TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Monitorování výroby pomocí Andon TV

Diplomová práce

Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie

Autor práce: **Bc. Tadeáš Vasko**
Vedoucí práce: Ing. Igor Kopetschke
Ústav nových technologií a aplikované informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Monitorování výroby pomocí Andon TV

Jméno a příjmení: **Bc. Tadeáš Vasko**
Osobní číslo: M20000176
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2021/2022**

Zásady pro vypracování:

1. Proveďte rešerši případných existujících řešení a zdůvodněte nutnost vývoje vlastní aplikace.
2. Navrhněte všechny potřebné funkcionality aplikace včetně použitých protokolů, datových modelů a algoritmů.
3. Implementujte navržené řešení, nasaďte jej do testovacího provozu a získejte zpětnou vazbu. Pro vývoj aplikace použijte programovací jazyk Dart a software Flutter.
4. Na základě výsledků zpětné vazby proveďte případné úpravy a aplikaci nasaďte do ostrého provozu.
5. Uveďte další možná vylepšení a rozšíření.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	40-50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] BALBAERT, Ivo a Dzenan RIDJANOVIC, 2015. Learning Dart. 2th. edition. Birmingham, UK: Packt Publishing Limited. ISBN 9781785287626.

[2] MODBUS Protocol Specification [online], 2012. 50 [cit. 2021-10-8]. Dostupné z: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[3] PAYNE, Rap, 2019. Beginning App Development with Flutter: Create Cross-Platform Mobile Apps. 1th. edition. New York, USA: Apress. ISBN 9781484251812.

Vedoucí práce:	Ing. Igor Kopetschke Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2021
Předpokládaný termín odevzdání:	16. května 2022

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2021

Prohlášení

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

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

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

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

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

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

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

15. 5. 2022 Bc. Tadeáš Vasko

Monitorování výroby pomocí Andon TV

Abstrakt

Cílem této diplomové práce je vytvoření Andon systému pro televizi se systémem Android. V práci je napsána motivace, která z důvodňuje fakty, proč tato práce vznikla, ačkoli Andon systémy existují. Práce se zabývá popisem celé aplikace, konkrétně jsou popsány jednotlivé použité typy komunikací, funkcionality, datové modely a jednotlivé algoritmy. V práci je také rozepsáno testování samotného řešení, které bylo nezbytné k detekci chyb a ověření správného fungování celého systému. Aplikace, díky použitému frameworku Flutter a programovacího jazyka Dart, je multiplatformní a je možné ji spustit jednak na televizi s operačním systémem Android, tak také na počítači s operačním systémem Windows. Práce se také zabývá výhodami a nevýhodami použitého frameworku, kde je i srovnáván s frameworkem React Native, se kterým lze také vytvořit multiplatformní aplikace. V závěru práce jsou navržena možná vylepšení a rozšíření stávající aplikace, která by toto řešení dokázala vylepšit.

Klíčová slova: Andon, PLC, Flutter, Dart

Production monitoring using Andon TV

Abstract

The aim of this thesis is to create an Andon system for Android TV. In the thesis a motivation is written that give reasons the facts why this thesis was created although Andon systems exist. The thesis deals with the description of the whole application, specifically the different types of communication used, the functionalities, the data models and the different algorithms. The thesis also discusses the testing of the solution itself, which was necessary to detect bugs and verify the correct functioning of the entire system. The application, thanks to the Flutter framework and the Dart programming language used, is cross-platform and can be run on both an Android TV and a Windows PC. The thesis also discusses the advantages and disadvantages of the framework used where it is also compared with the React Native framework with which multiplatform applications can also be created. The thesis concludes by suggesting possible improvements and extensions to the existing application that could improve this solution.

Keywords: Andon, PLC, Flutter, Dart

Poděkování

Rád bych poděkoval Ing. Igorovi Kopetschke, Michalovi Čermákovi a Clément Debette za pomoc a cenné rady při vzniku této práce.

Obsah

Seznam obrázků

Seznam zkratek

IoT	Internet věcí
PLC	Programovatelný logický automat
HTTP	Hypertext Transfer Protocol
SDK	Software development kit
OEM	Original Equipment Manufacturer
API	Application Programming Interface
UX	User Experience
JSC	JavaScriptCore
JIT	Just in Time
GUI	Grafické uživatelské rozhraní
HTML	Hypertext Markup Language
AOT	Ahead of Time
JSI	JavaScript Interface
OEE	Overall Equipment Effectiveness

1 Úvod

Od roku 2011, kdy byl na hannoverském veletrhu veřejně představen termín „Průmysl 4.0“, se tato revoluce v průmyslu stala velkým a aktuálním tématem téměř všech výrobních podniků po celém světě. [10] Využívání high-tech IoT (internet věcí) zařízení vede k vyšší produktivitě a lepší kvalitě. Jedním z nástrojů, používaných v průmyslu 4.0., jsou i vizuální systémy. Mezi tyto vizuální nástroje patří systém Andon, který je vyvinut v této práci.

Andon systémy dokáží v reálném čase informovat pracovníky nebo management o stavu výroby nebo o vzniklých a hlavně aktuálních problémech při výrobě. Firem, které vyvinuly vlastní Andon systém, lze najít celou řadu, ale tyto systémy jsou placené a navíc je ve většině případů nutné dokoupit další hardware k jejich fungování. Navíc hodně existujících Andon systémů nepodporuje všechny typy PLC zařízení, ze kterých se data získávají. Předmětem této práce bylo proto vyvinout Andon systém, který bude volně dostupný komukoli, bez nutnosti za tento systém platit. Navíc zákazník potřebuje pouze zařízení, kde se mu budou data zobrazovat. Konkrétně se v tomto případě jedná o televizi se systémem Android, kterou lze koupit za velmi příznivé ceny. Celé řešení je navrženo tak, aby si uživatel pouze stáhl a nainstaloval tuto aplikaci do televize a ihned ji mohl začít používat.

Řešení bylo naprogramováno ve frameworku Flutter za použití programovacího jazyka Dart od společnosti Google. Výhoda toho frameworku a jazyka je ta, že dokáže vytvářet multiplatformní aplikace s jednou kódovou základnou. Přestože je aplikace navržena pro televize se systémem Android, lze toto řešení taktéž spustit jako desktopovou aplikaci, neboli aplikaci pro počítače.

Aplikace zpracovává typy komunikací, díky kterým aplikace získává data z PLC zařízení. Typy komunikací byly vybrány tak, aby zákazník nemusel brát zřetel na to, jaký typ PLC je použit v dané výrobě a mohl tuto aplikaci využít ke sledování důležitých dat v reálném čase. Řešení také myslí na to, že v každém výrobním procesu mohou být data různého typu a různé struktury. Proto je vyvinutá aplikace velmi univerzální a dokáže se přizpůsobit různým strukturám dat. Aplikace nezobrazuje pouze syrová data v tabulkách, a tak jsou v této práci také rozepsány jednotlivé typy vizualizace dat, které si uživatel může zvolit. Díky tomu si uživatel může lépe představit vývoj a aktuální stav výroby. Lze předpokládat, že uživatel potřebuje vidět více různých dat zároveň, a tak řešení obsahuje různé layouty, které si uživatel může definovat dle své představy. Díky těmto layoutům si může zobrazit

různá data v různých formách vizualizace na jedné obrazovce.

Vzhledem k tomu, že se vyvíjelo nové řešení Andon systémů, bylo nutné aplikaci otestovat. Testování probíhalo pomocí frameworku Flutter a také pomocí reálných dat z výrobních linek. Tato data posílal reálný PLC systém, kde pro testovací účely bylo třeba vytvořit odpovídající program. Samotná aplikace byla testována jak v emulátorech, tak také na reálné televizi, kde se testovalo, zda aplikace funguje, jak má a správně zobrazuje přijímaná data.

2 Motivace

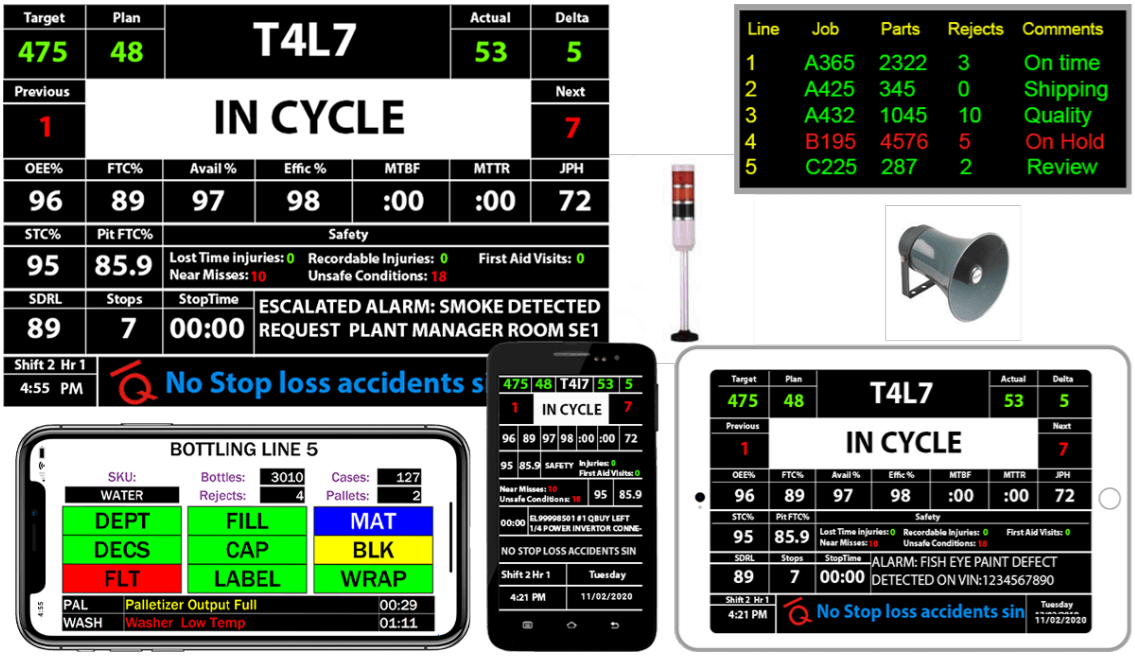
Motivací této práce je fakt, že neexistuje žádná volně dostupná aplikace tohoto typu. Většina řešení, dostupných na trhu, byla vytvořena na přelomu tisíciletí, tudíž jsou z dnešního pohledu zastaralá. Navíc zobrazují pouze data v textové formě. Svá řešení nabízejí například firmy Infinity Automation Systems Private Limited nebo Aveva, jejichž řešení Andon systému zobrazuje data v tabulkách, popřípadě si uživatel může zvolit, jaká konkrétní data chce sledovat (viz obrázky 2.1 a 2.2).

Obrázek 2.1: Ukázka Andon systému od Automation Systems Private Limited

Další nalezená řešení jsou nejen komerčně licencovaná a dostupná za poplatek, ale kromě samotného zobrazovacího zařízení vyžadují také další hardware. Většina řešení využívá počítač / Raspberry Pi a televize. Takovéto řešení využívají například firmy Atscada nebo Contec. Aby jejich Andon systémy mohli fungovat, je zákazník nucen si dokoupit další hardware a tím pádem se jeho pořizovací a provozní náklady zvyšují.

Nalezena byla také řešení, která využívají webserver PLC zařízení. Bohužel webserver není dostupný na všech PLC, tudíž se trh pro tato řešení zmenšuje a takové firmy nemohou svá řešení nabídnout komukoli. Na trhu je rovněž spousta Andon systémů, zaměřených pouze na jeden typ PLC, což z marketingového hlediska není příliš ideální.

Řešení, které bude popsáno v nadcházejících částech této práce, lze využít pro všechny typy PLC, které na trhu jsou. Řešením je plug&play („Připoj a Hraj“), tj.



Obrázek 2.2: Ukázka Andon systému od Aveva

vytvoření Andon televize pro zobrazení základních hodnot výrobních procesů bez nutnosti zásadního programování a přidávání dalšího hardwaru. K fungování celého řešení je nutná pouze televize se systémem Android, která musí být připojena do stejné sítě jako PLC, ze kterého data získává. Krom televize lze také v případě potřeby využít počítač, jelikož aplikace je multiplatformní. Řešení umožňuje jednoduše a v krátkém čase dosáhnout požadovaného výsledku a není ani potřeba disponovat velkými znalostmi z oblasti programování.

Vzhledem k tomu, že tento systém bude nasazen do výroby a bude se zobrazovat na televizích, lze předpokládat, že nebude volně k dispozici dálkový ovladač, aby neproškolené osoby nemohly například televizi přenastavit nebo třeba vypnout. Díky tomuto předpokladu je v nasazen HTTP server, aby se aplikace dala ovládat vzdáleně pomocí zařízení, která mají přístup k webovém prohlížeči, jako jsou počítače nebo mobilní telefony.

3 Flutter

Flutter je bezplatný a open-source framework uživatelského rozhraní společnosti Google pro vytváření nativně zkompilovaných aplikací pro mobilní zařízení, web, stolní počítače a vestavěná zařízení z jediné kódové základny. Díky této možnosti je vytváření multiplatformních aplikací jednodušší a rychlejší. Framework Flutter se skládá ze sady pro vývoj softwaru (SDK) a knihovny uživatelského rozhraní založené na widgetech. Tato knihovna zahrnuje různé opakovaně použitelné prvky uživatelského rozhraní, jako jsou tlačítka, textové vstupy nebo posuvníky.

Programovací jazyk fremeworku Flutter se nazývá Dart. Dart je objektový programovací jazyk, který se zaměřuje na vývoj nejen front-endu. Je to moderní alternativa jazyka JavaScript. Vzhledem k tomu, že se jedná o jazyk oficiálně podporovaný společností Google, zdá se, že podpora vývoje aplikací pro Android pomocí jazyka Java pomalu mizí. Takovéto aplikace navíc běží hladce a efektivně a také poskytují stejný výkon jako nativní aplikace. Flutter má také skvělou sadu testovacích a ladicích nástrojů pro optimalizaci aplikace.

3.1 Výhody frameworku Flutter

Mezi hlavní důvody, proč si vývojové týmy vybírají Flutter, patří:

- Zvýšená produktivita. Používání stejné kódové základny pro iOS, Android, web, stolní počítače a vestavěná zařízení. Šetří čas i zdroje, jelikož nativní widgety Flutteru také minimalizují čas strávený testováním, protože zajišťují, že u různých verzí operačních systémů nejsou téměř žádné problémy s kompatibilitou.
- Snadné učení. Flutter umožňuje vývojářům vytvářet nativní mobilní aplikace, aniž by museli přistupovat k OEM widgetům (nativní komponenty uživatelského rozhraní) nebo používat velké množství kódu. To, kromě mimořádně atraktivního uživatelského rozhraní Flutteru, výrazně zjednodušuje proces tvorby mobilních aplikací.
- Vysoký výkon. Aplikace vytvořené frameworkem Flutter fungují stejně dobře jako nativní aplikace.
- Cenově výhodné. Vytváření multiplatformních aplikací se stejnou kódovou základnou je v podstatě vytváření více aplikací za cenu jedné.

- K dispozici v různých vývojových prostředích. Vývojáři si mohou při úpravách kódu ve Flutteru vybrat mezi Android Studiem a VS Code.
- Skvělá dokumentace a komunita. Flutter má, díky rozsáhlé dokumentaci se snadno popsanými případy, k dispozici mnoho skvělých zdrojů, které vám odpoví na vaše otázky. Uživatelé Flutteru také mohou využívat komunitní centra, jako jsou Flutter Community a Flutter Awesome, kde si mohou vyměňovat nápady.

3.2 Nevýhody vývoje aplikací Flutter

I když vývoj aplikací Flutter přináší mnoho pozitiv, je důležité vzít v úvahu i méně významné aspekty tohoto frameworku:

Flutter je relativně nový. Nabízí mnoho zásuvných modulů a komponent uživatelského rozhraní, ale frameworky jako Xamarin a React Native poskytují mnohem větší výběr. To je současná situace, ale vývoj pokračuje. Některé komponenty jsou k dispozici pouze pro iOS nebo Android, ale ne pro oba systémy současně. Tyto typy komponent častěji podporují systém Android, protože Flutter pochází od společnosti Google a vývojáři pro systém Android se o Flutter obvykle zajímají více než vývojáři pro systém iOS.

3.3 Testování aplikací Flutter

Automatizované testy Flutteru se dělí do tří hlavních kategorií:

- Unit testování
- Testování widgetů
- Integrační testování

Zatímco testy jednotek testují jednu funkci, třídu nebo metodu, testy widgetů (známé také jako testy komponent) testují jeden widget. Testy jednotek a widgetů tvoří většinu testů Flutteru, které se sledují podle pokrytí kódu. Integrační testy testují buď velkou část, nebo celou aplikaci a pokrývají všechny důležité případy použití.

3.3.1 Unit testování

Unit testy aplikace Flutter ověřují chování jedné metody, funkce nebo třídy. Jinými slovy, unit testy zajišťují, že testovaná metoda poskytuje očekávaný výsledek na základě zadaného vstupu. Pomáhají uživatelům psát lépe testovatelný a udržovatelný kód. Uživatelé mohou spouštět automatizované unit testy prostřednictvím zásuvných modulů Flutter pro IntelliJ a VSCode. Kromě podpory spouštění testů

poskytují IntelliJ a VSCode také nejrychlejší zpětnou vazbu a umožňují uživatelům nastavit body přerušení.

3.3.2 Testování widgetů

Testy widgetů, v jiných frameworcích uživatelského rozhraní známé také jako testy komponent, ověřují, zda uživatelské rozhraní widgetu vypadá a chová se při reakci na určité interakce podle očekávání. Je mnohem komplexnější než jednotkový test, protože zahrnuje více tříd a vyžaduje testovací prostředí, které poskytuje příslušný kontext životního cyklu widgetu. Zároveň je však testovací prostředí widgetu mnohem jednodušší implementací než rozvinutější systém uživatelského rozhraní. Uživatelé nástroje Flutter by měli psát testy widgetů pro všechny běžné widgety, které pro své aplikace používají.

3.3.3 Integrační testování

Integrační testy testují buď celou aplikaci, nebo její velkou část. Tyto testy ověřují výkonnost aplikace a zajišťují, že všechny testované widgety a služby správně spolupracují. Uživatelé nástroje Flutter provádějí integrační testy buď na skutečných zařízeních, nebo na virtuálních zařízeních, jako jsou simulátory a emulátory. Tyto testy probíhají ve dvou krocích: nasazení testované aplikace do zařízení (skutečného nebo virtuálního) a následné řízení aplikace ze samostatné sady testů. Tím se zkontroluje, zda vše dohromady funguje podle očekávání.

3.4 Dart

Dart je otevřený, univerzální, objektově orientovaný programovací jazyk se syntaxí ve stylu jazyka C, který byl vyvinut společností Google v roce 2011. Je typově bezpečný, neboť používá statickou kontrolu typů, což zajišťuje, že hodnota proměnné vždy odpovídá statickému typu proměnné. Ačkoli jsou typy povinné, typové anotace jsou kvůli typové interferenci nepovinné. Systém typování Dart je také flexibilní a umožňuje použití dynamického typu v kombinaci s kontrolou za běhu, což může být užitečné při experimentování nebo pro kód, který musí být obzvláště dynamický. Někdy se tomu také říká zdravé typování. Účelem programování v jazyce Dart je vývoj rychlých aplikací na libovolné platformě. Jeho cílem je nabídnout nejproduktivnější programovací jazyk pro vývoj na více platformách ve spojení s flexibilní platformou pro spouštění aplikačních rámců. Dart je inspirován dalšími programovacími jazyky, jako jsou Java, JavaScript, C#. Protože Dart je kompilovaný jazyk, není možné spustit kód přímo. Místo toho jej kompilátor analyzuje a převede do strojového kódu. Na rozdíl od jiných programovacích jazyků podporuje většinu běžných konceptů programovacích jazyků, jako jsou třídy, rozhraní a funkce. Jazyk Dart nepodporuje přímo pole. Podporuje kolekce sloužící k replikaci datových struktur, jako jsou pole, generika a volitelné typování.

3.4.1 Kde lze Dart používat?

Dart je univerzální jazyk a může se použít téměř na cokoli:

- Ve webových aplikacích můžete Dart používat pomocí knihovny dart: html a transpileru pro transformaci kódu Dart do jazyka JavaScript nebo pomocí frameworků, jako je AngularDart.
- Na serverech pomocí knihoven dart: http a dart: io. Lze použít také několik frameworků, například Aqueduct.
- V konzolových aplikacích.
- V mobilních aplikacích díky Flutteru.

3.5 Flutter vs React Native

Flutter a React Native jsou nejoblíbenějšími frameworky pro vývoj moderních multiplatformních mobilních aplikací. Technicky nejlepším přístupem k vytváření hladkých a dobře fungujících mobilních aplikací je bezpochyby stále vývoj založený na nativním rozhraní API pro konkrétní platformu. Pro vývojové týmy je však časově náročné a těžkopádné udržovat samostatné nativní kódové základny pro každý mobilní operační systém. Proto nyní každý vývojový tým volí pro tvorbu mobilních aplikací buď Flutter nebo React Native. Zde ale vyvstává otázka. Co je lepší? - Flutter nebo React Native.

Někteří vývojáři mohou volit Flutter kvůli jeho téměř nativnímu výkonu a bohaté sadě nástrojů pro widgety, zatímco jiní mají rádi React Native pro jeho přívětivost pro vývojáře a podporu widgetů pro konkrétní platformu. Aplikace Flutter vykazují oproti aplikacím React Native velké zlepšení výkonu díky bleskurychlé binární komunikaci Dart-to-Native. React Native komunikuje s nativními API prostřednictvím JavaScriptového můstku. Koncept JavaScriptového můstku nefunguje dobře pro všechny požadavky na vývoj. V důsledku toho začala komunita vývojářů React Native kritizovat, protože vykazoval slabý výkon na low-endových zařízeních a kvůli nadprůměrné spotřebě prostředků tak rychle vybíjel baterie. Nyní React Native řeší hlavní problémy s výkonem přepracováním svých základních modulů. V následujících podkapitolách bude rozebráno, jak nová architektura React Native konkuruje Flutteru.

3.5.1 Jak Flutter vyřešil problémy s výkonem lépe než React Native

Flutter je každým dnem populárnější ze dvou hlavních důvodů: výkon a flexibilita. Ovládací prvky uživatelského rozhraní specifické pro platformu jsou pro moderní případy použití a vlastní vylepšení použitelnosti omezené. Například implementace bočního menu v systému iOS je bez knihovny třetí strany obtížná, protože boční

menu není součástí směrnic UX společnosti Apple. Flutter nepoužívá nativní prvky uživatelského rozhraní specifické pro platformu, ale nabízí vlastní sadu nástrojů uživatelského rozhraní a je dodáván s vestavěnou knihovnou pro 2D vykreslování (Skia). React Native vykresluje nativní prvky uživatelského rozhraní specifické pro danou platformu na základě změn stavu komponent React. Kód JavaScriptu React Native běží uvnitř enginu JavaScriptu a kontext JavaScriptu komunikuje s nativním kódem (nativní hostitelskou aplikací) prostřednictvím mostu. Most JavaScriptu využívá asynchronní komunikaci založenou na protokolu JSON. Pokud nepoužíváte Hermes, bude běhové prostředí React Native interpretovat svazek JavaScript pomocí JavaScriptCore (JSC). Výsledkem je, že aplikace pracují pomalu a snižují použitelnost, pokud často používáte nativní rozhraní API a přidáváte další kód JavaScriptu. Flutter pro produkční aplikace nepoužívá ani JavaScriptový můstek, ani kompilaci JIT. Komunikace mezi Dart a nativními aplikacemi v enginu Flutter používá rychlejší binární paměťové buffery.

3.5.2 Porovnání React Native vs. Flutter

Když pomocí Flutteru vytvoříte multiplatformní aplikaci, stane se z ní aplikace Flutter, nikoli nativní aplikace pro konkrétní platformu. Aplikace Flutter přicházejí s odlišnými specifikacemi uživatelského rozhraní a poskytují koncovým uživatelům odlišný uživatelský zážitek. Na druhou stranu React Native nabízí způsob, jak používat nativní prvky uživatelského rozhraní, specifické pro danou platformu, prostřednictvím multiplatformního vývojářského rozhraní (React). Ve většině scénářů používáme multiplatformní frameworky, protože vývoj nativních aplikací je časově náročný - nikoli kvůli omezením sady nástrojů GUI specifické pro danou platformu. Flutter nám však poskytuje aplikace Flutter, nikoli nativní aplikace jako React Native.

Vývoj aplikací se stává jednodušším, když je k dispozici systém rozvržení pro více platforem. Proto se Electron stal volbou číslo jedna pro vytváření multiplatformních desktopových aplikací. Jako multiplatformní systém rozvržení používá jazyk HTML. Podobně i React Native poskytuje systém multiplatformního rozvržení založený na flexboxu, který slouží k vykreslování prvků uživatelského rozhraní specifických pro danou platformu. Na druhou stranu Flutter nemá systém rozvržení vhodný pro frontendové vývojáře a nabízí nové stromové rozvržení založené na widgetech, což někdy mate moderní vývojáře, kteří přišli z prostředí webového vývoje. Také díky podpoře JavaScriptu, podobnému rozhraní API prohlížeče Hermes a vývojovému prostředí poháněnému technologií React, je produktivita vývojářů React Native lepší než u Flutteru.

3.5.3 Jak nová architektura React Native řeší problémy s výkonem

Hlavní nevýhodou architektury React Native byl nedostatečný výkon na mobilních zařízeních střední a nižší třídy. Dříve se aplikace založené na React Native spouš-

těly pomalu, protože běhové prostředí JavaScriptCore při spuštění aplikace interpretovalo balíček JavaScript. Později tým React Native vytvořil Hermes a zrychlil dobu spouštění aplikace zavedením kompilace AOT (ahead-of-time) pro balíček JavaScript. Koncept AOP v Hermesu pomohl aplikacím React Native snížit spotřebu baterie tím, že snížil využití zdrojů. Díky těmto vylepšením výkonu bude React Native brzy používat Hermes jako výchozí engine JavaScriptu pro všechny nové projekty. JavaScriptový most však nefunguje dobře pro všechny případy vývoje, protože používá asynchronní komunikaci založenou na JSON serializací (serializaci) objektů JavaScript/native. React Native upgradoval svou současnou architekturu, aby tyto výkonnostní problémy vyřešil. Stávající modul bridge nyní nahrazuje nová komponenta nazvaná JSI (JavaScript Interface). JSI umožňuje kontextu JavaScriptu volat nativní metody přímo bez mostu tím, že v kontextu JavaScriptu drží nativní reference. Také umožňuje nativnímu kontextu React Native manipulovat přímo s objekty JavaScriptu. Komunikace událostí uživatelského rozhraní nové architektury (známá také jako Fabric) využívá JSI. Tato nová architektura nabízí rovněž nový koncept líného načítání nativních modulů prostřednictvím TurboModulů. TurboModuly dále zrychlí dobu spouštění aplikace. Nová architektura řeší téměř všechny výkonnostní problémy, se kterými se aplikace React Native v současnosti potýkají. Implementace nové architektury React Native zatím nebyla zveřejněna, ale je otevřená. Společnost Meta již architekturu otestovala na svých produkčních aplikacích.

3.5.4 Závěr porovnání

Neexistuje nic jako „nejlepší multiplatformní framework“ - každý framework je dobrý pro konkrétní případy použití. Dříve se však mnoho vývojářů shodlo na tom, že Flutter je díky svému výkonu a flexibilitě nejvýkonnějším multiplatformním frameworkem, jaký kdy byl vytvořen. Nová architektura React Native srovná výsledky konfrontace React Native i Flutteru. Nakonec můžeme porovnání React Native vs. Flutter považovat jen za porovnání JavaScript/React vs. Dart/MaterialUI. Možná jde jen o zkušenosti vývojářů. Nyní lze ignorovat faktor výkonu a zaměřit se pouze na to, který framework je vhodnější pro danou aplikaci. Jinými slovy, pokud je potřeba vlastní grafické uživatelské rozhraní, Flutter je správná volba. Pokud je potřeba nativní GUI, je správnou volbou React Native.

4 Andon TV

Cílem práce bylo vytvořit aplikaci pro průmyslovou automatizaci, která zobrazuje data v reálném čase z PLC. Aplikace byla primárně navržena pro televize se systémem Android, ale je možné ji spustit například i jako desktopovou aplikaci. Výběr systému Android nebyl náhodný, ale byl zvolen za účelem snadného nahrání aplikace na online obchod aplikací Google Play od společnosti Google a byl tak snadno přístupný. Motivací tohoto projektu bylo vytvoření uživatelsky přívětivé aplikace, která poběží na televizi nad jednotlivými linkami ve výrobních závodech a provozech, jež je schopná zobrazovat uživatelem definovaná data po různých komunikací. Tato data jsou zobrazována různými typy grafů, pomocí tabulky, ale je možné si nechat zobrazovat jen jeden určitý údaj. Aplikace také dokáže zobrazit až čtyři různé vizualizace zároveň. Uživatel si vybere požadované rozložení a následně si určí, který typ grafu nebo tabulky se má na televizi zobrazit.

4.1 Andon systém

Andon systém je aplikace, která informuje pracovníky firem, jako jsou operátoři výroby nebo management, o stavu výroby. Taková to aplikace dokáže nejen informovat, kolik bylo vyrobeno OK (odpovídají kritériím) a NOK (špatných/vadných) dílů, o aktuální teplotě nebo OEE (Overall Equipment Effectiveness - celková efektivita zařízení), umí ale také upozornit na vzniklé problémy, jako jsou technické poruchy nebo chybějící materiál.

4.2 pubspec.yaml

Každý Flutter projekt obsahuje soubor pubspec.yaml. Základní pubspec je vygenerován při vytváření nového projektu. Tento soubor se nachází na samotném vrcholu stromu projektu a obsahuje metadata o projektu, která Dart a nástroje Flutteru potřebují znát. Pubspec je napsán v jazyce YAML, který je pro člověka velmi dobře čitelný. Tento jazyk slouží pro serializaci dat textových souborů a slouží v případech, kdy aplikace potřebuje načíst nebo uložit strukturovaná data. Soubor pubspec definuje závislosti, které projekt vyžaduje. Můžou to být například knihovny a jejich verze, font písma nebo obrázky, které se v aplikaci nacházejí. Také specifikuje požadavky, jako jsou závislosti na vývojářských balíčcích nebo konkrétní omezení týkající se verze Flutter SDK. Veškeré knihovny, které lze do Flutter projektu při-

dat, jsou k dispozici na stránkách https://pub.dev. Na těchto stránkách jsou, krom návodů na instalaci knihoven a jejich popisů, také informace, na které platformy lze danou knihovnu využít.

4.3 AndroidManifest.xml

Jelikož je aplikace navržena pro televizi se systémem Android, je nutné, aby byl také správně definován soubor AndroidManifest.xml. Tento soubor musí být v každé aplikaci pro systém Android. Soubor manifest popisuje základní informace o aplikaci nástrojům pro sestavení systému Android, operačnímu systému Android a službě Google Play. V tomto souboru lze nalézt například oprávnění, které aplikace potřebuje pro přístup k chráněným částem systému nebo jiným aplikacím. Konkrétně v této aplikaci obsahuje AndroidManifest oprávnění k přístupu k internetu, aby se aplikace dokázala připojit k PLC systému a dala se ovládat přes HTTP klienta. Dalším oprávněním, které tento soubor obsahuje, je povolení čtení a zápisu z externího úložiště a také definování, že k běhu aplikace není vyžadována dotyková obrazovka. Tento soubor obsahuje spousty dalších příkazů, ale za zmínku stojí ještě například konfigurace aplikace. Tato konfigurace obsahuje parametry, které určují orientaci aplikace (to slouží primárně pro telefony, kde se určuje, jestli se aplikace spustí v režimu na výšku nebo šířku), či zda se při zapnutí zobrazí klávesnice nebo velikost okna.

4.4 Asynchronní programování

Asynchronní programování bylo navrženo tak, aby se vyrovnalo se zpožděním mezi voláním funkce a okamžikem, kdy je hodnota této funkce vrácena. Zatímco funkce čeká na návratovou hodnotu, umožňuje programu dokončit práci. Asynchronní programování umožňuje spustit více procesů, nechá je dělat jejich práci, a po jejím dokončení získá výsledek a zařadí ho do dalších kroků. Bez asynchronního programování by se aplikace dlouho načítaly, byly pomalejší, jelikož by se čekalo na výstupní hodnoty funkce a až pak by se dokončil zbytek programu. Používat tento způsob programování je běžné například při načítání dat po síti, zápisu do databáze nebo čtení dat ze souboru.

4.4.1 Asynchronní programování v programovacím jazyce Dart

K provádění asynchronních operací v programovacím jazyce Dart se využívá třídy Future a klíčových slov async a await. Instance třídy Future reprezentuje výsledek asynchronní operace, která může mít jeden ze dvou stavů: nedokončený nebo dokončený. Nedokončená operace znamená, že tato funkce stále čeká na dokončení operace nebo čeká na chybu. Naopak dokončená operace znamená, že byla úspěšně vrácena hodnota, nebo chyba. K definování asynchronní funkce je nutné napsat klíčové slovo async před tělo funkce. Zároveň je také nutné napsat klíčové slovo await, které slouží

k tomu, aby program věděl, že volaná funkce je asynchronní a bude se tudíž čekat na výsledek (viz obrázek 4.1).

```
Future<void> load() async {
  String jsonStringValues =
      await rootBundle.loadString('lib/lang/${locale.languageCode}.json');
  Map<String, dynamic> mappedJson = json.decode(jsonStringValues);
  _localizedValues =
      mappedJson.map((key, value) => MapEntry(key, value.toString()));
}
```

Obrázek 4.1: Ukázka asynchronní funkce v programovací jazyce Dart

4.5 Knihovna SharedPreferences

SharedPreferences se využívá k ukládání malého množství dat. V případě, že potřebujeme uložit několik málo hodnot, je zbytečné k tomu využívat databázi a ještě k tomu psát několik funkcí, které tuto databázi budou obsluhovat. Přesně z tohoto důvodu přichází na řadu již zmíněné SharedPreferences. Objekt SharedPreferences ukazuje na soubor obsahující dvojice klíč-hodnota a poskytuje jednoduché metody pro jejich čtení a zápis. Tato data existují i po vypnutí a opětovném spuštění aplikace, což je velmi užitečné. Toho lze využít například pro ukládání čísel, textu nebo polí a samozřejmě je možné tato uložená data načíst, třeba při spuštění aplikace. Díky tomu je možné načíst uživatelem uložené nastavení z předchozí relace bez použití databáze. Samozřejmostí je také možnost uložená data měnit nebo mazat dle potřeby. Objekt SharedPreferences ukládá, jak již bylo zmíněno, data ve dvojici klíč-hodnota a využívá k tomu výše zmíněné asynchronní programování (viz obrázek 4.2).

```
void saveValuesString(String key, String value) async {
  SharedPreferences _prefs = await SharedPreferences.getInstance();
  _prefs.setString(key, value);
}
```

Obrázek 4.2: Ukázka objektu SharedPreferences v programovacím jazyce Dart

4.6 Barevné téma aplikace

V současné době je přepínání barevných motivů jednou z nejoblíbenějších funkcí každé aplikace. Přepínáním motivů mohou uživatelé snížit namáhání očí, popřípadě

zvýšit čitelnost v různých světelných podmínkách. Dalším důvodem může být i obyčejná obliba jiné barvy, nežli je výchozí barva aplikace. Z těchto důvodů má i tato aplikace možnost dynamicky přepnout barevný motiv.

K dynamické změně barevného motivu bylo zapotřebí přidat dvě knihovny, Provider a výše zmíněnou knihovnu SharedPreferences. Dále bylo zapotřebí definovat jednotlivá barevná témata, konkrétně jsou čtyři. Každé barevné téma má tři parametry. Jas, který je u všech stejný a je nastaven na hodnotu světlý. Primární barva, kde je definována barva motivu. Jak bylo zmíněno, aplikace obsahuje čtyři barevné motivy, tudíž každý motiv má jinou primární barvu. Konkrétně specifickou zelenou (hexadecimální hodnota je #00985F), černou, červenou a modrou. Posledním parametrem je barevné schéma, které ovlivňuje například barvu písma. Z tohoto důvodu byla tato hodnota nastavena na bílou u všech čtyř barevných témat, jelikož bílé písmo je při použití těchto barev nejlépe čitelné.

Pro definování barevných tématu bylo nutné vytvořit třídu ThemeNotifier, která informuje aplikaci, že barevné téma bylo uživatelem změněno. Tato třída obsahuje getter, který načítá uloženou hodnotu z předchozí relace pomocí SharedPreferences a setter, který tuto hodnotu nastavuje v aplikaci. Setter zahrnuje také metodu notifyListener, která slouží pro nové sestavení grafického rozhraní, aby se požadovaná změna barevného tématu projevila. Ke správnému načtení barevného tématu z předchozí relace po startu aplikace, popřípadě načtení výchozí hodnoty, bylo nutné implementovat do souboru main.dart (spouští celou aplikaci) funkci, která pomocí knihovny Provider a třídy ThemeNotifier načte požadovanou hodnotu a nastaví barevné téma celé aplikace.

4.7 SetState

SetState je metoda, která frameworku Flutter oznámí, že se změnil vnitřní stav objektu State a znovu sestaví uživatelské rozhraní. Tato metoda se využívá, chceme-li změnit uživatelské rozhraní. Pro vysvětlení lze uvést dva příklady. Zaprvé, uživatel aplikace je požádán, aby vložil textový vstup, například svůj e-mail. Po stisknutí tlačítka se uživatelem zadaný e-mail zobrazí ve stejném okně aplikace v tabulce. Bez použití třídy SetState by se e-mail v tabulce objevil až ve chvíli, kdy by uživatel ručně aktualizoval stránku, popřípadě když by stránku opustil a opět se k ní vrátil. Při použití této metody se uživatelské rozhraní automaticky znovu sestaví a e-mail se objeví takřka okamžitě po stisknutí tlačítka. Jako druhý příklad lze uvést situaci, kdy jsou některé prvky aplikace neviditelné a objeví se jen v případě, že uživatel správně zadá textový vstup, nebo vybere požadované hodnoty z rozbalovacího menu. Jakmile jsou tyto volby správné, skryté prvky se objeví. Aby se tato metoda dala použít, je nutné, aby byla třída rozšířena o takzvaný StatefulWidget, který zaručuje možnost využití metody State a lze tudíž dynamicky aktualizovat grafické rozhraní (viz obrázek 4.3).

Obrázek 4.3: Ukázka StatefullWidgetu v programovacím jazyce Dart

4.8 Jazyk aplikace

Pro přepínání mezi jednotlivými jazykovými mutacemi byla vytvořena obecná třída, která obsahuje tři vlastnosti, a to identifikátor, jméno a kód jazyka. Všechny texty, které se v aplikaci nacházejí, jsou uloženy v JSON souborech ve formátu klíč - hodnota. Klíče textů jsou pro všechny jazyky stejné, liší se pouze v jejich hodnotách. Aby bylo možné vybrat a přepnout jazyk aplikace, bylo nutné napsat několik funkcí, které tuto činnost obsluhují. Po zvolení požadovaného jazyka je načten správný JSON soubor s požadovanou jazykovou mutací a její hodnoty jsou uloženy do objektu typu Map. Tato funkce by mohla zdržovat běh programu, a proto je asynchronní. Díky objektu Map je snadné vybrat požadovaný text dle klíče. Pro zvolení správné hodnoty byla vytvořena funkce, která vrací String hodnoty a jejím parametrem je požadovaný klíč. K jednoduššímu přístupu k těmto datům se využívá delegát, což je typově bezpečný ukazatel.

4.9 Main.dart

Tento soubor spouští celou aplikaci a nastavuje několik základních funkcí, jako je výše podrobněji popsaná změna barevného tématu nebo změna jazyka. V aplikaci jsou k dispozici čtyři různé jazykové mutace, čeština, angličtina, němčina a francouzština. Dále jsou k dispozici také čtyři barevné témata, černé, modré, červené a zelené. Tato dvě nastavení se načtou z předchozí relace a v případě, že se aplikace pouští poprvé, nastaví se výchozí hodnoty, neboli anglický jazyk a zelené téma. Tento soubor obsahuje dva společné prvky grafického rozhraní celé aplikace. Prvním z nich je název této aplikace s logem a druhým je menu. Menu zahrnuje dvě položky. První odkazuje k nastavení a druhý na připojení k PLC zařízení. Obě tyto sekce jsou podrobněji popsány níže.

4.10 Settings.dart

Tato třída slouží k nastavení prostředí celé aplikace. Na této stránce můžeme nalézt nastavení jazyka aplikace a změnu barevného tématu. Výchozím nastavením těchto hodnot je anglický jazyk a zelené téma. Všechny texty nacházející se v aplikaci jsou generovány pomocí funkce, která detekuje aktuálně zvolený jazyk a všechny texty přeloží do požadovaného jazyka (viz obrázek 4.4). To samé platí pro změnu barevného tématu. Pro všechny widgety, které na stránce nalezneme a u nichž lze definovat barvu, jako jsou tlačítka, výsuvné menu, pozadí a další, je místo statické

hodnoty použita funkce, která vrací požadovanou barvu (viz obrázek 4.5).

```
layoutColor(SettingsPage.selectedColor)
```

Obrázek 4.4: Funkce pro změnu barvy

```
getTranslated(context, 'settings_design')
```

Obrázek 4.5: Funkce pro vybrání správné jazykové mutace

Další nastavení, které zde můžeme nelézt se skrývají pod tlačítky + a -. Tyto tlačítka mění velikost fontu písma. Funkce byla přidána z předpokladu, že koncový zákazník nebude stát před televizí, na které aplikace poběží, ale bude nasazena ve výrobě. Díky tomuto předpokladu se tato funkce přidala, aby bylo možné v případě špatné čitelnosti textu nebo čísel, font zvětšit nebo zmenšit. Další informace k této funkcionalitě budou popsány v podkapitole 4.10.1.

Stránka obsahuje také tlačítko reset, které vrátí veškeré nastavení na výchozí hodnotu. V případě, že byly hodnoty uživatelem nastaveny na výchozí hodnoty, popřípadě nebyly změněny a jsou shodné s výchozími, je toto tlačítko neaktivní a nelze na něj tudíž kliknout.

Krom nastavení je zde také tlačítko, které po stisknutí zobrazí kontakty, v případě že jsou s aplikací nějaké potíže, nebo že by koncový zákazník požadoval přidat nějakou funkci, která se v aplikaci nenachází.

Poslední nastavení, které lze na této stránce nelézt je HTTP server. Popis této funkcionality je popsán v podkapitole 4.10.2.

Vzhledem k tomu, že se ve frameworku Flutter využívají widgety pro programování uživatelského rozhraní, bylo nutné správně definovat i proměnné, k předávání hodnot mezi všemi skripty. Jakmile uživatel přepne z jedné stránky na druhou, aplikace předchozí stránku odebere ze své paměti, jelikož není nutné widgety vykreslovat. Díky této vlastnosti byl během testování zjištěn problém, kdy aplikace spadla, vyhodila chybu, když stránka, ze které uživatel odcházel, měla nedokončené asynchronní funkce, popřípadě v ní běžel časovač. Chyby v programu způsoboval fakt, že předchozí stránka již neexistovala, jelikož se načetla stránka nová s definovanými widgety a jakmile tyto asynchronní funkce, popřípadě

časovače vracely hodnotu, popřípadě měnily vzhled aplikace (například upravovaly grafy), odkazovaly se na widgety, proměnné nebo funkce, které již neexistovaly.

Z tohoto důvodu bylo nutné přidat do třídy mixin třídu, která neaktivní třídu nepřepíše, ale nechá ji běžet na pozadí. Mixin je třída, která se využívá v objektově orientovaných programovacích jazycích a obsahuje metody z několika dalších tříd. Díky tomuto rozšíření aplikace ví, že daná třída zůstane na pozadí, i když uživatel stránku opustí (viz obrázek 4.6).

```
class _ConnectionsPageState extends State<ConnectionsPage>
    with AutomaticKeepAliveClientMixin<ConnectionsPage> {
  @override
  bool get wantKeepAlive => true;
```

Obrázek 4.6: Ukázka mixin třídy v programovacím jazyce Dart

4.10.1 Změna velikosti fontu písma

Jelikož text a čísla se objevují na každé stránce aplikace, neboli v různých třídách, bylo zapotřebí vytvořit třídu novou, která obsahuje globální funkce, ke kterým má přístup celá aplikace. Globální funkce pro změnu velikosti fontu byly nezbytné, jelikož v opačném případě by takovéto funkce musely být v každé třídě duplicitně, a to by z programátorského hlediska nedávalo smysl.

Původní návrh, který byl vytvořen, měnil font písma na všech stránkách stejně, takže velikost fontu byla stejná na všech stránkách. A nejenom na každé jednotlivé stránce, ale také ve vyskakovacích oknech, která neslouží jen pro nastavení různých parametrů aplikace. Při zvětšování fontu písma se nemění pouze velikost písma, ale také velikost widgetů, které text obsahují. Jako příklad lze zmínit tlačítka, které obsahují text (popis), aby uživatel věděl, k čemu slouží. Jakmile se font písma zvětšil, zvětšila se tím i velikost widgetu. Tento návrh ale časem vedl k problémům, jelikož každá stránka obsahuje jiný počet elementů, které jsou jinak uspořádány. Konkrétně docházelo k takzvanému přetečení, neboli k tomu, že text, popřípadě widgety, zasahoval (vykresloval se) mimo okno aplikace. Jakmile tato situace nastala, aplikace vyhodila chybu. Jak bylo zmíněno, vzhledem k různému počtu a rozložení elementu na stránce, některé části aplikace se vykreslovaly normálně a vypadalo to, že vše funguje jak má, na jiných stránkách však k tomuto přetečení docházelo.

První nápad byl omezit velikost fontu globálně tak, že se hledalo největší možné písmo, které nepůsobilo problém přetečení na žádné stránce či vyskakovacím okně. To se ale později ukázalo jako nepříliš dobré řešení. Vedlo to k tomu, že některé

části aplikace bylo možno přečíst z dálky velmi dobře, ale jiné už nikoli.

Z tohoto důvodu přišlo finální řešení, které zohledňuje každou stránku a vyskakovací okno zvlášť, neboli každá část aplikace obsahuje své vlastní proměnné. Proměnné obsahují informace o velikosti fontu písma pro nadpisy, podnadpisy a normální text. Každá stránka nebo vyskakovací okno má různé limity pro velikost fontu písma. Zároveň však bylo třeba, aby maximální počet zvětšení (kliknutí na tlačítko se symbolem plus) a zmenšení (kliknutí na tlačítko se symbolem minus) fontu písma byl všude stejný (viz obrázek 4.7). Konkrétně bylo naprogramováno šest různých úrovní velikosti fontu. Díky tomu se na každé stránce při zvětšení nebo zmenšení fontu písma přidává nebo ubírá jiná hodnota tak, aby v žádné části aplikace nedocházelo k výše zmíněnému přetečení. Globální funkce, které se starají o změnu velikosti fontu písma, obsahují nejen podmínky pro hlídání maximálních a minimálních hodnot, jakých může velikost fontu nabývat, ale také volají funkce obsahující objekt SharedPreferences, zmíněné v podkapitole 4.5, které se starají o to, aby se tyto hodnoty uložily do paměti. Díky tomu pak uživatel nemusí při každém spuštění tyto hodnoty nastavovat.

Obrázek 4.7: Tlačítka pro zvětšení a zmenšení velikosti fontu písma

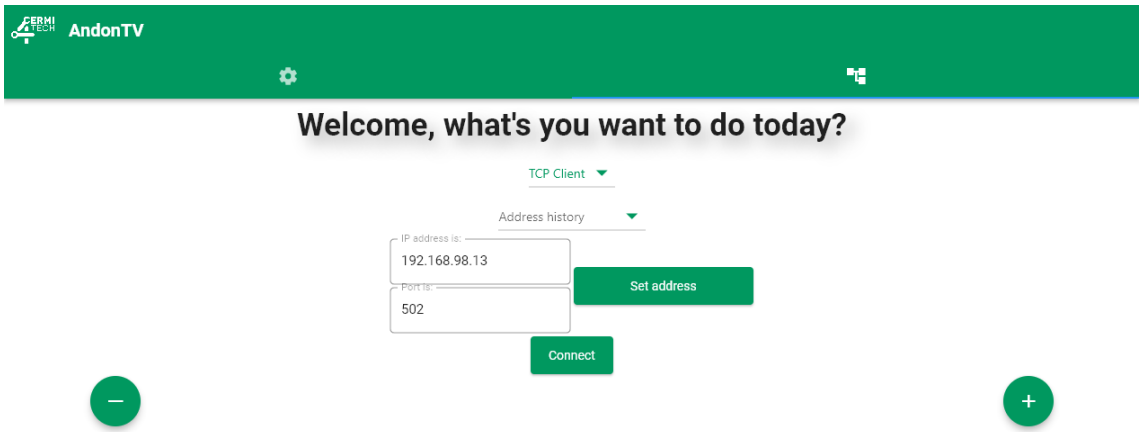
4.10.2 HTTP server

Andon systém slouží k tomu, aby informoval o stavu výroby v reálném čase. Vzhledem k tomu, že systém vyvíjený v této práci bude běžet na televizi umístěné nad výrobními linkami, lze předpokládat, že nebude volně k dispozici dálkový ovladač, aby nekompetentní osoby nemohli tuto televizi ovládat, například ji přepnout jinam, nastavit jiné parametry v aplikaci, popřípadě ji úplně vypnout. Kvůli tomu do aplikace přidán HTTP server, díky němuž lze ovládat aplikaci vzdáleně. Hlavní parametry HTTP serveru, které je třeba nastavit, jsou IP adresa a port. Vzhledem k tomu, že HTTP server běží přímo v televizi, bude se ve většině případů nastavovat IP adresa dané televize. Aby aplikace byla uživatelsky přívětivější, byla přidána funkce, která vypíše IP adresu televize a nechá uživatele rozhodnout, zda tuto IP adresu chce použít, či nikoli. Funkce byla přidána proto, aby uživatel nemusel hledat v nastavení televize, respektive sítě, jakou IP adresu daná televize má. V případě, že se uživatel rozhodne tuto IP adresu použít, stačí následně zadat libovolný port a potvrdit volbu. V případě, že se uživatel rozhodne použít z určitých důvodů jinou IP adresu, musí kromě portu zadat také tuto jím požadovanou IP adresu. Při potvrzení těchto dvou zadaných parametrů, aplikace zkontroluje pomocí regulárního výrazu, zda zadaná IP adresa a port jsou validní. V případě že ne, aplikace na to uživatele upozorní pomocí stavové hlášky ve spodní části obrazovky a napíše mu,

která část adresy je zadána špatně. V opačném případě je uživateli povoleno HTTP server spustit. Jakmile uživatel klikne na tlačítko, kterým se HTTP server zapíná, aplikace se pokusí HTTP server vytvořit dle uživatelem zadaných parametrů. V případě, že aplikace tento server nedokáže vytvořit, například IP adresa je validní, ale není ji možno použít, informuje o tom uživatele pomocí stavové hlášky a také ho informuje, že je nutné zadané parametry upravit. V opačném případě aplikace vytvoří HTTP server a vypíše na obrazovku informace o tom, jaká IP adresa a port je použit a také, že tento server je úspěšně vytvořen a běží.

4.11 Connections.dart

Tato stránka aplikace obsluhuje připojení k PLC zařízení pomocí protokolů TCP a Modus TCP. Vzhledem k této skutečnosti se tato stránka zobrazí jako první, jakmile uživatel tuto aplikaci zapne (viz obrázek 4.8). Aby se aplikace dokázala připojit k PLC, je nutné znát IP adresu a port zařízení.

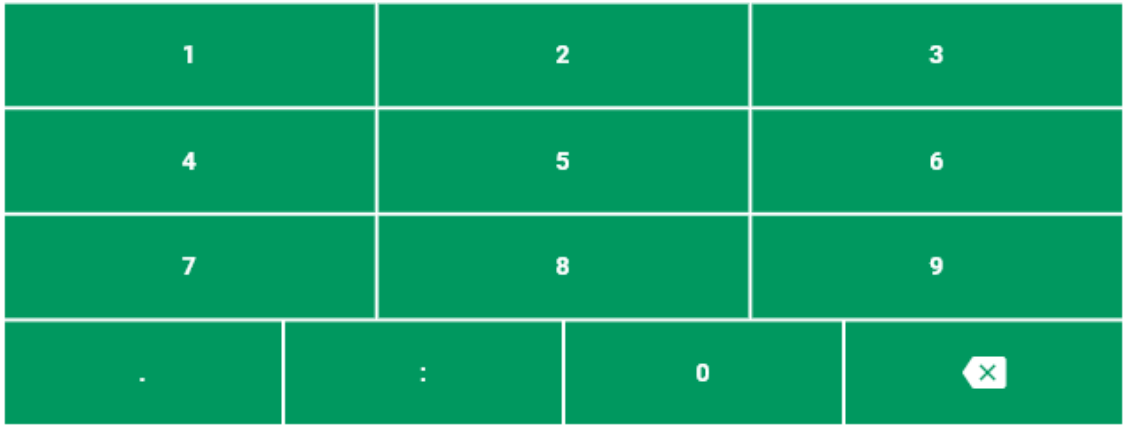


Obrázek 4.8: Úvodní obrazovka aplikace

Framework Flutter obsahuje widget pro uživatelský textový nebo číselný vstup. První kroky proto vedly k využití tohoto widgetu, do něhož uživatel zadá IP adresu PLC a jeho port. Zde ale vznikl problém. Bylo zjištěno, že televize, v níž aplikace běží, zamrzne, jakmile se tento widget zaktivuje. Aktivací widgetu je myšleno to, že uživatel na něj klikne nebo se na něj přesune pomocí šipek na ovladači. Jelikož bylo nezbytné tuto situaci vyřešit, bylo nutné přistoupit k jinému řešení. Tím bylo vytvoření vlastní číselné klávesnice, která tento widget pro textový a číselný vstup nahradila.

4.11.1 Klávesnice

V řešení byla zformována třída, která vytváří číselnou klávesnici (viz obrázek 4.9). Klávesnice se skládá ze čtyř řádků a tří sloupců. V prvních třech řádcích nalezneme čísla od jedné do devíti. Čtvrtý a zároveň poslední řádek klávesnice obsahuje číslo nula, znak tečky, dvojtečky a také klávesu, která obsluhuje mazání posledního znaku. K obsluze klávesnice byly vytvořené dvě přidružené třídy, které definují vzhled jednotlivých tlačítek. První zahrnuje vzhled tlačítka pro mazání a druhá definuje vzhled všech ostatních kláves, jelikož vypadají stejně, až na obsažený znak. Obalová třída také vytváří dvě metody, která zaručují smazání posledního znaku v textovém řetězci a vkládání vybraného znaku na konec řetězce.



Obrázek 4.9: Klávesnice

4.11.2 Vyskakovací okno

V případě, že není možné použít textový widget, který otevírá nativní klávesnici zařízení a je nutné využít vlastní klávesnici, bylo nezbytné vytvořit další stránku, kde bude uživatel moci vložit požadovanou adresu (IP adresa a port) PLC zařízení. Aby aplikace nemusela přepínat mezi více stránkami, bylo vytvořeno vyskakovací okno, kam uživatel zadá požadovanou adresu (viz obrázek 4.11). Okno se otevře, jakmile uživatel klikne na příslušné tlačítko v aplikaci. Okno obsahuje, krom výše zmíněné klávesnice, také místo, kde se zobrazují informace o zadané adrese, a dvě tlačítka pro potvrzení adresy a zrušení akce.

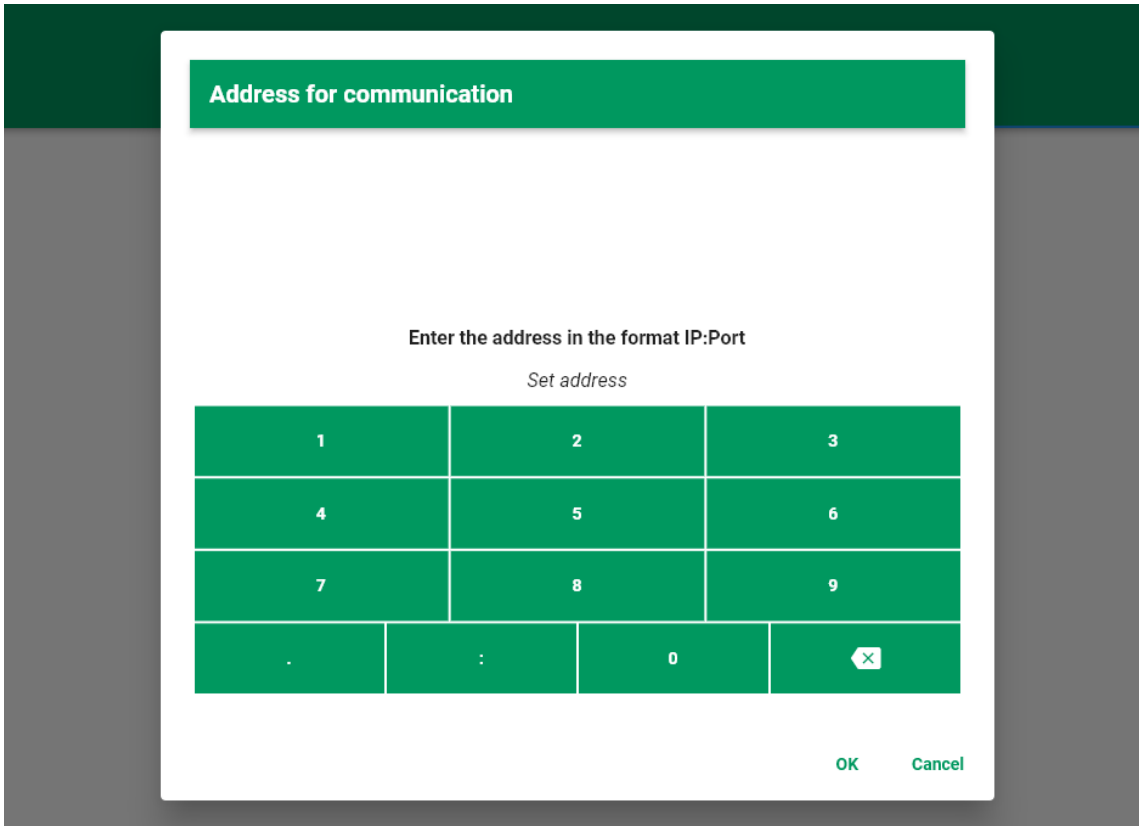
V této části aplikace informuje uživatele, jaký formát adresy je požadován a také, jaké hodnoty již uživatel napsal. Formát IP adresy je následující: X.X.X.X, kde X zastupuje jedno až třímístné číslo. Aby tento vstup byl uživatelsky přívětivější, byla vytvořena funkce, která automaticky vkládá na požadované místo tečku. Vzhledem k tomu, že číslo X může obsahovat jedno až třímístné číslo, tečka se automaticky vkládá jen v případě, že zadané číslo je trojmístné. Například v situaci, kdy uživatel zadá IP adresu 192.168.72.15, je tečka automaticky vložena pouze za první dvě

čísla. Následně je nezbytné, aby uživatel definoval port, který se nachází ihned za IP adresou, a rozděluje je dvojtečka.

Vyskakovací okno pro zadání adresy také obsahuje funkci, která zadanou adresu zkontroluje a v případě chybného uživatelského vstupu informuje uživatele, která část adresy je chybná, a nechá uživatele tuto chybu opravit. Validace zadané adresy zajišťuje regulární výraz, který kontroluje, zda zadaná adresa je validní (viz obrázek 4.10). V případě, že adresa validní není, zobrazí se ve spodní části na několik vteřin stavová zpráva, která informuje uživatele, kterou část adresy definoval špatně - IP adresu, port, nebo obojí. Tato validace se spustí v okamžiku, kdy uživatel klikne na potvrzovací tlačítko. V případě, že vstup je chybný, okno zůstane otevřené a zobrazí se výše zmíněná chybová hláška. V opačném případě se okno zavře, daná adresa se vyplní do řádků pro tuto adresu a také je vložena do historie adres. Když uživatel klikne na tlačítko zrušit, okno se zavře a veškeré změny se zahodí.

```
static RegExp regExpIP = new RegExp(
    r"^(?=\d+\.\d+\.\d+\.\d+$)(?:(?:25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9][0-9]|[0-9])\.?){4}$");
```

Obrázek 4.10: Nastavení adresy PLC zařízení



Obrázek 4.11: Nastavení adresy PLC zařízení

4.12 Komunikace

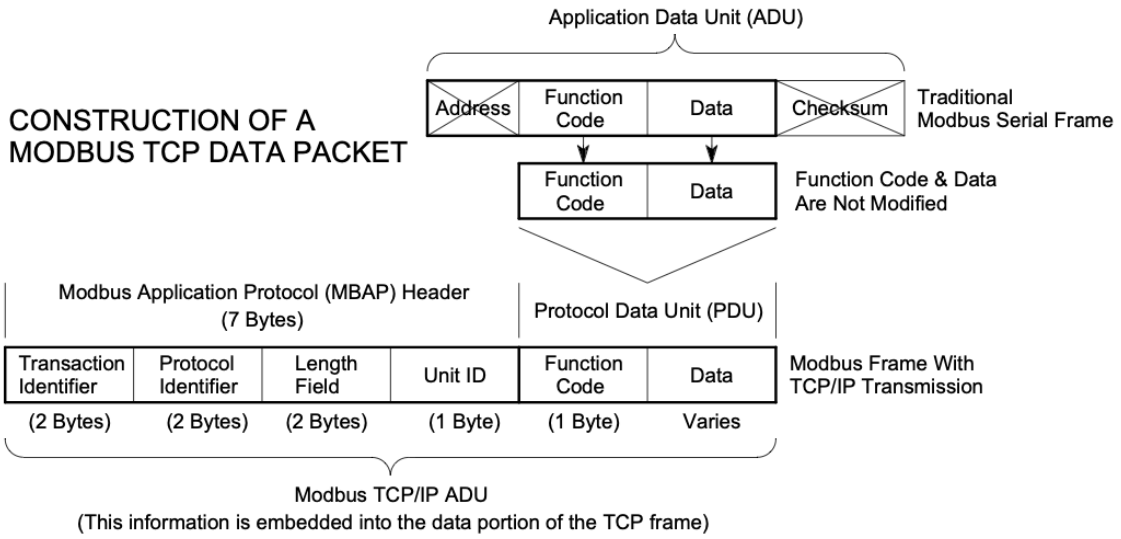
4.12.1 TCP/IP

TCP/IP je zkratka pro Transmission Control Protocol/Internet Protocol a jedná se o sadu komunikačních protokolů používaných k propojení síťových zařízení na internetu. Protokol TCP/IP se používá také jako komunikační protokol v soukromé počítačové síti (intranetu nebo extranetu). Celá sada protokolů IP - sada pravidel a postupů - se běžně označuje jako TCP/IP. Protokoly TCP a IP jsou dva hlavní protokoly, ale do sady jsou zahrnuty i další. Sada protokolů TCP/IP funguje jako abstraktní vrstva mezi internetovými aplikacemi a směrovací a přepínací strukturou. Protokol TCP/IP využívá komunikační model klient-server, kdy je uživateli nebo počítači (klientovi) poskytována služba, například odeslání webové stránky, jiným počítačem (serverem) v síti. Souhrnně je sada protokolů TCP/IP klasifikována jako bezstavová, což znamená, že každý požadavek klienta je považován za nový, protože nesouvisí s předchozími požadavky. Bezstavovost uvolňuje síťové cesty, takže je lze používat nepřetržitě. Samotná transportní vrstva je však stavová. Přenáší jedinou zprávu a její spojení zůstává zachováno, dokud nejsou všechny pakety ve zprávě přijaty a znovu sestaveny v cíli. Model TCP/IP se mírně liší od sedmivrstvého síťového modelu Open Systems Interconnection (OSI) navrženého podle něj. Referenční model OSI definuje, jak mohou aplikace komunikovat v síti.

4.12.2 Modbus TCP

Modbus TCP (také Modbus-TCP) je jednoduše protokol Modbus RTU s rozhraním TCP, který běží na síti Ethernet. Struktura zpráv Modbus je aplikační protokol, který definuje pravidla pro uspořádání a interpretaci dat nezávisle na přenosovém médiu. TCP/IP označuje Transmission Control Protocol a Internet Protocol, který zajišťuje přenosové médium pro zprávy Modbus TCP. Zjednodušeně řečeno, TCP/IP umožňuje výměnu bloků binárních dat mezi počítači. Je to také celosvětový standard, který slouží jako základ pro World Wide Web. Hlavní funkcí protokolu TCP je zajistit, aby byly všechny datové pakety správně přijaty, zatímco protokol IP zajišťuje správné adresování a směrování zpráv. Všimněte si, že kombinace TCP/IP je pouze transportní protokol a nedefinuje, co data znamenají nebo jak mají být interpretována (to je úkolem aplikačního protokolu, v tomto případě Modbusu). Takže shrnuto, Modbus TCP používá TCP/IP a Ethernet k přenosu dat struktury zpráv Modbus mezi kompatibilními zařízeními. To znamená, že Modbus TCP kombinuje fyzickou síť (Ethernet) se síťovým standardem (TCP/IP) a standardní metodou reprezentace dat (Modbus jako aplikační protokol). Zpráva Modbus TCP je v podstatě jednoduše komunikace Modbus zapouzdřená v obalu Ethernet TCP/IP. V praxi Modbus TCP vkládá standardní datový rámec Modbus do rámce TCP bez kontrolního součtu Modbus, jak ukazuje následující schéma na obrázku 4.12.

Příkazy Modbus a uživatelská data jsou zapouzdřeny do datového kontejneru telegramu TCP/IP, aniž by byly jakkoli upravovány. Pole pro kontrolu chyb



Obrázek 4.12: Schéma komunikace Modbus TCP/IP

Modbusu (kontrolní součet) se však nepoužívá, protože k zajištění integrity dat se místo toho používají standardní metody kontrolního součtu linkové vrstvy Ethernet TCP/IP. Dále je pole adresy rámce Modbus nahrazeno identifikátorem jednotky v protokolu Modbus TCP a stává se součástí hlavičky aplikačního protokolu Modbus (MBAP).

Z obrázku je patrné, že pole kódu funkce a dat jsou absorbována v původní podobě. Aplikační datová jednotka (ADU) protokolu Modbus TCP má tedy podobu 7bajtové hlavičky (identifikátor transakce + identifikátor protokolu + pole délky + identifikátor jednotky) a datové jednotky protokolu (kód funkce + data). Záhlaví MBAP má 7 bajtů a obsahuje následující pole:

- Identifikátor transakce/vyvolání (dva bajty): Toto identifikační pole se používá pro párování transakcí, když klient posílá více zpráv podél stejného spojení TCP, aniž by čekal na předchozí odpověď.
- Identifikátor protokolu (dva bajty): Toto pole má pro služby Modbus vždy hodnotu 0 a další hodnoty jsou vyhrazeny pro budoucí rozšíření.
- Délka (dva bajty): Toto pole je počtem bajtů zbývajících polí a zahrnuje bajt identifikátoru jednotky, bajt kódu funkce a datová pole.
- Identifikátor jednotky (jeden bajt): Toto pole se používá k identifikaci vzdáleného serveru umístěného v jiné síti než TCP/IP (pro sériové přemostění). V typické aplikaci serveru Modbus TCP je identifikátor jednotky nastaven na 00 nebo FF, server jej ignoruje a v odpovědi se jednoduše opakuje.

Kompletní aplikační datová jednotka Modbus TCP je vložena do datového pole standardního rámce TCP a odeslána prostřednictvím TCP na známý systémový

port 502, který je vyhrazen speciálně pro aplikace Modbus. Klienti a servery Modbus TCP naslouchají a přijímají data Modbus přes port 502.

4.12.3 Komunikace po TCP/IP

V případě, že si uživatel vybere tento typ komunikace, aplikace otevře novou stránku s animací načítání, které informuje uživatele, že se aplikace snaží připojit k PLC zařízení pomocí zadané IP adresy a portu. V případě neúspěšného navázání spojení se aplikace vrátí na hlavní stránku a informuje uživatele, že spojení nebylo navázáno. V opačném případě, tedy při úspěšném navázaní spojení, začíná aplikace „poslouchat“ a přijímá data. Zároveň se zapíná i časovač, který hlídá, zda data jsou neustále přijímána. Tato funkce je více rozebrána v podkapitole 4.13.1.

Data jsou přijímána ve formátu JSON, tudíž je zapotřebí tato data dekódovat a uložit je. Dekódovaná data jsou uložena do objektu typu Map. Tento objekt je velmi podobný formátu JSON, jelikož obsahuje páry klíčů a hodnot. Funkce aplikace, která obstarává zpracování přijímaných dat, obsahuje také inicializaci potřebných proměnných ke správnému fungování aplikace. Tato inicializace se provede při prvním přijmutí dat. Inicializace vynuluje všechna předešlá data z předchozího připojení a uloží si jednotlivé klíče datové struktury. Tyto klíče následně slouží k identifikaci jednotlivých hodnot, podle kterých je umožněno uživateli vybrat potřebná data a zobrazit je na televizi.

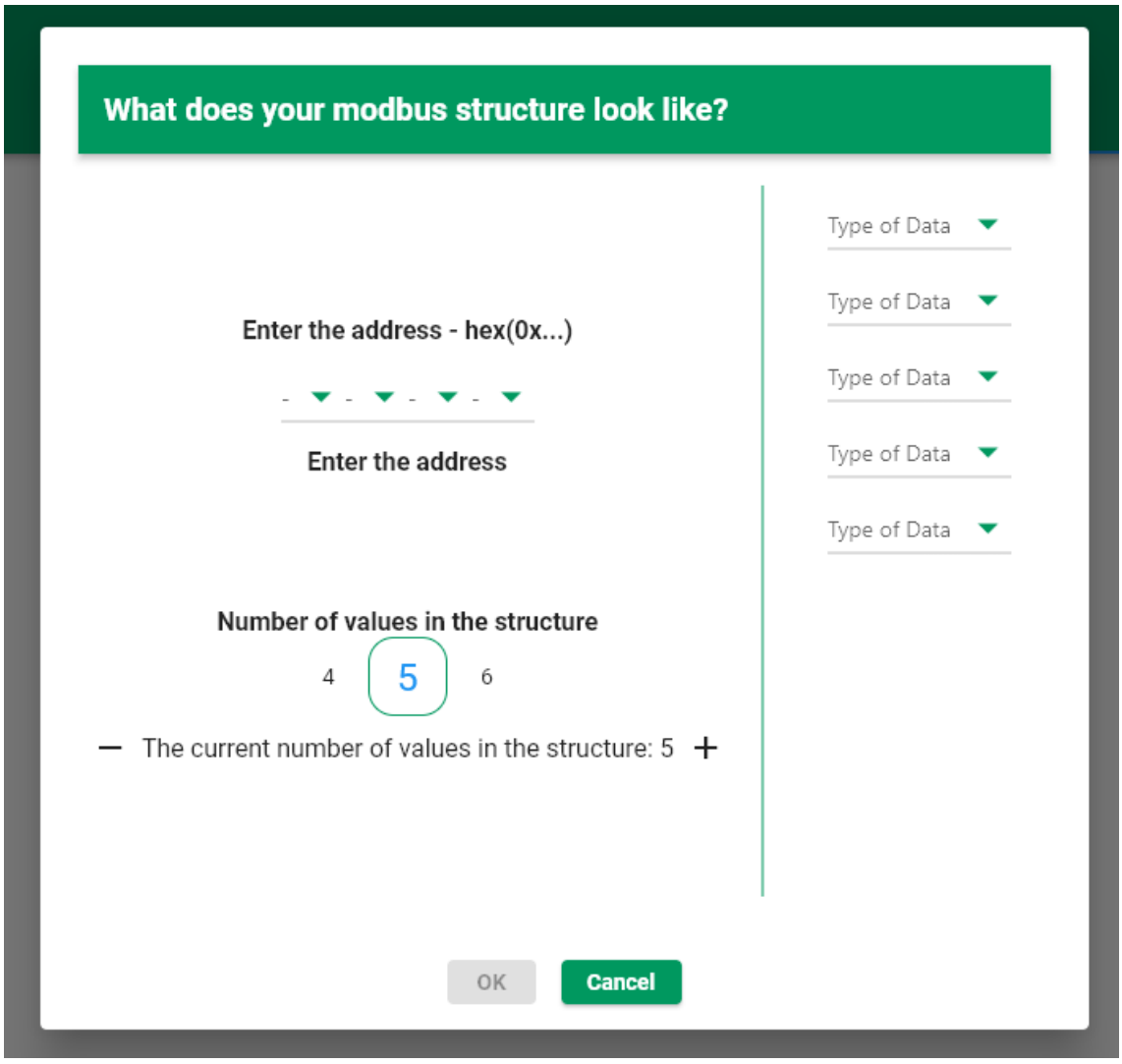
Jelikož ne každý graf se může skládat ze všech datových typů, například sloupcový graf na ose Y musí obsahovat pouze číselné hodnoty, je nutné, aby aplikace roztřídila přijímaná data dle jejich datových typů. Data se třídí na dvě skupiny. Klíče, jejichž hodnoty obsahují čísla, a klíče, jejichž hodnoty obsahují vše ostatní.

Po inicializaci si aplikace začne ukládat jednotlivá přijímaná data. Ukládání dat se taktéž rozlišuje do dvou skupin. První skupina ukládá všechna přijímaná data. Tato skupina slouží k následnému zobrazení tabulky, která všechna tato data obsahuje. Do druhé skupiny se ukládají hodnoty definující jednotlivé grafy. Data, která grafy definují, se začnou ukládat, jakmile uživatel definuje, jaké hodnoty mají být na osách X a Y. Do té doby se ukládání přeskakuje.

Vzhledem k tomu že funkce, která přijímání dat obsluhuje, běží v nekonečné smyčce, do doby, než se uživatel odhlásí, bylo nutné tuto funkci vytvořit jako asynchronní, aby aplikace nečekala, až se funkce dokončí. Bez definování asynchronní funkce by aplikace zamrzla a nebylo by možné ji dále používat. Definování grafů, rozložení a dalších funkcí, které se zobrazování dat týkají, bude popsáno v podkapitole 4.13.

4.12.4 Komunikace po Modbus TCP/IP

Pokud si uživatel vybere komunikaci Modbus TCP, na stránce se zobrazí tlačítko, které otevírá vyskakovací okno pro nastavení parametrů požadovaných ke správnému fungování této komunikace. Tlačítko pro připojení k PLC zařízení je neaktivní (nelze na něj kliknout) do doby, než jsou všechny parametry komunikace vyplněny. Vyskakovací okno slouží pro nastavení parametrů komunikace Modbus TCP (viz obrázek 4.13).



Obrázek 4.13: Nastavení parametrů pro komunikaci Modbus TCP/IP [8]

Prvním parametrem je adresa, ze které začne aplikace číst. Tato adresa se zadává v hexadecimální soustavě. Dalším nastavovaným parametrem je struktura přijímaných dat. Vzhledem k tomu, že Modbus přijímá a odesílá pouze data datového typu Word, který má šestnáct bitů, neboli dva bajty, je nezbytné, aby

uživatel specifikoval datovou strukturu. K definování této struktury slouží dva prvky. Prvním z nich je definování počtu přijímaných dat. K tomu slouží karusel s čísly, kde si uživatel zvolí celkový počet dat. Druhý prvek, definuje samotnou strukturu. Zde si uživatel zvolí datový typ každého jednoho členu této struktury. Datové typy mohou být Byte, Integer, Real a další. Strukturu si uživatel definuje ze dvou hlavních důvodů. Tím první je fakt, že se tak aplikace stává univerzálnější vůči různým strukturám dat. Druhý důvod již zde byl naznačen. Modbus posílá a přijímá datový typ typu Word, neboli šestnáct bitů.

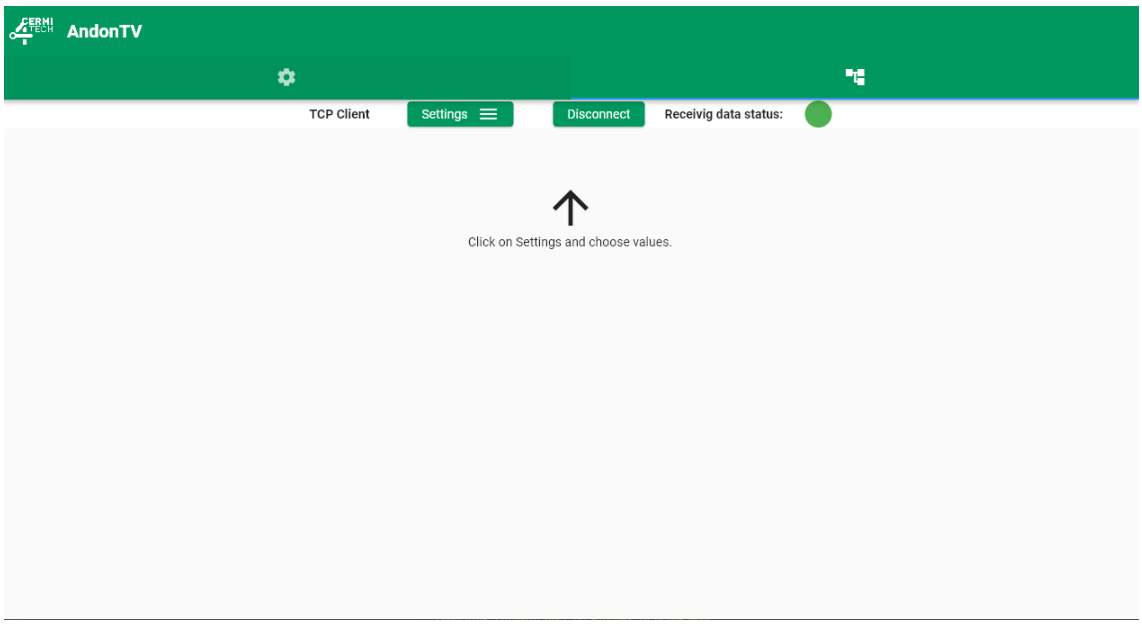
Vzhledem k této skutečnosti vyvstává otázka, jak aplikace dokáže přijmout například datový typ, který má třicet dva bity, jako je například datový typ Real nebo DInt, nebo osmibitový datový typ, jakým je například Byte. Pokud jde o osmibitový datový typ, je to jednoduché, jelikož při konverzi na šestnáctibitovou hodnotu se tato hodnota pouze doplní nulami na významnější bity, aby se nezměnila posílaná hodnota. Díky tomu je pak snadné přijmout a přečíst správnou hodnotu, kterou PLC zařízení posílá. V případě, že PLC zařízení posílá hodnoty o velikosti třicet dva bity, je nutné, aby aplikace načetla dva Wordy a ty správně zpracovala. Funkce, která obsluhuje tuto konverzi, využívá metodu setUint16. Tato metoda převede dva bajty s daným ofsetem na binární reprezentaci čísla bez znaménka. Metoda je využita na obě Wordové hodnoty, kde první hodnota má ofset nastavený na nulu a druhá na číslo dvě. Následně se tyto hodnoty uloží do čtyřbajtové proměnné, která se pomocí metody getFloat32 převede na reálnou hodnotu, kterou PLC systém do televize poslal.

Jakmile jsou veškerá potřebná data pro komunikaci přes Modbus vyplněna, je možné kliknout na tlačítko připojit, díky kterému se aplikace pokusí navázat spojení. V případě, že připojení k PLC systému selže, aplikace se vrátí na domovskou stránku a informuje uživatele o vzniklé chybě. V opačném případě, kdy je spojení navázáno úspěšně, aplikace začne načítat přijímaná data. Ke čtení dat se používá funkce, která na uživatelem zadané adrese začne číst data z registrů. Ke čtení z registrů je kromě počáteční adresy nutná také délka celé struktury. Tuto délku si aplikace jednoduše zjistí z uživatelsky definované struktury v aplikaci. Jakmile aplikace data přijme, začne si je ukládat do připravených proměnných. Jak už bylo zmíněno, je nutné, aby aplikace rozlišila, kolik bajtů má každá jednotlivá hodnota a výslednou hodnotu správně uložila, jak bylo zmíněno výše. Při prvním obdržení hodnot, jak tomu bylo i u komunikace TCP/IP, se provede inicializace. Aplikace vymaže veškeré hodnoty (hodnoty, které definují grafy/tabulky) uložené z předchozí relace a z nově příchozích si uloží jejich klíče. Aplikace opět rozlišuje, zda hodnoty, které spadají pod daný klíč, jsou čísla či nikoli a podle toho rozdělí klíče do dvou skupin.

Funkce, která obsluhuje přijímání dat je taktéž asynchronní, aby aplikace mohla vykonávat i ostatní běžící úlohy a nezastavila se v nekonečné smyčce. Funkce si ihned začne ukládat přijímaná data do proměnné, aby bylo možné následně zobrazit data v tabulce. Proměnné, které definují grafy, zůstávají prázdné do doby, než si uživatel v aplikace zvolí, která data chce v grafech zobrazit.

4.13 Dashboard

Stránka aplikace, kde se zobrazují přijímané informace z PLC zařízení, se nazývá dashboard. Na tuto stránku se uživatel dostane jen v případě, že se aplikace připojí k PLC a přijme data. Jakmile jsou obě tyto podmínky splněny, zobrazí se úvodní obrazovka (viz obrázek 4.14).



Obrázek 4.14: Úvodní obrazovka dashboardu

V horní části stránky se nacházejí tři hlavní prvky. Prvním z nich je tlačítko, které po kliknutí zobrazí nastavení přijímaných dat. Druhým prvkem je taktéž tlačítko, které slouží pro odhlášení ze zvoleného protokolu. V případě, že uživatel na toto tlačítko klikne, aplikace nastaví všechny proměnné, které jsou spjaty s touto stránkou, protokoly a přijatými daty, na výchozí hodnotu. Zároveň se zobrazí úvodní stránka aplikace, aby si uživatel mohl opět zvolit protokol a připojit se k PLC. Posledním prvkem horní části stránky aplikace je stavový řádek s diodou. Tento stavový řádek informuje uživatele, zda aplikace stále přijímá data. Dioda má tři stavy, respektive může svítit třemi barvami. Zelená barva znázorňuje, že aplikace data přijímá. Oranžová barva informuje uživatele, že po definovaný čas aplikace nepřijala žádná data a uživatel by měl věnovat pozornost PLC zařízení, zda data do televize opravdu posílá. Poslední barva je červená; ta se zobrazí v případě, že aplikace data stále nepřijímá. Červená dioda se zobrazuje pouze na deset vteřin, jelikož pak se aplikace automaticky odhlásí a zobrazí se úvodní obrazovka. Stavový řádek bude více popsán v následující podkapitole. Krom horního řádku s výše popsanými prvky, se na stránce taktéž objeví text se šipkou na tlačítko nastavení, který informuje uživatele, že pro zobrazení přijímaných dat je nutné nastavit, jak se data mají zobrazovat. Nastavení zobrazení bude popsáno 4.13.2.

4.13.1 Stavový řádek

Jelikož aplikace zobrazuje data v reálném čase a informuje uživatele o stavu výroby, je vhodné, aby aplikace uživatele také informovala o tom, zda je aplikace k PLC systému stále připojena a zda stále přijímá data. O tuto funkcionalitu se starají dva, v podstatě tři časovače.

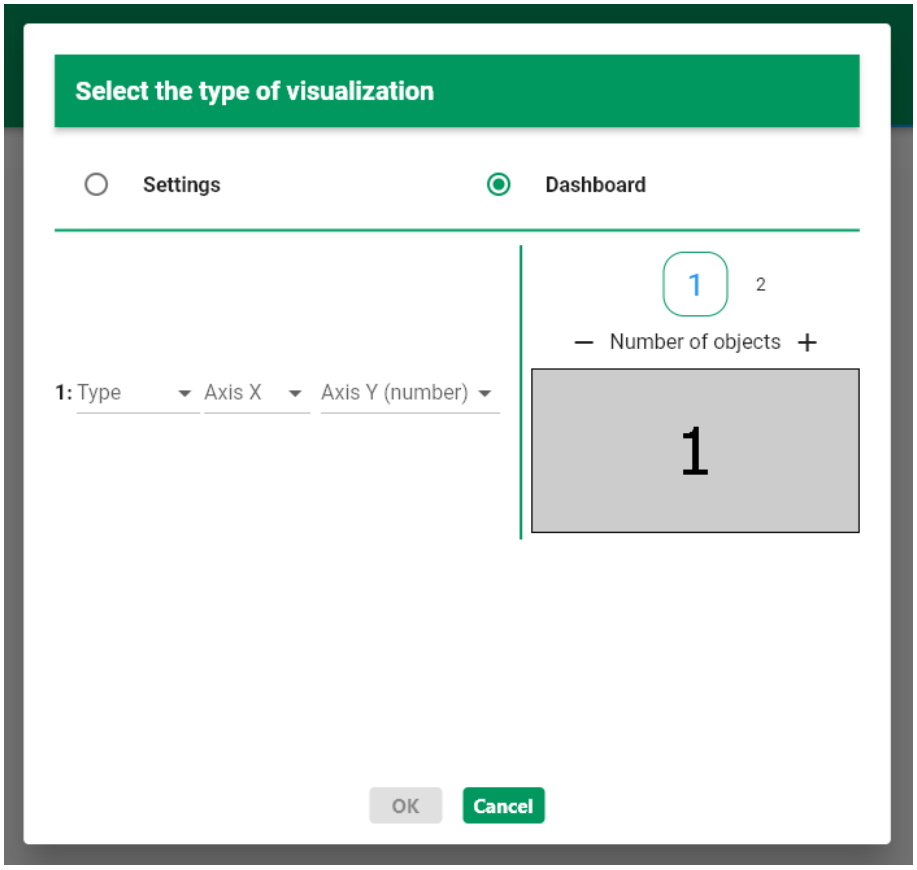
První časovač se spustí, jakmile aplikace přijme data a začne odpočítávat čas uběhlý od posledního přijetí dat. Vzhledem k tomu, že je to časovač a ne stopky, bylo nutné definovat dobu, za kterou časovač skončí. Aplikace má výchozí dobu časovače nastavenou na jednu minutu, ale tuto dobu může uživatel přenastavit. V případě, že aplikace přijme další data a časovač neskončil svou činnost, je časovač zrušen a opět nastaven a spuštěn s počáteční hodnotou. Jakmile časovač neskončí svou činnost samovolně, neboli neuplyne nastavená doba, stavový řádek svítí zeleně a tím dává uživateli najevo, že data jsou aplikací přijímána.

V momentě, kdy v prvním časovači uplyne nastavená doba běhu, je zavolána druhá funkce, která spouští časovač číslo dva. V tuto chvíli se stavový řádek změní na oranžovou barvu a zároveň vyskočí stavová hláška na televizi, která upozorní uživatele, že aplikace neobdržela data po X minutách. Druhý časovač funguje velmi obdobně jako první. Odpočítává definovaný časový úsek do doby, nežli aplikace přijme nějaká data z PLC systému. V případě, že se tak stane, časovač se zruší, funkce se ukončí a spustí se opět funkce s prvním časovačem. V případě, kdy časovač doběhne, je zavolána třetí a poslední funkce, která obsluhuje stavový řádek.

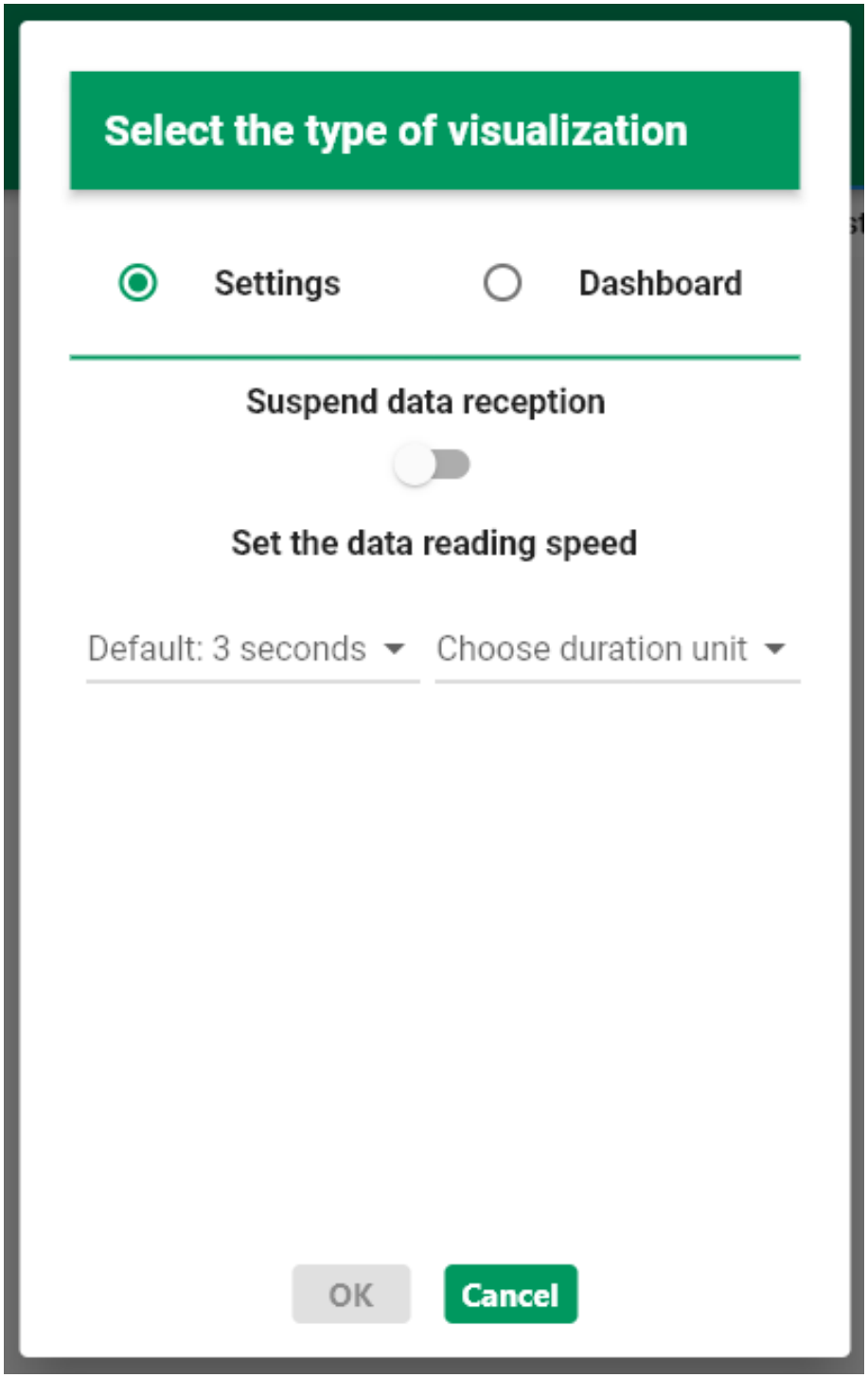
Stavový řádek se změní na červenou barvu, aby dal uživateli najevo, že data stále nechodí. Navíc uživateli vyskočí stavová hláška, že aplikace nepřijala data po Y minutách a také, že bude odhlášen. Tato informace a odhlášení proběhne po deseti vteřinách od doběhnutí druhého časovače. Na začátku této podkapitoly bylo zmíněno, že stavový řádek obsluhují tři časovače. Třetí je právě ten, který zajistí, že se aplikace odhlásí po zmíněných deseti vteřinách.

4.13.2 Nastavení dashboardu

Jak již bylo výše zmíněno, na stránce se nachází tlačítko, které po kliknutí zobrazí nastavení dashboardu. Nastavení se otevře jako vyskakovací okno, které překryje z velké části stránku dashboard. Skládá se ze dvou částí, mezi kterými se lze přepínat (viz obrázky 4.15 a 4.1).



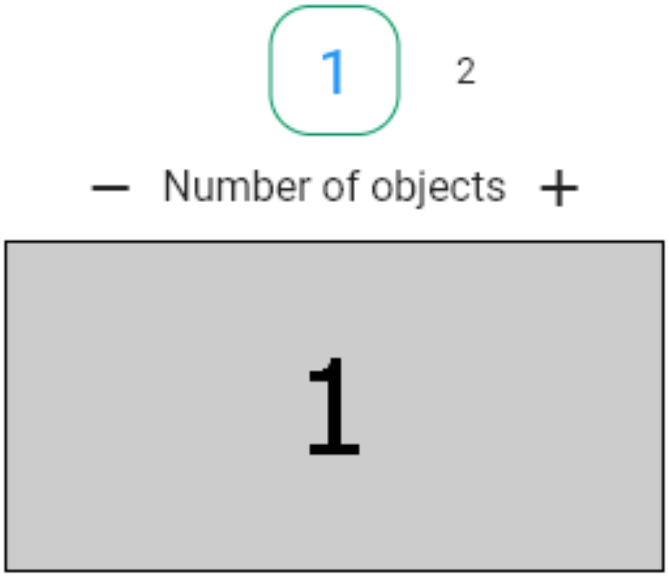
Obrázek 4.15: Nastavení dashboardu 1



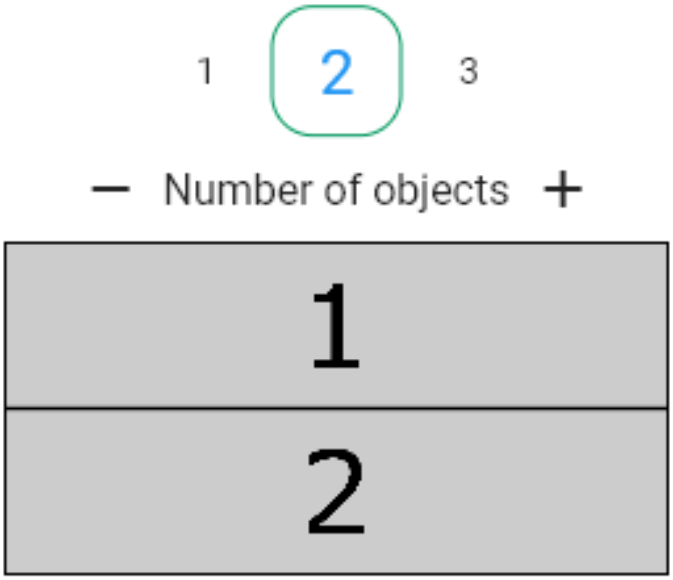
Obrázek 4.16: Nastavení dashboardu 2

Při otevření tohoto nastavení se jako první zobrazí stránka, na které lze definovat, jak a jaká data se budou v dashboardu zobrazovat. Tato stránka se skládá ze dvou sloupců. V druhém sloupci se nachází widget, díky kterému si uživatel přepíná mezi čísly jedna až čtyři. Tato čísla definují počet prvků (grafů, tabulek) v dashboardu. Aby si to uživatel dokázal lépe představit, je pod tímto widgetem na vyběr čísel

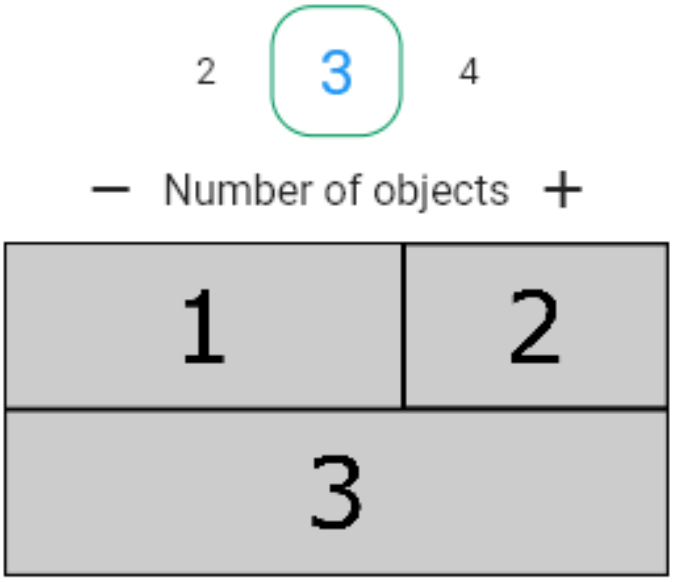
umístěn obrázek, který znázorňuje, jak budou jednotlivé prvky v dashboardu rozmístěny (viz obrázky 4.17, 4.18, 4.19 a 4.20). Zároveň je každý prvek očíslován, aby při následném výběru zobrazení uživatel věděl, kde se jaký graf či tabulka zobrazí. Tento obrázek se mění v závislosti na uživatelem zvoleném čísle. V prvním sloupci se nachází tolik řádků, kolik uživatel zvolí požadovaných prvků v dashboardu. V každém řádku se nachází číslo, které se shoduje s číslem v obrázku, který určuje, kde se jaký prvek zobrazí. Dále, rozbalovací menu, které obsahuje všechny typy zobrazení - sloupcový graf, spojnicový graf, koláčový graf, tabulku a jednu hodnotu. V případě, že uživatel zvolí sloupcový graf, má možnost si také vybrat, zda jej chce zobrazit vertikálně nebo horizontálně. Pokud uživatel zvolí jeden ze tří grafů, tak se v řádku zobrazí další dvě rozbalovací menu, kde uživatel definuje jednotlivé osy, konkrétně osu X a osu Y. V těchto rozbalovacích menu se nachází jednotlivé názvy sloupců přijatých hodnot. Jak již bylo zmíněno v předchozí kapitole, nemusí se zde nacházet všechny názvy sloupců. To je zapříčiněno tím, že ne každý graf může mít na svých osách nečíselné hodnoty. Z toho důvodu zde nejsou uvedeny všechny názvy sloupců, aby aplikace neskončila v chybě. Jakmile uživatel zvolí jako typ zobrazení tabulku, tak nemusí dále definovat nic, jelikož tabulka zobrazí všechny sloupce a jejich hodnoty, které aplikace z PLC přijímá. Poslední možností je volba zobrazení pouze jedné hodnoty. Toto zobrazení bylo přidáno pro případ, že uživatel chce sledovat pouze jednu hodnotu, například teplotu, a nechce ji nechat zobrazit v grafu. Při volbě této možnosti je pouze zapotřebí vybrat požadovaný sloupec, který požadovaná data obsahuje.



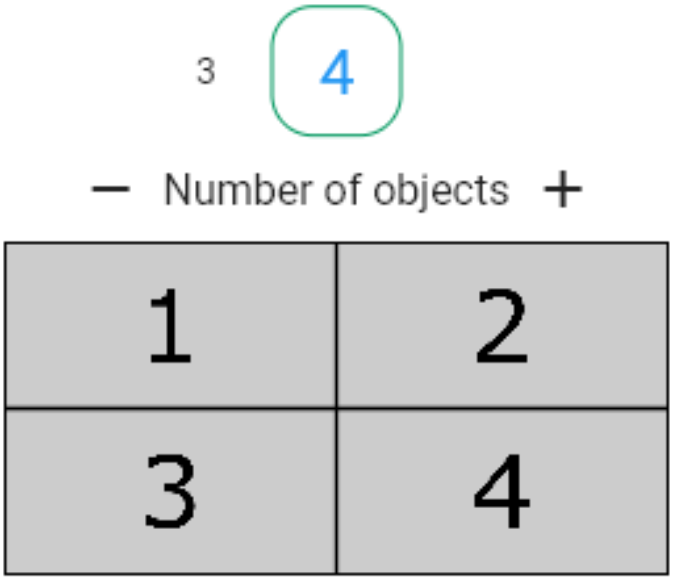
Obrázek 4.17: Rozložení layoutu 1



Obrázek 4.18: Rozložení layoutu 2



Obrázek 4.19: Rozložení layoutu 3



Obrázek 4.20: Rozložení layoutu 4

Jakmile uživatel otevře toto nastavení, aplikace nadále na pozadí přijímá data, ale obnova nově přijatých dat je pozastavena. Důvodem je to, že při testování aplikace bylo zjištěno následující. Uživatel si již zobrazuje přijímaná data na televizi, ale po nějaké době zjistí, že by rád nějaký graf či tabulku změnil, popřípadě by chtěl přidat/odebrat prvky z dashboardu. Jakmile si ale zobrazí nastavení a začne měnit jednotlivé grafy či tabulky, aplikace začne vyhazovat chyby, jelikož se snaží zobrazit grafy, které neměly definované osy, popřípadě se již v dashboardu nenacházely. Z tohoto důvodu se při otevření nastavení obnova dat pozastaví a po opuštění tohoto menu se celý dashboard aktualizuje s nově zvolenými parametry.

Druhá stránka nastavení dashboardu obsahuje přepínač, kterým je možné přijímaná data pozastavit. Respektive aplikace na pozadí stále data přijímá, ale zobrazení je pozastavené. V případě, že se tímto přepínačem pozastaví zobrazování nově příchozích dat, ve spodní části se objeví stavová hláška, která informuje uživatele aplikace, že data jsou pozastavena. Tato funkce byla přidána proto, že PLC může data posílat do televize ve vysoké frekvenci a pokud by uživatel chtěl v určitém čase data analyzovat, má možnost zobrazení nově příchozích dat pozastavit. Druhým prvkem v této části nastavení je čas, který definuje, za jak dlouho se zobrazí nově příchozí data. Výchozí čas obnovy je nastaven na tři vteřiny. Tento čas taktéž definuje čas časovačů, které byly popsány v podkapitole 4.13.1.

Při zvolení času obnovy, který je kratší než třicet vteřin, jsou časovače nastaveny na jednu minutu. To znamená, že pokud televize nepřijme data po dobu jedné minuty, stavová dioda změní barvu na oranžovou a informuje uživatele o tomto problému ve stavovém řádku, který se zobrazí ve spodní části stránky. Po další minutě, tedy celkově po dvou minutách, kdy aplikace nepřijme data, se stavová dioda změní na červenou, informuje uživatele a po dalších deseti vteřinách se

aplikace automaticky odhlásí a zobrazí úvodní obrazovku. Tento princip je totožný pro jakoukoli zvolenou dobu obnovy, mění se jen v závislosti na definovaném čase. Jakmile je čas definován ve vteřinách a je větší nebo rovný třiceti vteřinám, pak se násobí hodnotou tři a výsledek je nastaven do zmíněných časovačů. Pokud jsou zvoleny minuty, do prvního časovače se uloží dvojnásobná hodnota a do druhého hodnota zvolená. V případě volby hodin se do prvního časovače ukládá hodnota, která je jeden a půl krát větší než hodnota zadaná, a do druhého časovače se uloží uživatelem zvolená doba v hodinách.

Posledními widgety, které se v tomto nastavení nacházejí, jsou dvě tlačítka. První tlačítko zruší zvolené parametry, zavře toto vyskakovací okno a ponechá dashboard beze změny. Druhé tlačítko slouží pro potvrzení zvolených parametrů. Toto tlačítko je neaktivní, nedá se na něj kliknout do doby, než jsou zvoleny všechny požadované parametry. To znamená, že pokud si uživatel vybere, že chce, aby dashboard obsahoval tři prvky (například grafy) a definuje pouze grafy dva, je tlačítko neaktivní. Jakmile vyplní všechny požadované hodnoty, tlačítko se zaktivuje a je možné na něj kliknout a přijmout zvolené změny. Po kliknutí si aplikace nastaví potřebné proměnné dle uživatelem zvolených parametrů, zavře okno nastavení a v dashboardu zobrazí požadované informace.

4.13.3 Způsob zobrazení přijímaných dat

Vzhledem k předpokladu, že přijímaná data aplikací, budou v každé firmě a také na každé lince různá, nelze předem definovat strukturu přijímaných dat. Proto byla definována třída s dynamickými proměnnými a konstruktorem definujícím objekt obsahující tyto proměnné, konkrétně osou X a osou Y, které slouží k naplnění grafů hodnotami. Proměnné jsou typu dynamic.

Datový typ dynamic

Dynamický typ je podobný typu Object v tom smyslu, že každý typ „je dynamický typ“. Proto se může anotace dynamického typu používat v deklaracích proměnných a parametrů a jako návratové typy funkcí. Z hlediska jazyka je uvedení dynamického typu totéž, jako není uvedena žádná informace o typu. Když je však dynamický typ uveden, dává se tím ostatním vývojářům najevo, že se programátor rozhodl pro použití dynamického typu, a ne že ho prostě neuvedl. Rozhraní dynamic neposkytuje žádné metody a vlastnosti, je natvrdo zakódováno do virtuálního stroje a na rozdíl od třídy Object jej nelze rozšířit ani z něj dědit. Klíčové slovo dynamic, které identifikuje dynamický typ, je možné použít výslovně místo jiných typových informací. Obvykle se používá tam, kde je potřeba výslovně naznačit, že programátor se rozhodl neposkytovat žádné typové informace. Toto použití se jemně liší od použití slova Object, které se používá, když se programátor výslovně rozhodl povolit jakýkoli objekt.

Vzhledem k tomu, že objekt obsahuje dvě proměnné datového typu dynamic, je

možné přijímat data jakéhokoli typu s tím, že se jejich datový typ zachová. Data přijímaná z PLC zařízení je nutné taktéž někde ukládat. Tudíž bylo vytvořeno čtyřrozměrné pole, které je taktéž datového typu dynamic. Pole je čtyřrozměrné z toho důvodu, aby se do něj mohla ukládat potřebná data k vykreslování jednotlivých grafů; v dashboardu mohou být až čtyři.

Jak bylo výše zmíněno, rozmístění jednotlivých částí dashboardu je označeno unikátním číslem. Díky tomuto číslu lze jednoduše uložit a vyčíst data z čtyřrozměrného pole, které přijímaná data obsahuje. Při zobrazení dat v tabulce, lze uvažovat stejný předpoklad, jako u grafů a to ten, že datové typy sloupců budou obsahovat různé datové typy. Kvůli tomu se vytvořilo pole s datovým typem dynamic, které ukládá všechna přijímaná data. K vykreslení grafů byla použita knihovna s názvem Flutter Charting library, která poskytuje veškeré požadované třídy a funkce zajišťující vykreslování grafů v aplikaci. Pro každý použitý typ grafu byla vytvořena třída, ve které byl definován vzhled a chování konkrétního grafu. Každá třída obsahuje konstruktor, který třídu inicializuje.

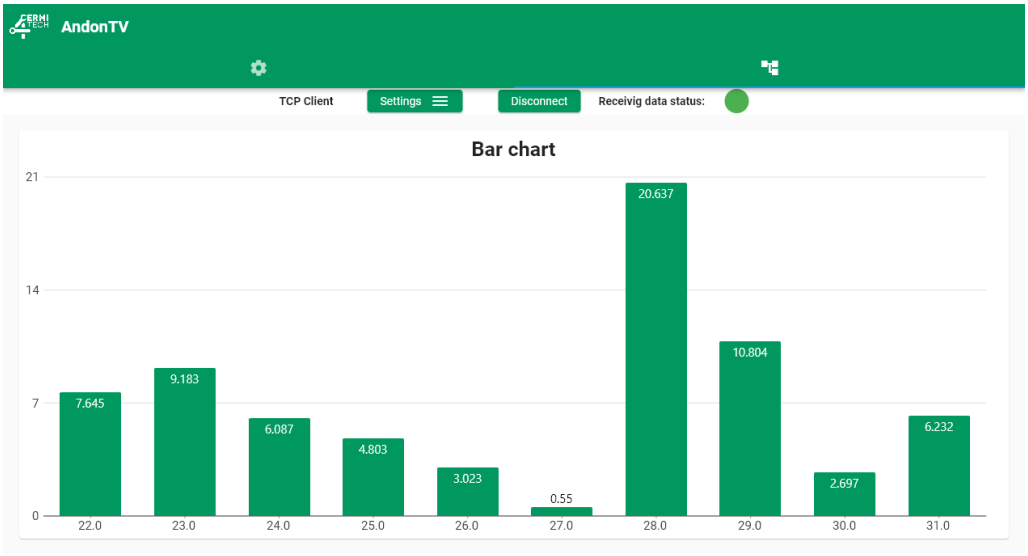
Konstruktor

Konstruktor je speciální metoda třídy nebo struktury v objektově orientovaném programování, která inicializuje nově vytvořený objekt daného typu. Kdykoli je objekt vytvořen, je konstruktor automaticky zavolán. Konstruktor je podobný metodě instance, která má obvykle stejné jméno jako třída, a lze jej použít k nastavení hodnot členů objektu, a to buď na výchozí, nebo na uživatelem definované hodnoty. Konstruktor však, ačkoli se jí podobá, není řádnou metodou, protože nemá návratový typ. Konstruktor inicializuje objekt a nemůže být statický, finální, abstraktní a synchronizovaný. Konstruktory nejsou volány explicitně a jsou vyvolány pouze jednou za dobu své životnosti. V případě hierarchie tříd, kdy odvozená třída dědí z nadřazené třídy, je posloupnost provádění konstruktoru taková, že se nejprve zavolá konstruktor nadřazené třídy a poté konstruktor odvozené třídy. Konstruktory nelze dědit.

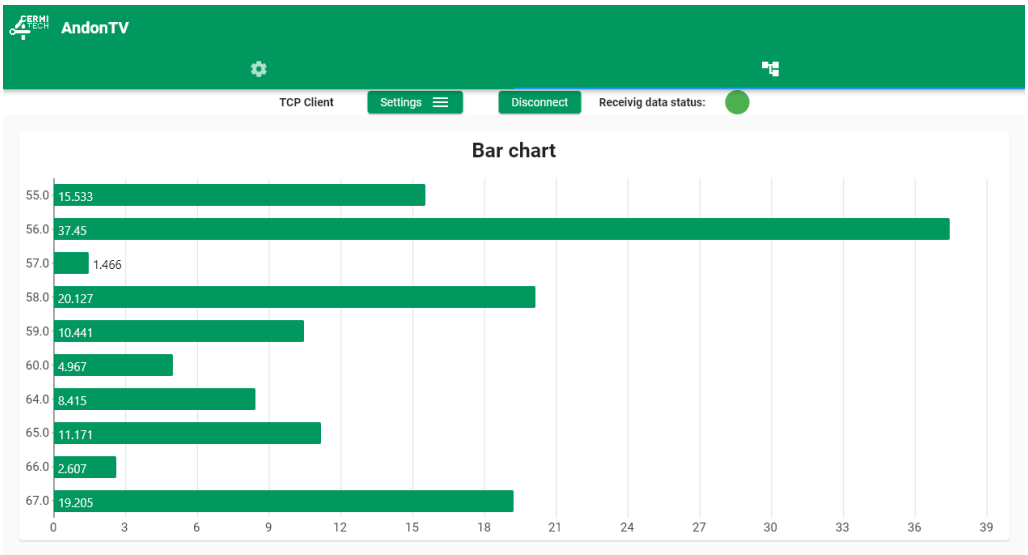
Sloupcový graf

Parametry konstruktoru třídy, která definuje sloupcový graf, je pole přijatých dat definovaných výše zmíněnou a vytvořenou třídou obsahující dvě dynamické hodnoty a údaj datového typu Bool o tom, zda uživatel chce sloupcový graf zobrazit vertikálně, nebo horizontálně. Ve třídě se taktéž nachází definice widgetu, jímž je sloupcový graf. K jeho správnému fungování bylo nezbytné definovat několik parametrů, jako jsou data, data pro osu X a data pro osu Y. Dále se také definovala barva grafu, respektive jeho sloupců. Barva je závislá na uživatelem aktuálně zvoleném barevném tématu. Widgetu se předává také údaj od uživatele, zda chce zobrazit graf vertikálně nebo horizontálně. V případě, že by přijímané hodnoty na ose X byly typu String, byla pro lepší čtení zobrazené hodnoty přidána funkce, která jednotlivé popisky otáčí o čtyřicet pět stupňů. Tato funkce zjistí datový typ dané hodnoty a v případě, že hodnota má datový typ typu String, vrátí číslo čtyřicet pět. V opačném případě

vrátí číslo nula, jakož to otočení o nula stupňů. Dalším parametrem tohoto widgetu, který byl nastaven, jsou popisky jednotlivých sloupců v sloupcovém grafu, které udávají jeho hodnotu. V případě, že je takovýto sloupec dostatečně vysoký (při horizontálním zobrazení dlouhý), hodnota je vepsána do sloupce. V případě, že výška (popřípadě délka) není dostatečná, hodnota je napsána nad (vedle) konkrétní sloupec ve sloupcovém grafu (viz obrázky 4.21 a 4.22). Posledním nastaveným hlavním parametrem byla animace. Animace slouží k tomu, aby při obnově grafu (zobrazení nových dat) se vizualizace pouze neobjevila, ale aby vznikl animační přechod a tím se dosáhlo toho, že vizualizace grafu vypadá plynuleji.



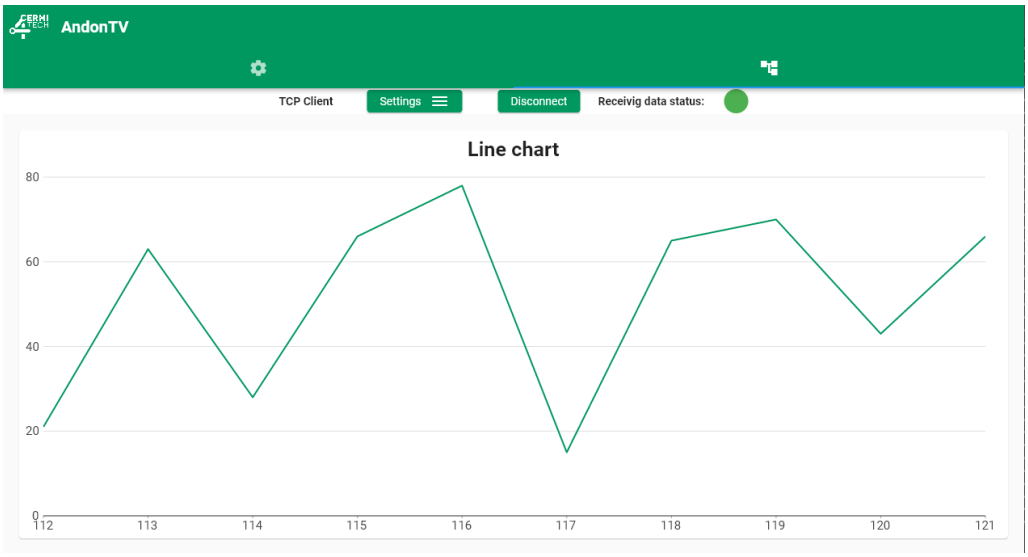
Obrázek 4.21: Sloupcový graf 1



Obrázek 4.22: Sloupcový graf 2

Spojnicový graf

Konstruktor třídy, která definuje spojnicový graf, obsahuje pouze jeden parametr, což jsou data, která jsou definována výše zmíněnou třídou obsahující dvě dynamické hodnoty. V třídě se nachází widget, který definuje vzhled spojnicového grafu a k jeho správnému fungování je nutné mu přiřadit data a také data, která se budou zobrazovat na ose X a ose Y. Dále se také definovala barva grafu, respektive jeho sloupců. Barva je závislá na uživatelem aktuálně zvoleném barevném tématu. Stejně jako u sloupcového grafu i zde se využívá funkce, která otočí popisek hodnoty o čtyřicet pět stupňů na ose X v případě, že datový typ hodnoty je String. Hodnoty spojnicového grafu nemusí začínat od nuly, ale jejich průměrná hodnota může být například sto padesát. Z toho důvodu byla do spojnicového grafu přidána funkce, která zajišťuje to, aby osa Y nezačínala nulou, ale dynamicky se přizpůsobila aktuálně zobrazeným hodnotám. Pokud tedy uvedeme příklad, kdy zobrazené hodnoty ve spojnicovém grafu jsou v rozmezí sto až dvě stě, pak na ose Y nebude zobrazen rozsah hodnot od nuly do dvou set, ale od osmdesáti do dvou set dvaceti. Samozřejmě, že pokud se hodnoty v čase zvyšují nebo snižují, rozsah hodnot na ose Y se také dynamicky mění. Graf díky této funkci vypadá graf lépe, a zobrazované hodnoty jsou plus mínus ve středu (viz obrázek 4.23).

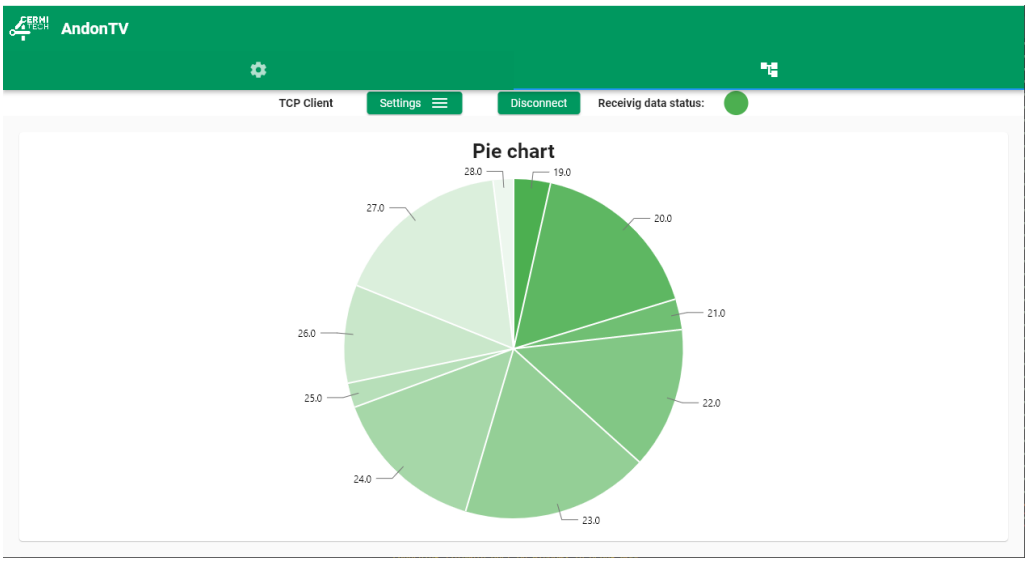


Obrázek 4.23: Spojnicový graf

Koláčový graf

Konstruktor třídy, která definuje koláčový graf, obsahuje taktéž jako spojový graf pouze jeden parametr a to data, která jsou definována výše zmíněnou třídou obsahující dvě dynamické hodnoty. I zde se nachází hlavní widget, který definuje vzhled koláčového grafu a pro jeho správné fungování potřebuje definovat několik parametrů. Základní parametry tohoto widgetu jsou data grafu a data na jednotlivých osách, konkrétně na ose X a Y. Co se týká barvy, i zde je barva závislá na uživate-

lem zvoleném barevném tématu aplikace, ale na rozdíl od předešlých dvou grafů, je zde potřeba mít více definovaných barev, a to z jednoho prostého důvodu. Koláčový graf, jak už název napovídá, je kruh a aby se v něm rozlišily jednotlivé hodnoty, je zapotřebí, aby barev bylo více. Z tohoto důvodu byla přidána funkce, která vytváří různé odstíny uživatelem zvoleného barevného tématu aplikace, tak aby každá zobrazená hodnota měla svou jedinečnou barvu, ale zároveň barva vycházela z uživatelem zvoleného barevného tématu (viz obrázek 4.24). Byla také přidána funkce, která zajišťuje to, že hodnoty jednotlivých výseků koláčového grafu se zobrazují vně. Tato funkce byla přidána proto, že ne každý výsek grafu musí být dostatečně velký, aby se číslo vešlo dovnitř. Posledním nastaveným parametrem byla animace. Animace slouží k tomu, aby se při obnově grafu (zobrazení nových dat) vizualizace pouze neobjevila, ale aby vznikl animační přechod, a tím se dosáhlo toho, že vizualizace grafu vypadá plynuleji.



Obrázek 4.24: Koláčový graf

Tabulka

K vytvoření tabulky v dashboardu nebyla zapotřebí žádná externí knihovna, stačilo využít widgetů, kterými framework Flutter disponuje. Využil se k tomu widget nazvaný Table, díky kterému lze vytvořit tabulku dle představ programátora. Aby se v tabulce lépe daly odlišit jednotlivé hodnoty, bylo do tabulky přidáno ohraničení, které odděluje jak jednotlivé sloupce, tak i jednotlivé řádky. Barva tohoto ohraničení se mění v závislosti na uživatelem zvoleném barevném tématu aplikace. První řádek tabulky definuje názvy jednotlivých sloupců. Aby to bylo pro uživatele na první pohled zřejmé, pozadí toho řádku je barevně odlišeno, a to konkrétně barvou uživatelem zvoleného tématu aplikace. Zároveň bylo písmo nastaveno na tučné, aby byl řádek, respektive názvy sloupců, ještě více odlišen od zbytku hodnot v tabulce (viz obrázek 4.25). Názvy sloupců se generují v závislosti na přijatých klíčích v datech. Zbylé řádky tabulky, kde se nachází už konkrétní přijaté hodnoty z PLC

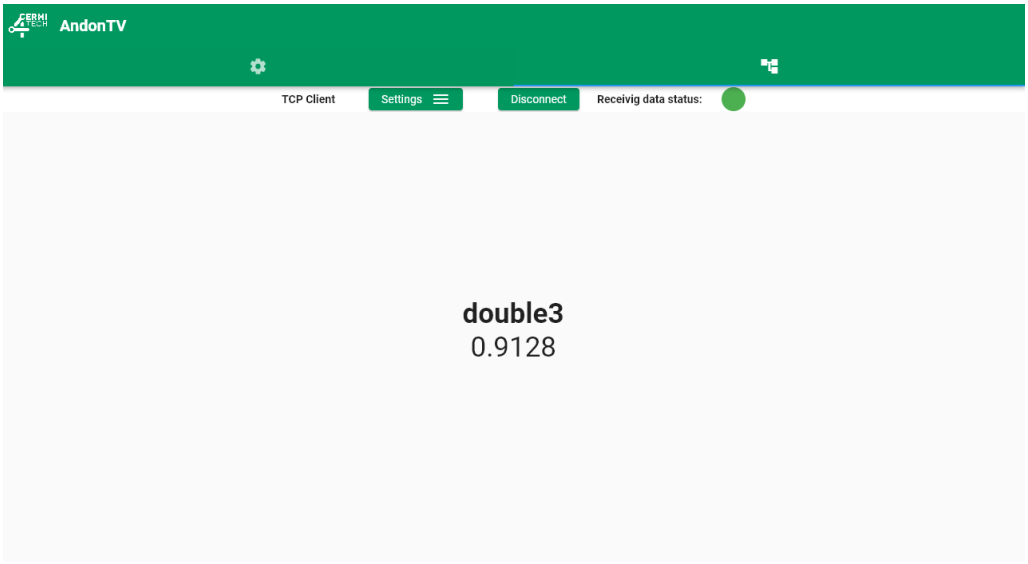
zařízení, nemají žádné barevné pozadí a taktéž nemají tučné písmo. Jelikož se přijatá data ukládají do jednorozměrného pole, bylo nutné správně definovat cyklus, který generoval jednotlivé řádky tabulky. Index v cyklu začíná od čísla nula až do hodnoty, která je menší nebo rovna délce pole přijatých dat mínus délka pole, které obsahuje všechny klíče neboli všechny názvy jednotlivých sloupců. Index se inkrementuje o číslo, které je rovno délce pole klíčů. Při každém cyklu se vytvoří nový řádek tabulky. Pro vytvoření řádku se správnými hodnotami bylo využito dalšího cyklu, který iteruje od nuly až po délku pole, které obsahuje všechny klíče (názvy sloupců). Díky tomuto cyklu bylo pak možné vybírat z pole všech přijatých hodnot správné hodnoty, které do daného řádku a daného sloupce patří. Aby aplikace nevypisovala nekonečně dlouhou (nekonečný počet řádků) tabulku, je zde přidána funkce, která hlídá počet hodnot v poli obsahujícím všechny hodnoty přijaté z PLC zařízení. Před popisem této funkce je nutné definovat jednu proměnnou, která obsahuje hodnotu rovnou maximálnímu počtu všech hodnot v tabulce. Tato hodnota se počítá tak, že se vynásobí délka pole, která obsahuje všechny klíče, číslem udávajícím počet řádků. Díky tomu dostáváme maximální počet hodnot v tabulce. Číslo, které udává maximální počet řádků, se mění dle uživatelem zvolené velikosti fontu písma a také dle uživatelem zvoleného layoutu. Počet řádků je závislý i na layoutu z toho důvodu, že pokud si uživatel chce zobrazit jen a pouze tabulku, počet řádků může být vyšší než v případě, kdy si uživatel chce zobrazit tabulku a například graf. Ve funkci se následně výsledné číslo porovnává s délkou pole, které obsahuje všechny hodnoty. Pokud je délka pole větší než zmíněné číslo, funkce začne odstraňovat přijaté hodnoty ze začátku pole do doby, než je délka pole menší nebo rovna proměnné s maximálním počtem všech hodnot v tabulce. Díky tomu bude aplikace vypisovat jen určitý počet řádků a nejstarší přijaté hodnoty odstraní.

id	Date	int1	int2	int3	int4	double1	double2	double3	bool
129.0	2017-12-26 13:08:43.338782	39.0	46.0	123.0	2.0	0.14	1.824	0.0056	true
130.0	2020-08-07 14:08:44.339754	59.0	71.0	121.0	0.0	35.4	1.732	0.6883	true
131.0	2018-08-14 14:08:45.340755	20.0	413.0	209.0	0.0	23.49	38.438	0.9498	true
132.0	2019-12-07 13:08:46.339754	85.0	282.0	57.0	3.0	49.0	0.494	0.0346	true
133.0	2018-02-12 13:08:47.339777	84.0	476.0	217.0	1.0	6.94	0.822	0.2202	true
134.0	2021-12-04 13:08:48.338760	8.0	109.0	4.0	3.0	12.08	5.423	0.874	false
135.0	2017-06-24 14:08:49.338785	4.0	93.0	220.0	3.0	70.8	1.795	0.9143	true
136.0	2017-11-01 13:08:50.338792	75.0	241.0	96.0	1.0	30.49	22.125	0.3274	false
137.0	2019-05-23 14:08:51.338816	46.0	310.0	148.0	2.0	25.81	15.579	0.9354	false
138.0	2017-10-16 14:08:52.338816	89.0	213.0	107.0	2.0	47.95	44.096	0.313	true
144.0	2021-03-05 13:08:58.339276	92.0	0.0	183.0	2.0	57.85	29.463	0.8609	false
145.0	2019-08-28 14:08:59.340274	98.0	421.0	149.0	0.0	23.72	3.145	0.2586	false
146.0	2020-10-25 13:09:00.339272	81.0	231.0	124.0	2.0	26.62	22.604	0.5074	false
147.0	2018-01-17 13:09:01.339275	0.0	390.0	167.0	2.0	49.8	2.199	0.3754	true
148.0	2021-07-10 14:09:02.339279	65.0	42.0	16.0	2.0	1.0	11.716	0.5697	true
149.0	2020-12-18 13:09:03.338694	72.0	350.0	25.0	1.0	38.37	4.879	0.3581	true

Obrázek 4.25: Tabulka

Jedna hodnota

Aplikace také uživateli nabízí možnost, nechat si zobrazovat pouze jednu hodnotu. Přijímané parametry pro fungování této funkcionality jsou dva, název sloupce a hodnotu datového typu Bool, která rozhoduje o tom, zda se vybraná data mají sčítat. Aplikace následně tuto hodnotu zobrazuje na uživatelem určeném místě v layoutu (viz obrázek 4.26). Krom samotné hodnoty, se také zobrazuje název vybraného sloupce. Pokud je uživatele označen požadavek, aby se přijímané hodnoty sčítali, na daném místě v layoutu jsou také dvě časové informace, které ukazují od kdy a jak dlouho se sčítají vybrané hodnoty. Sčítat lze pouze čísla, takže bylo přidáno do řešení ošetření, aby po zvolení sloupce, který obsahuje nečíselné hodnoty, byla tato možnost zakázána.



Obrázek 4.26: Jedna hodnota

Jak bylo již výše v této kapitole zmíněno, je na uživateli, za jak dlouho chce, aby se data v dashboardu aktualizovala. Pro tuto aktualizaci dat byl vytvořen periodický časovač, který vždy po uživatelem definovaném čase volá další funkci, která slouží jako kontrola uživatelských vstupů (definování grafů a další), inicializace proměnných, popřípadě kontrola počtu zobrazených hodnot v grafech.

V této funkci se nachází několik podmínek. První z nich je ta, která kontroluje, zda uživatel nezměnil v grafu jeho osy. V případě, že ano, vyprázdní pole dat, která se mají zobrazit v grafu, aby se do toho pole mohly začít ukládat nově příchozí, uživatelem požadované hodnoty. Další podmínka kontroluje aktuální počet zobrazených hodnot v grafu. Maximální počet hodnot, které lze v grafu najednou zobrazit, je nastaven na deset, takže pokud pole, které obsahuje všechny hodnoty aktuálně zobrazené v grafu, přesáhne toto číslo, odeberou se jednotlivé hodnoty ze začátku tohoto pole, aby délka pole nepřesahovala číslo deset. Další podmínka kontroluje maximální počet hodnot pro tabulku. Tato funkce byla popsána v podkapitole 4.13.3. Na konci celé této funkce se do výše zmíněného

čtyřrozměrného pole obsahující obsahuje hodnoty pro každý prvek v dashboardu, přiřazují klíče (názvy sloupců) tak, aby aplikace mohla správně vyčíst uživatelem požadované hodnoty.

Jak bylo na začátku této kapitoly zmíněno, když se uživatel poprvé přihlásí do dashboardu, zobrazí se výše popsaná úvodní obrazovka. Obrazovka se objeví pokaždé, kdy v dashboardu není specifikován ani jeden prvek. V opačném případě, aplikace zavolá widget, který obsahuje dashboard. V tomto widgetu se nachází cyklus, který inkrementuje index o jedničku až do počtu zvolených prvků v dashboardu. Díky tomuto cyklu, respektive jeho indexu, lze jednoduše vybrat uživatelem požadované hodnoty z čtyřrozměrného pole, které tyto hodnoty ukládá. Tento widget obsahuje několik podmínek, které volají další widgety definující jednotlivé typy zobrazení. Podmínky jsou závislé na uživatelem vybraných typech zobrazení v nastavení této stránky, které byly popsány v podkapitole 4.13.2.

Volané widgety, což jsou prvky tohoto obalujícího widgetu, potřebují ke správnému fungování několik parametrů. Tyto parametry se předávají při volání jednotlivých widgetů. Parametry pro sloupcový graf jsou osa X, osa Y, data, která se mají zobrazit a informace o tom, zda má být graf vertikální, nebo horizontální. Widget pro spojnicový graf přijímá parametry, jimiž jsou osa X, osa Y a data, která se v grafu mají zobrazit. Parametry koláčového grafu jsou stejné jako u spojnicového grafu, tudíž osa X, osa Y a samozřejmě data, která se mají v grafu zobrazit. Všechny tři druhy grafů následně v těle svých widgetů, volají instanci odpovídající třídy, která definuje jednotlivé grafy. Tyto třídy byly popsány výše v podkapitole 4.13.3. Co se týká tabulky, tak ta přijímá parametr pouze jeden, a to jsou její data. Widget pro zobrazení jedné hodnoty, obsahuje parametry dva, konkrétně název sloupce a informaci, zda se daný sloupec má, nebo nemá sčítat.

Pro správné vykreslení více jednotlivých druhů zobrazení zároveň, bylo také nutné definovat jejich rozložení na obrazovce. Původní záměr bylo využít widget Grid (mřížka), který už v základu obsahuje framework Flutter. Problémem tohoto widgetu ale je, že programátor má kontrolu pouze nad tím, kolik sloupců, popřípadě řádků rozložení může mít. Vzhledem k situaci, kdy uživatel zvolí tři prvky vizualizace, nevypadala by aplikace hezky, jelikož by čtvrtina obrazovky byla nevyužita. Navíc programátor nemá možnost definovat velikost každého okénka v mřížce (Grid) zvlášť. Díky této skutečnosti bylo zjištěno, že tento způsob není ideální a bylo nutné najít jiné řešení.

Tímto řešením bylo využít externí knihovnu Flutter Staggered Grid View. Tato knihovna programátorovi umožňuje definovat nejen různý počet prvků na jednotlivých řádcích, ale také různou velikost každého okénka v mřížce. Aby se dala knihovna využít, bylo nutné vytvořit funkci, která vrací konkrétní rozložení. Ve funkci se nachází switch a ten zvolí správné rozložení dle uživatelem zvoleného čísla v nastavení, které definuje počet prvků v dashboardu. Každé rozložení obsahuje informace o tom, kolik prvků se má v dashboardu zobrazit, jejich pozici a také velikost. Celá

tato funkce následně vrací pole s těmito hodnotami. Díky tomuto listu, který dědí funkce z knihovny, aplikace dokáže správně rozložit jednotlivé prvky v dashboardu.

4.14 Testování

Testování aplikace nebo systému je nutné z důvodu odladění všech chyb tak, aby aplikace za běhu neskončila v chybě, popřípadě aby grafické rozhraní bylo konzistentní a vypadalo za všech situací tak jak má. Toto řešení bylo při vývoji testováno průběžně, a to z toho důvodu, aby se případné chyby nekupily a odladění následně nebylo složitější. Proto se průběžně testovala každá nově přidaná funkcionalita, aby se včas odchytily největší problémy a chyby. Kromě funkcionalit se testovalo také grafické rozhraní na různých velikostech displejů televize, a to z toho důvodu, aby se všechny widgety v aplikaci zobrazovaly správně a na požadovaném místě. Vzhledem k tomu, že existuje velké množství velikosti úhlopříček televizí, je téměř nemožné mít všechny tyto televize k dispozici. Z tohoto důvodu se krom reálné televize využívaly také emulátory, které nahrazovaly reálné televize. K testování bylo také nezbytné mít k dispozici data, díky jimž lze simulovat příjem dat z PLC zařízení. Veškeré druhy testování budou rozepsány v následujících podkapitolách.

4.14.1 Testování grafického uživatelského rozhraní

Pro testování grafického uživatelského rozhraní bylo nutné mít k dispozici různé velikosti televizí neboli televize s odlišnou úhlopříčkou, aby se zajistilo to, že jednotlivé widgety a jejich rozložení budou vypadat na všech typech televizí stejně. Jak bylo zmíněno, je téměř nemožné mít tolik různých televizí a ani při vývoji této aplikace nebylo k dispozici více typů televizí. Z tohoto důvodu bylo nutné využít emulátory, které simulují reálné televize. Jelikož vyvíjený Andon systém je navržen pro televize se systémem Android, lze tyto emulátory získat zdarma díky vývojovému prostředí Android Studio od firmy Google. Android Studio nabízí ke stažení velké množství emulátorů, s jejichž pomocí lze vyvíjené aplikace snadno testovat. Díky tomuto vývojovému prostředí lze stáhnout emulátory se systémem Android všech různých typů telefonů, tabletů, chytrých hodinek a samozřejmě také televizí. Jelikož naprostá většina televizí má poměr 16:9, stačilo stáhnout a nainstalovat několik málo emulátorů s různou úhlopříčkou. To bylo nezbytné, aby se aplikace dala otestovat jak na menších tak na větších televizích. Zároveň lze také u emulátorů vybrat různé verze operačního systémů a díky tomu bylo možné vyvíjenou aplikaci otestovat důkladněji a zjistit, zda různé verze operačního systému Android nemají problém s vyvíjeným Andon systémem.

Aplikace byla z počátku testována pouze na jednom typu, kde se ladily veškeré požadavky na grafické uživatelské rozhraní a také jednotlivé chyby. Následně se aplikace pouštěla na jednotlivých emulátorech, aby se ověřilo, že vzhled se nemění. Díky těmto emulátorům byly objeveny nedostatky a chyby grafického uživatelského rozhraní, které nebyly patrné v původním testovaném emulátoru. Jako první se

projevovaly chyby v některých vyskakovacích oknech, kde původní návrh rozložení byl vcelku roztažený do okrajů a na menším displeji proto widgety takzvaně přetekly. To se projevovalo tím, že se všechny widgety v daném rozložení na obrazovku, respektive do vyskakovacího okna, nevešly a byly vykreslovány mimo toto ohraničení. Aplikace kvůli tomu začala vykazovat chyby, které jsou v aplikaci nežádoucí a způsobují pád aplikace; nebylo samovolné vypnutí. Bylo tedy nezbytné pozměnit původní návrh rozložení a upravit ho tak, aby se tyto chyby nadále neobjevovaly.

Další chyby, zjištěné při testování grafického uživatelského rozhraní, se týkaly případů, kdy se měnila velikost písma pomocí tlačítek. Tato chyba byla zmíněna v podkapitole 4.10.1. Jak bylo v této podkapitole zmíněno, bylo nezbytné, aby každá stránka aplikace měla vlastní proměnné, které definují velikost písma, a tím se zamezilo přetečení. Jelikož prioritou pro zvětšení fontu písma je to, aby zobrazovaný text byl co největší, bylo taktéž nutné, krom vlastních proměnných každé stránky, upravit rozložení na některých stránkách. To bylo třeba upravit, jelikož bylo zjištěno, že při maximálním zvětšení fontu, kdy widgety nepřetekly, neboli nezasahovaly mimo okno aplikace, není font písma dostatečně velký, aby se text dal přečíst z větší vzdálenosti od televize.

4.14.2 Testování funkcionalit

Jednou z hlavních částí testování, bylo testování jednotlivých funkcí, které se v aplikaci nacházejí. Pro změnu jazyka byla vytvořena obalující funkce, jež volá ostatní funkce, které zajišťují vypsání správného textu v požadované jazykové mutaci (viz obrázek 4.5). Funkce, nacházející se v obalové funkci načítají aktuálně zvolený jazyk, tudíž zde bylo testováno, zda funkce vrací požadovaný objekt, který obsahuje aktuálně zvolený jazyk. Jelikož veškeré texty ve všech jazykových mutacích jsou uloženy v JSON souborech, bylo nezbytné otestovat, zda funkce dokázala vyčíst požadované hodnoty dle zadaných parametrů. Jelikož obalující funkce přijímá pouze jeden parametr, což je klíč daného textu, bylo nezbytné ověřit, zda všechny funkce volané touto obalující funkcí vracejí požadovanou hodnotu.

Co se týká barevného tématu, i zde byla vytvořena obalující funkce, která volá jednotlivé vnitřní funkce (viz obrázek 4.4). Funkce přijímá taktéž pouze jeden parametr, který udává uživatelem zvolenou barvu. Na základě této barvy, vnitřní funkce následně požadovanou barvu, popřípadě barevné téma. Zde bylo testování poměrně snadné, jelikož jedna funkce obsahovala switch, který vracel klíč barvy. Zde se pouze testovalo, zda daná funkce vrací správnou hodnotu, dle vstupu. Další funkce vrací objekt, který definuje barevné téma. Tudíž i zde bylo testování vcelku jednoduché, kdy se testovalo, zda při zadaném vstupu vrátí správný objekt.

Pro správu spuštění HTTP serveru je nutné, aby uživatel zadal IP adresu a port. Jak bylo zmíněno v podkapitole 4.10.2 je v řešení přidána funkce, která nalezne IP adresu zařízení, na němž se daná aplikace spouští. Pro testování této funkce bylo nutné pouze ověřit, zda funkcí vrácená IP adresa se shoduje s aktuální

adresou zařízení.

Regulární výraz detekující validní IP adresu, který byl využit u zadávání IP adresy pro HTTP server a PLC zařízení. Zde probíhalo testování následovně. Vytvořily se dva seznamy IP adres, první pro adresy validní a druhý pro adresy nevalidní. Následně se tyto adresy zadávaly do vstupu a testovalo se, zda regulární výraz správně detekuje validní a nevalidní adresy. Díky tomu šlo snadno ověřit, zda regulární výraz je správně napsán.

4.14.3 Testování dashboardu

Při testování dashboardu bylo nejdůležitější zjistit, zda aplikace přijímá data a zda je přijímá ve správném formátu a zda také přijímá správné hodnoty. Otestovat se museli oba typy komunikací, tak aby výsledky testů byly validní. Pro obě použité komunikace v řešení byl vytvořen program pro PLC ve vývojovém prostředí CoDeSyS. Díky tomuto vývojovému prostředí bylo možné se připojit k reálnému PLC a nahrát do něj program. Toto PLC po spuštění programu začalo odesílat testovací data a pomocí aplikace v televizi byla tato data přijímána.

CoDeSyS

CoDeSyS je zkratka pro Controller Development System. Jedná se o vývojové prostředí pro programování řídicích aplikací v souladu s normou IEC 61131-3. Vyvinula jej a stále udržuje německá společnost 3S (Smart Software Solutions). Toto vývojové prostředí je nezávislé na platformě a je kompatibilní s PLC a automatizačními komponenty více než 250 společností. Systém CoDeSyS podporuje všech pět programovacích jazyků PLC, na něž se vztahuje norma IEC 61131-3. Patří mezi ně seznam instrukcí (IL), strukturovaný text (ST), žebříkové diagramy, diagramy funkčních bloků (FBD) a sekvenční diagram funkcí (SFC). Pro výrobce OEM tak vzniká společný systém, který používá společný software. Nezáleží tedy na tom, jaké PLC nebo automatizační komponenty společnosti se používají, společné vývojové prostředí umožňuje návrhářům systémů pracovat s jakýmikoliv z nich. Z pohledu OEM je to mnohem jednodušší, protože člověk nikdy neví, které produkty řídicí společnosti budou do návrhu systému integrovány. CoDeSyS se používá v řadě automatizačních aplikací, včetně automatizace továren, mobilních zařízení a procesů, ale i v automatizaci energetiky a dalších. Pro prostředí CoDeSyS existuje dokonce modul SoftMotion, který spojuje řízení a pohyb do jednoho celku.

TCP/IP

Data použitá pro testování byly reálné hodnoty z PLC z několika výrobních linek, tak aby se ověřilo, zda aplikace dokáže správně přijmout a zobrazit různé datové typy, jako jsou Integer, Byte, Real a další. Při testování byla zjištěna jedna chyba. Jakmile totiž PLC posílala datový typ data, času nebo kombinaci těchto typů, aplikace se zastavila v chybě při čtení posílaných dat. To způsoboval fakt, že se datum, čas a jejich kombinace neposílala jako Stringová hodnota, ale jako výše napsané tři

datové typy. Bylo nutné tedy upravit program pro PLC tak, aby před odesláním dat, převedl výše tři zmíněné datové typy na String. Další úpravou, tentokrát aplikace, kterou bylo třeba udělat, bylo omezení počtu desetinných míst. Při testování bylo zjištěno, že jakmile přijatá hodnota měla více než čtyři desetinná místa a byla zobrazována v grafu, byly hodnoty uváděné v grafu vypisovány do dvou řádků, což mělo za následek špatnou čitelnost. Z toho důvodu bylo do programu přidána funkce pro zaokrouhlování, která přijaté číslo zaokrouhlila na tři desetinná místa.

Modbus TCP

Jak už v práci bylo napsáno, Modbus TCP dokáže přijímat a odesílat pouze data datového typu Word. Z tohoto důvodu byla do aplikace přidán požadavek na uživatele, aby definoval přijímanou strukturu dat, viz podkapitole 4.12.4. Prvními kroky testování bylo ověření, zda aplikace dokáže správně číst data, které mají stejnou nebo menší bitovou délku. Tak se zjistilo, že aplikace dokáže přijímat i analyzovat (parsing) data správně. Následovalo testování datových struktur, které měly bitovou délku větší než datový typ Word, tj. větší než šestnáct bitů. U třiceti dvou bitových hodnot bylo zapotřebí přečíst dvě hodnoty typu Word a následně je správně zpracovat. Toto zpracování bylo pospáno v 4.12.4.

4.14.4 Aplikační provoz

Řešení nebylo v době psaní této práce v ostrém provozu, ale bylo pouze testováno. Důvodem je fakt, že aplikace je vyvíjena pro firmu, která by při vydání aplikace, měla přidanou knihovnu, která bude obsahovat funkce, pro v řešení použité komunikace. Další informace budou rozepsány v následující podkapitole.

4.15 Vylepšení a rozšíření

Řešení v této verzi aplikace obsahuje všechny požadované náležitosti. Avšak několik dalších rozšíření, popřípadě vylepšení, se nabízí. První rozšíření, které se nabízí, je přidávat k samotné aplikaci také knihovny, které budou obsahovat připravené funkce pro PLC, díky kterým bude snadné použít tuto aplikaci i uživateli, kteří nemají s programování PLC zařízení velké zkušenosti. Cílem je, aby aplikace byla co nejvíce takzvaně user-friendly („uživatelsky přátelská“), a proto se již v době psaní této práce pracuje na vytvoření těchto knihoven, pro různé typy PLC. Dalším možným rozšířením je zasílání notifikací uživateli. Notifikace by upozorňovali na různé uživatelem definované milníky formou SMS zpráv nebo e-mailů. Třetím možným rozšířením této aplikace je také přidání dalšího typu komunikace. Tímto typem se nabízí komunikace S7. Tento typ komunikace do řešení přidán prozatím nebyl, jelikož takováto knihovna pro framework Flutter neexistuje. I když takováto knihovna neexistuje, neznamená to, že přidání komunikace S7 je nereálné. Bylo by zapotřebí knihovnu vytvořit a následně ji implementovat do vyvíjené aplikace. V plánu je také rozšíření o funkci, díky které půjde uložit a nahrát do aplikace

soubor, který bude obsahovat všechna uživatelem zvolená nastavení. Po vložení souboru s nastavením do aplikace, se televize, popřípadě počítač, automaticky připojí k PLC a začne zobrazovat požadovaná data v požadovaném layoutu. Zároveň si aplikace také nastaví požadovaný jazyk a barevné téma.

Nápady jsou také na různá vylepšení. Například možnost si nechat zobrazovat tabulku s méně sloupci. Následně bude možné sledovat v tabulce například pouze čtyři hodnoty, a k tomu grafy, které budou obsahovat odlišné data, než jsou obsažena v tabulce.

5 Závěr

Zadáním diplomové práce bylo vyvinou Andon aplikaci pro televizi se systémem Android, která bude v reálném čase zobrazovat data, přijímané z PLC zařízení. Před vývojem samotné aplikace, byla provedena rešerše aktuálně dostupných řešení na trhu. Tato rešerše byla provedena z důvodu zjištění, jejich hlavních nedostatků, které je potřeba do aplikací přidat, aby bylo možné vyvíjený systém nabídnout komukoli. Řešení je napsáno ve frameworku Flutter a programovacím jazyce Dart, díky kterému je možné aplikaci spustit také na počítačích. Kromě multiplatformnosti, řešení také nabízí čtyři různé jazykové mutace, a také čtyři různá barevné témata. Aplikace dokáže přijímat data v reálném čase pomocí komunikací TCP/IP a Modbus TCP, které lze použít všech typech PLC zařízení. Krom toho, je aplikace velice univerzální, jelikož dokáže přijímat různé struktury dat, různých datových typů. U komunikace TCP/IP si aplikace sama zjistí strukturu přijímaných dat, které následně správně rozdělí, k budoucímu zobrazování v layoutu. Z důvodu toho, že komunikace Modbus TCP posílá a přijímá pouze data datového typu Word, byla přidána funkcionalita, díky které lze nastavit adresu registru, počet dat a také její strukturu, včetně datových typů. Funkcionalita umožňuje aplikaci následně přijímat různé struktury dat a tím být univerzálnější.

Při vývoji se také přihlédlo k předpokladu, že uživatel aplikace si bude chtít zobrazit více dat zároveň. A proto byla přidána funkcionalita, která nabízí výběr mezi čtyřmi různými typy layoutů. Každý layout obsahuje různý počet míst o různé velikosti, kde je možné si nechat požadovaná data zobrazit. Aby aplikace nezobrazovala pouze samotné přijímané hodnoty, do řešení bylo přidáno několik typů zobrazení krom samotné hodnoty. Konkrétně se jedná o tabulku, sloupcový graf, spojnicový graf, a také koláčový graf. V práci je také popsána funkcionalita, která hlídá, za je aplikace stále připojena k PLC, a také zda nějaká data posílá.

Z řešení byl také vygenerován APK a EXE soubor, díky kterým bylo možné aplikaci spustit v televizi s operačním systémem Android, a také na počítači s operačním systémem Windows. Řešení bylo následně testováno, pomocí reálného PLC zařízení, které posílalo data po použitých komunikacích do televize a počítače. Testování odhalilo při vývoji některé nedostatky, jako byl problém s velikostí fontu, nebo nesprávnému vyčítání hodnot z komunikace Modbus TCP. Všechny nalezené nedostatky, které byly objeveny, byly opraveny a následně opět testovány, aby se ověřila správnost daných oprav.

Aplikace je plně funkční na televizí se systémem Android, a také na počítačích se systémem Windows. Celé řešení tohoto Andon systému bude k dispozici jako OpenSource, čímž se i s přidanými funkcionalitami stává unikátní. Andon systém je zároveň navržen tak, aby byl co nejvíce user-friendly („uživatelsky přátelský“) a bylo ji možné používat i uživateli, kteří nejsou v tomto oboru příliš znalý.

Seznam použité literatury

1. App Manifest Overview | Android Developers [online] [vid. 2022a-03-17]. Dostupné z: https://developer.android.com/guide/topics/manifest/manifest-intro

2. AVEVA - Global Leader in Industrial Software [online] [vid. 2022b-05-04]. Dostupné z: https://www.aveva.com/en/

3. Dart overview [online] [vid. 2022c-01-28]. Dostupné z: https://dart.dev/overview.html

4. Embedded Industrial Computers for Edge, IoT, & AI | CONTEC. CONTEC [online] [vid. 2022d-05-04]. Dostupné z: https://www.contec.com/

5. Flutter. GitHub [online] [vid. 2022e-01-26]. Dostupné z: https://github.com/flutter

6. Flutter vs Java: Which is Best for Android App Development? [online] [vid. 2022f-01-26]. Dostupné z: https://nimapinfotech.com/blog/flutter-vs-java/

7. Flutter: What is Dart Programming - Javatpoint [online] [vid. 2022g-01-26]. Dostupné z: https://www.javatpoint.com/flutter-dart-programming

8. INTRODUCTION TO MODBUS TCP/IP: Technical Reference – Modbus TCP/IP [online]. 42 [cit. 2022-05-09]. Dostupné z: https://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf

9. Plastic welding machine | Infinity Automation Systems Private Limited. Infinity Automation Systems Pvt. Ltd. [online] [vid. 2022i-05-04]. Dostupné z: http://www.infinityautomations.com/

10. Short history of manufacturing: from Industry 1.0 to Industry 4.0 - KFactory [online] [vid. 2022j-05-04]. Dostupné z: https://kfactory.eu/short-history-of-manufacturing-from-industry-1-0-to-industry-4-0/

11. TCP/IP: What is TCP/IP and How Does it Work? SearchNetworking [online] [vid. 2022k-04-04]. Dostupné z: https://www.techtarget.com/searchnetworking/definition/TCP-IP

12. Visual Studio SCADA | SCADA Solutions in Visual Studio | ATSCADA | SCADA Software | Phan mem SCADA | - ATSCADA - PROFESSIONAL SCADA SOFTWARE. Visual Studio SCADA | SCADA Solutions in Visual Studio | ATSCADA | SCADA Software | Phan mem SCADA | [online] [vid. 2022l-05-04]. Dostupné z: http://atscada.com/

13. What Is the Flutter Framework? | by Perforce. Perfecto by Perforce [online] [vid. 2022m-01-26]. Dostupné z: https://www.perfecto.io/blog/what-is-flutter-framework

14. You Don't Have to Compare Flutter and React Native Anymore | by Shalitha Suranga | Dec, 2021 | Better Programming [online] [vid. 2022n-01-28]. Dostupné z: https://betterprogramming.pub/you-dont-have-to-compare-flutter-and-react-native-anymore-15ddc4c1342a

15. MALÝ, Martin, 2009. YAML: Serializační formát pro ukládání dat. Zdroják [online]. [vid. 2022-03-17]. Dostupné z: https://zdrojak.cz/clanky/yaml-serializacni-format-pro-ukladani-dat/

16. ZAMPARAS, M., V.C. KAPSALIS, G.L. KYRIAKOPOULOS, K.G. ARAVOSSIS, A.E. KANTERAKI, A. VANTARAKIS a I.K. KALAVROUZIOTIS, 2019. Medical waste management and environmental assessment in the Rio University Hospital, Western Greece. Sustainable Chemistry and Pharmacy [online]. 13, 100163. ISSN 23525541. Dostupné z: doi:10.1016/j.scp.2019.100163

17. BALBAERT, Ivo a Dzenan RIDJANOVIC, 2015. Learning Dart. 2th. edition. Birmingham, UK: Packt Publishing Limited. ISBN 9781785287626.

18. MODBUS Protocol Specification [online], 2012. 50 [cit. 2021-10-8]. Dostupné z: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

19. PAYNE, Rap, 2019. Beginning App Development with Flutter: Create Cross-Platform Mobile Apps. 1th. edition. New York, USA: Apress. ISBN 9781484251812