V praxi se naopak ukázalo, že první dostupný nástroj nemusí být vždy ten správný. Například webový server bylo nutné vyměnit po zkušenostech z bakalářské práce, kdy se dříve vybraná knihovna ukázala jako značně nespolehlivá při větší zátěži.

Po vyřešení počátečních potíží vznikly programy (VDV301tester a VDV301display), které již posloužily v praxi při vývoji pro PID specifické verze 2.2CZ1.0 protokolu IBIS-IP a ukázaly, že je velice složité navrhovat nová řešení bez praktického ověření jejich funkčnosti.

Otevřenost a jednoduchost implementace předurčují protokol VDV301 jako jednu z možností, jak nahradit ostatní zastaralé způsoby komunikace zařízení vozidlového informačního systému ve veřejné hromadné dopravě. Poskytnutí programů vzniklých v této práci vývojářům reálných zařízení by mohlo pomoci v rozšíření tohoto protokolu a modernizaci morálně zastaralých zařízení.

12 Literatura

Reference

[1] **EICHLER, Adam.** Model komunikace vozidlového informačního systému pro cestující [online]. Praha, 2019. [vid. 2021-11-28]. Dostupné z: https://dspace.cvut.cz/handle/10467/85249. Bakalářská práce. České vysoké učení technické v Praze.

[2] **LAZAR, Guillaume, PENEA, Robin.** Mastering Qt 5. Packt Publishing Limited, 2016. ISBN: 9781786467126.

[3] **ŠTURSA, M.,BRYNDA, F.** Odbavovací a informační zařízení ve vozidlech PID. In: PID - Pražská integrovaná doprava [online]. ROPID, srpen 2021. [vid. 2021-08-20]. Dostupné z: https://pid.cz/wp-content/uploads/2021/07/Odbavovaci-a-informacni-zarizeni.pdf

[4] **VDV.** Internet protocol based communication services in public transport. In: VDV Die Verkehrs-unternehmen [online]. Verband Deutscher Verkehrsunternehmen, 2021. [vid. 2021-11-28]. Dostupné z: https://www.vdv.de/ip-kom-oev.aspx

[5] **THE QT COMPANY.** Qt WebEngine Overview. In: Qt Documentation [online]. The Qt Company, 2020. [vid. 2021-11-28]. Dostupné z: https://doc.qt.io/qt-5/qtwebengine-overview.html

[6] **THE QT COMPANY.** SQL Database Drivers. In: Qt Documentation [online]. The Qt Company, 2020. [vid. 2021-11-28]. Dostupné z: https://doc.qt.io/qt-5/sql-driver.html

[7] **DE VLEESCHAUWER, Koen.** Qt5.15.2 LTS with OpenGL for Raspberry. In: GitHub [online]. GitHub, Inc., 2021. [vid. 2021-11-28]. Dostupné z: https://github.com/koendv/qt5-opengl-raspberrypi

[8] **OPERÁTOR ICT.** O projektu: Co je Datová platforma Golemio?. In: GOLEMIO Prague City Data [online]. Operátor ICT, a. s., 2018. [vid. 2021-11-28]. Dostupné z: https://golemio.cz/cs/o-projektu

[9] **KABELOVÁ A., DOSTÁLEK L.** Velký průvodce protokoly TCP/IP a systémem DNS. 5. vyd. Brno: Computer Press, 2012. ISBN: 978-80-251-2236-5

[10] **MARKACZ, T.** Testování integrací client-server [online]. Praha: ČVUT, 2018. Diplomová práce, ČVUT v Praze, Fakulta elektrotechnická. [vid. 2021-11-28]. Dostupné z: https://dspace.cvut.cz/handle/10467/76943

[11] **OPERÁTOR ICT.** PID Departure Boards. In: Golemio API [online]. Oracle and/or its affiliates, 2021. [vid. 2021-11-28]. Dostupné z: https://golemioapi.docs.apiary.io/#reference/public-transport/departure-boards/get-departure-board

[12] **JBAGG.** QtZeroConf. In: GitHub [online]. GitHub, Inc., 2021. [vid. 2021-11-28]. Dostupné z: https://github.com/jbagg/QtZeroConf

[13] **ROPID.** Zastávky a označníky PID. In: opendata hlavního města Prahy [online]. Operátor ICT, a. s., 2021. [vid. 2021-11-28]. Dostupné z: https://opendata.praha.eu/dataset/zastavky-a-oznacniky-pid

[14] **POSTMAN.** Test script examples. In: Postman Learning Center [online]. Postman, Inc., 2021. [vid. 2021-11-28]. Dostupné z: https://learning.postman.com/docs/writing-scripts/script-references/test-examples/

[15] **THE QT COMPANY.** qthttpserver. In: GitHub [online]. GitHub, Inc., 2021. [vid. 2021-11-28]. Dostupné z: https://github.com/qt-labs/qthttpserver/blob/master/.qmake.conf

13 Seznam obrázků a tabulek

Seznam obrázků

Seznam tabulek

14 Seznam použitých zkratek

API	Application programming interface
ARM	Advanced RISC Machine
ASW	aplikační SW
AVL	Automatic vehicle location
CDV	Centrum dopravního výzkumu
CIS JŘ	Celostátní informační systém o jízdních řádech
ČVUT	České vysoké učení technické v Praze
DB	databáze
DNS	Domain Name System
DNS-SD	Domain Name System – Service Discovery
FD	Fakulta dopravní
FTP	foil screened twisted pair
GNSS	Global Navigation Satellite System
HTTP	Hypertext Transfer Protocol
HW	hardware
ID	identifikátor
IP	internet protocol
JDF	jednotný datový formát
LCD	liquid-crystal display
LED	light emitting diode
mDNS	Multicast DNS
MHD	městská hromadná doprava
MPV	Monitorování provozu vozidel
OBU	On-Board Unit
OIS	Odbavovací a informační systém
PC	personal computer
PID	Pražská integrovaná doprava
PMDP	Plzeňské městské dopravní podniky a.s.
ROPID	Regionální organizátor Pražské integrované dopravy
SQL	Structured Query Language
SW	software
TCP/IP	Transmission Control Protocol/Internet Protocol
TFT-LCD	Thin-film-transistor liquid-crystal display
UDP	User Datagram Protocol
UI	user interface
VDV	Verband Deutscher Verkehrsunternehmen (sdružení německých dopravních podniků)
XML	eXtensible Markup Language
XSD	XML Schema Definition

15 Seznam příloh - USB disk s následujícím obsahem:

Složka zdrojové kódy Obsahuje kompletní zdrojové kódy obou programů, lze zkompilovat pomocí Qt verze 5.15.2 a vyšší pod OS Linux i Windows.

Složka Release Obsahuje oba zkompilované programy pro spuštění ve Windows. Pro správnou funčnost je vyžadována instalace Bonjour. VDV301display není k dispozici ve webové verzi kvůli omezení maximální velikosti souboru. Oba programy lze stáhnout z GitHubu z odkazů uvedených v textu.