Nástroj na tvorbu grafických prvků pro HMI robotických kontrolérů ABB OmniCore

Kristián Pilát

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Kristián Pilát**
Osobní číslo:	**A19089**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Nástroj na tvorbu grafických prvků pro HMI robotických kontrolérů ABB OmniCore**
Téma práce anglicky:	**Toolkit for Creating Graphical Elements for HMI of ABB OmniCore Robot Controllers**

Zásady pro vypracování

1. Vypracujte literární rešerši na dané téma.
2. Popište způsob tvorby uživatelského HMI pomocí aplikace ScreenMaker na systémech IRC5 a porovnejte se způsobem pro novější typ řízení OmniCore.
3. Vytvořte rozhraní pro snadný výběr a rozložení grafických prvků z knihovny OmniCore App SDK s možností jednoduchého přiřazení uživatelských signálů a stavů robota na tyto prvky.
4. Pomocí navrženého rozhraní vytvořte konkrétní aplikaci pro HMI robota a porovnejte výsledek se stejnou aplikací vytvořenou v aplikaci ScreenMaker pro řízení IRC5.
5. K rozhraní vytvořte krátký manuál v anglickém jazyku –postup instalace a popis jednotlivých ovládacích prvků.
6. Celý projekt vložte na GitHub s korektně vyplněnou README sekcí.

Forma zpracování bakalářské práce: **tištěná/elektronická**
Jazyk zpracování: **Slovenština**

Seznam doporučené literatury:

1. Omnicore SDK: Application Manual. ABB Deceloper Center [online]. ABB, 2021. Dostupné z: https://developercenter.robotstudio.com/omnicore-sdk
2. Omnicore SDK: Design Guideline. ABB Deceloper Center [online]. ABB, 2021. Dostupné z: https://developercenter.robotstudio.com/omnicore-sdk
3. HAVERBEKE, Marijn. Eloquent Javascript: a modern introduction to programming. San Francisco: No Starch Press, 2011. ISBN 978-1-59327-282-1.
4. DUCKETT, Jon. HTML and CSS: Design and Build Websites. Indianapolis: John Wiley, 2011. ISBN 978-1-118-00818-8.
5. Omnicore SDK: Robot Web Services. ABB Deceloper Center [online]. ABB, 2021. Dostupné z: https://developercenter.robotstudio.com/api/RWS

Vedoucí bakalářské práce: **Ing. Ľuboš Spaček**
Ústav řízení procesů

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Kristián Pilát v.r.
podpis studenta

ABSTRAKT

Cieľom bakalárskej práce je zjednodušiť konštruovanie aplikácií pre HMI novej produkčnej rady ABB robotických ramien, ktoré využívajú nový kontrolér typu OmniCore. Výstupom praktickej časti práce bude voľne dostupná aplikácia na platforme Github, umožňujúca užívateľovi jednoducho vytvoriť aplikáciu pre zariadenia HMI bez nutnosti mať akékoľvek znalosti v oblasti webových technológií. Výsledný produkt bude k dispozícii pre viaceré operačné systémy. K aplikácii bude priložený aj návod pre inštaláciu a popis základných ovládacích prvkov.

Klíčová slova: ABB, OmniCore, ScreenMaker, HMI, flexpendant

ABSTRACT

The goal of the bachelor thesis is to simplify the application building proces for HMI of the newest ABB robots production line that are using the new controller type known as OmniCore. Practical part of the thesis will output an open source application available on Github platform, which will help the user to build his very own application for HMI even without knowing anything about web technologies. The result will be compatible with several operating systems. With the application a user manual will be given away that includes installation steps and the core control elements description

Keywords: ABB, OmniCore, ScreenMaker, HMI, flexpendant

Chcem poďakovať vedúcemu práce Ing. Ľubošovi Spačkovi za úsilie a čas, ktorý vynaložil, aby mi pomohol zrealizovať túto prácu. Taktiež ďakujem svojej rodine za podporu, ktorú mi odovzdali nielen počas písania práce, ale aj počas celého štúdia.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Nové typy kontrólerov rady OmniCore sa čoraz viac stávajú súčasťou nových modelov ABB robotov. Výrobca sľubuje vysokú efektívnosť pri menšej spotrebe energie a plno ďalších pre HMI na ovládacom paneli robota. S novou rodinou OmniCore však prichádza aj nový spôsob vývoja aplikácií pre HMI. Pre staršiu verziu kontrolérov IRC5 existuje aplikácia ScreenMaker zabudovaná priamo v RobotStudiu. Vývojár si v tomto grafickom nástroji jednoducho „naklikal" komponenty a priradil im požadovanú funkcionalitu, no pri novej rade OmniCore, výrobca vydal Software Development Kit (SDK), ktorý umožňuje vývoj aplikácií pre HMI. SDK však vyžaduje od užívateľa znalosť základných webových technológii, a preto je cieľom tejto práce vyvinúť aplikáciu podobnú aplikácií ScreenMaker, ktorá bude využívať nový SDK a podporovať kontroléry rady OmniCore. [13]

I. TEORETICKÁ ČÁST

1 WEBOVÉ TECHNOLÓGIE

Webové technológie sa bežne využívajú v každodennom živote, či už na komunikáciu, vyhľadávanie informácií alebo zábavu. Sú to vlastne nástroje a techniky, ktoré napomáhajú v komunikácii medzi rôznymi zariadeniami pomocou internetu. Túto komunikáciu sprostredkúvajú webové prehliadače. Sú to nástroje, ktoré majú za úlohu zobraziť požadované dáta tak, aby boli pre človeka zrozumiteľné.

1.1 HTML

HTML je predvolený jazyk pre webové stránky a aplikácie. Webovým prehliadačom pomáha zobraziť dáta rôznych formátov v definovanej štruktúre a takisto poskytuje možnosť navigácie na stránke vďaka hypertext alebo navigácie medzi ďalšími stránkami.

1.1.1 HTML dokument

Primárnym komponentom HTML dokumentu sú tzv. tagy a elementy. Práve tieto komponenty hovoria webovému prehliadaču ako sa dáta majú zobraziť. Na začiatku súboru je potrebné špecifikovať typ dokumentu.

```
<!DOCTYPE html>
```

Pod deklaráciou typu dokumentu je vložený html tag, ktorý slúži ako kontajner a drží v sebe všetky ostatné tagy. V tomto tagu je možné špecifikovať jazyk, v akom je text v dokumente písaný.

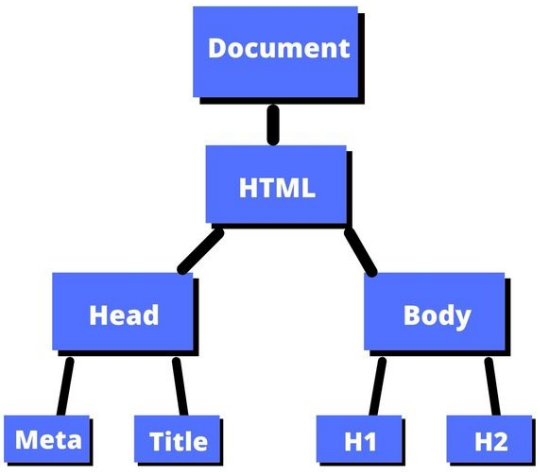
```
<!DOCTYPE html>
<html lang="en-US">
</html>
```

V html tagu sa nachádza head tag, ktorý obaľuje metadáta s informáciami o webovej stránke. Tieto informácie sú zapísané v ďalších tagoch so špecifickou funkciou. Pod head tagom sa nachádza body tag, ktorý obaľuje všetok obsah webovej stránky. Ide napríklad o text, obrázky, tabuľky a mnoho ďalších formátov dát. [8]

```
<!DOCTYPE html>
<html lang="en-US">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Document</title>
    </head>
    <body>
        <h1>Title</h1>
        <p>paragraph</p>
    </body>
</html>
```

1.1.2 Document object model (DOM)

DOM je dátová reprezentácia objektov, ktorá tvorí štruktúru a obsah celého dokumentu. Je to programovacie rozhranie, ktoré umožňuje meniť alebo zmazať elementy nachádzajúce sa v dokumente, ale takisto je možné aj vytváranie nových elementov. DOM sa vyobrazuje ako strom a každý jeden element dokumentu je ako vetva tohoto stromu. [14]



Obrázok 1: DOM [14]

1.2 CSS

CSS je style sheet jazyk, ktorý slúži pre úpravu vzhľadu a formátu dokumentu. Najčastejšie je tento jazyk používaný s HTML typom dokumentu, avšak jazyk je schopný vizuálne upravovať aj dokumenty viacerých typov XML.

1.3 Typy CSS

- Inline CSS – tento typ CSS sa zapisuje pre konkrétne elementy pomocou atribútu „style“, do ktorého sa vkladajú kľúčové slová a ich hodnoty oddelené bodkočiarkou.

```
<p style="font-size: 20px; color:rgb(118, 118, 118)">
  paragraph
</p>
```

- Embedded CSS – pri tomto type je potrebné vložiť style tag do tagu head, kde sa nachádzajú aj ostatné nastavenia dokumentu. Štýly sa následne píšu do tohoto tagu.

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        p {
            font-size: 20px;
            color:rgb(118, 118, 118);
        }
    </style>
</head>
```

- External CSS – v poslednom, externom type je potrebné vytvoriť samostatný súbor s príponou css, do ktorého sa štýly zapisujú. Tento súbor sa nalinkuje v head tagu dokumentu pomocou link tagu. [8]

```
<head>
    <link rel="stylesheet" href="style.css"/>
</head>
```

1.4 JavaScript

JavaScript je dynamický programovací jazyk, používaný pre vývoj webových aplikácií, hier a mnoho ďalších produktov. Pomocou JavaScriptu je možné implementovať dynamické prvky, ktoré by HTML a CSS samé o sebe nezvládli. Štýl použitia v dokumente je podobný ako v prípade CSS. Je na výber inline, teda priamo v elemente, embedded so script tagom v hlavičke dokumentu a external JavaScript, kód v samostatnom súbore, ktorý je do dokumentu nalinkovaný. [9]

```
<body>
    <!-- Inline -->
    <button onclick="alert('Button has been clicked')">Click me!</button>
    <!-- Embedded -->
    <script>
        alert("I am inside a script tag");
    </script>
    <!-- External -->
    <script src="scripts.js"></script>
</body>
```

JavaScript je client-side, teda programovací jazyk, čo z neho technicky robí programovací jazyk pre front-end, avšak v dnešnej dobe sú už k dispozícii rôzne frameworky, ktoré JavaScript obohacujú o novú funkcionalitu a použitie.

2 FRAMEWORKY

Framework je v podstate rozšírenie programovacieho jazyka. Používanie frameworkov je veľmi dôležité, pretože vývojárom ušetria značnú časť práce a tým pádom aj čas. Vývojár nemusí začať vyvíjať aplikáciu úplne od začiatku, pretože vybraný framework sa už stará o základné veci. Vývojár sa teda musí postarať len o vyššie levely funkcionality pri stavbe aplikácií.

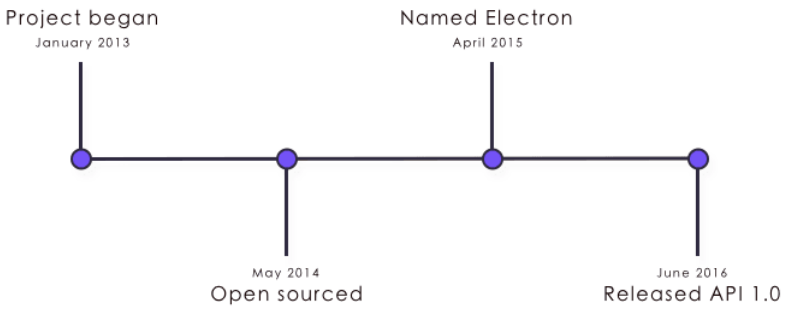
2.1 Electron.js

Electron.js je najpopulárnejší voľne dostupný javascript runtime framework pre vývoj cross-platform desktopových aplikácií za použitia bežných webových technológií. Framework je stavaný na Node.js a Chromium a je používaný na vývoj veľa známych aplikácií ako napríklad Visual Studio Code, Slack, Microsoft Teams alebo Discord a mnoho ďalších. [1][3]

Obrázok 2: Logo Electron.js [15]

Vývoj Elentron.js začal už v januári roku 2013. Hlavná myšlienka bola vytvoriť nástroj pre zjednodušenie vývoja cross-platform textového editoru, na ktorom bude možné pracovať s technológiami ako HTML, CSS a JavaScript. Framework bol založený 15. júla v roku 2013, s úmyslom zjednodušiť vývoj aplikácie „Atom“, ktorá bola zo začiatku známa ako

„Atom Shell“. Na obrázku nižšie je možné vidieť graficky znázornený priebeh vývoja frameworku. [1]

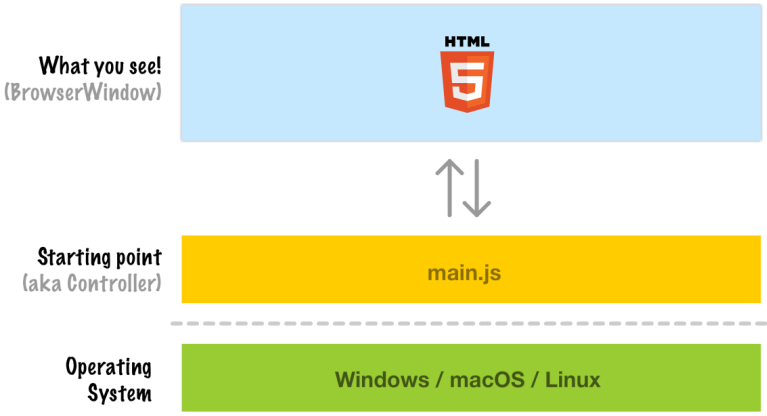


Obrázok 3: Vývoj Electron.js [1]

Akákoľvek webová aplikácia môže byť prestavaná na desktopovú aplikáciu za použitia Electron.js. Vďaka tomu, že framework využíva bežné webové technológie, ako HTML, CSS a JavaScript, vývojár nepotrebuje mať skúsenosť s natívnym vývojom aplikácií pre každú platformu, pre ktorú má byť aplikácia určená. Aplikáciu stačí prispôsobiť pre jeden internetový prehliadač. Súborový systém funguje cez Node.js api, a teda funguje na operačnom systéme Windows, Linux alebo Mac OS X. [1]

Node.js využíva npm modul, ktorý sa v dnešnej dobe používa pre vývoj webových aplikácií, a ktorý obsahuje veľké množstvo balíkov a knižníc, ktoré z veľkej časti zjednodušujú vývoj. Node.js taktiež poskytuje natívne menu pre dialógy a notifikácie. Inštalačný wizard na platforme Windows nepotrebuje zložité konfigurácie.

Z webových aplikácií sme zvyknutí, že „štartovací bod“ aplikácie je html súbor, no s Electron.js to tak nefunguje. Štartovacím bodom aplikácie je v tomto prípade JavaScript súbor, ktorý sa tvári ako kontrolér, v ktorom je potrebné deklarovať, čo má aplikácia vykonávať.



Obrázok 4: Vizualizácia štruktúry Electron.js aplikácie [2]

V kóde je potrebné vytvoriť okno aplikácie, v ktorom sa špecifikujú základné parametre a cesta, kde má aplikácia hľadať prvý HTML súbor. Následne sa vytvorí okno aplikácie, ktoré je v podstate okno prehliadača, ktorý zobrazí webovú stránku v poskytnutom HTML súbore [1]

```
import { BrowserWindow } from 'electron';

const createWindow = () => {
    const win = new BrowserWindow({
        width: 800,
        height: 600
    });

    win.loadFile('index.html');
};
```

Výhodu aplikácií vytvorených pomocou Electron.js je, že na rozdiel od tradičných webových aplikácií, majú prístup k natívnym api. Dôvod je taký, že webový prehliadač poskytuje rôzne bezpečnostné prvky, aby prístup k natívnym api zabránil, avšak Electron tieto bezpečnostné prvky obchádza. Existuje viacero spôsobov ako sa k natívnym api dostať, vo väčšine prípadoch sa pre to využíva Node.js. Napríklad pri implementácii menu stačí vytvoriť šablónu s preddefinovanými rolami, avšak možné je definovať aj vlastné role, pre ktoré je potrebné špecifikovať event listener a jeho funkcionalitu. Určite treba podotknúť, že týmto sa vytvorí menu pre všetky podporované platformy. Názvy jednotlivých položiek sa však dajú podmienečne upravovať, aby jednotlivé hlášky evokovali natívnosť aplikácie čo najdôveryhodnejšie.

```
import { Menu } from "electron";

const template = [
    ...(isMac
        ? [
              {
                  label: app.name,
                  submenu: [
                      { role: "about" },
                      { type: "separator" },
                      { role: "services" }
                  ],
              },
          ]
        : []),
    {
        role: "help",
        submenu: [
            {
                label: "Learn More",
                click: async () => {
                    const { shell } = require("electron");
                    await shell.openExternal("https://github.com/kpilat/omnicore-
screenmaker");
                },
            },
        ],
    },
]

const menu = Menu.buildFromTemplate(template)
Menu.setApplicationMenu(menu)
```

2.2 Vue.js

Vue.js je progresívny JavaScript framework, používaný pre vývoj užívateľských rozhraní a jednostránkových aplikácií. Framework je veľmi obľúbený najmä vďaka jednoduchosti. Pre začatie vyvíjania aplikácií pomocou Vue.js prakticky stačí znalosť technológií HTML, CSS a JavaScript. Zakladateľ Vue.js, Evan You, prišiel s myšlienkou vytvoriť najlepší framework kombináciou už existujúcich frameworkov Angular a React. O obľúbenosti Vue.js hovoria aj čísla. Projekt na github má najviac hviezdičiek v porovnaní s konkurenčnými frameworkmi, napriek tomu, že jeho komunita je rozhodne menšia. [3]

Obrázok 5: Logo Vue.js [16]

Oficiálna dokumentácia je rozsiahla a patrí medzi najlepšie popísané dokumentácie. Vývojára prevedie a všetko dopodrobna vysvetlí na rôznych príkladoch. Tento bod je jeden z dôvodov, prečo sa dá Vue.js relatívne rýchlo naučiť a začať používať na svojich projektoch.

2.2.1 Výhody

- Framework má veľkosť len 21kB, čo z neho robí malý a rýchly framework oproti konkurencii.
- Vue.js disponuje tzv. virtual DOM. Je to v podstate kópia reálneho DOM, ktorá je synchronizovaná s reálnym DOM. Tento prístup je oveľa viac efektívnejší ako pri každej zmene priamo aktualizovať reálny DOM. Toto zaručuje veľmi rýchle renderovanie.

- Možnosť používať oficiálne knižnice frameworku. Tieto knižnice rozširujú Vue.js o novú a lepšiu funkcionalitu.

2.2.2 Nevýhody

Vue.js má taktiež svoje nevýhody, ktoré sa odrážajú na jeho popularite. Framework nie je na trhu tak dlho ako jeho konkurenti, čo zapríčiňuje, že vývojári sa prikláňajú k výberu z konkurenčných frameworkov.

- Prvou z nevýhod je určite jazyková bariéra, keďže framework je pôvodom z Číny, kde je popularita frameworku najväčšia. Rôzne diskusie na fórach alebo inštrukcie k pluginom sú práve v čínskom jazyku. V preklade teda môže dôjsť k nedorozumeniam.
- Vue.js chýba podpora adaptácie pre rozsiahle projekty. Technológia nie je úplne dostačujúco stabilná a neponúka instantné opravy, ktoré veľké spoločnosti môžu vyžadovať. Toto však konkurencia Vue.js ponúka v plnom rozsahu.

2.3 Sass

Sass je jeden z najpoužívanejších css preprocesorov. Je to rozšírenie pre css, ktoré však nijak nerozširuje funkcionalitu. Preprocesor dokáže len to, čo dokáže css. Rozdiel je v zápise kódu. Je to skriptovací jazyk, ktorý sa skompiluje do podoby obyčajného css. Vývojárom dokáže ušetriť množstvo času vďaka možnosti zanorenia selektorov, vlastných sass premenných, možností vytvárania „funkcií“ a mnoho ďalších užitočných vecí. [7]

Obrázok 6: Sass logo [17]

Výnimočná možnosť, ktorú sass ponúka, je možnosť vybrať si z dvoch typov syntaxí. Záleží od prípony sass súboru. V prípade, že má súbor príponu „.scss“, tak za každým selektorom je potrebné použiť zložené zátvorky a na konci riadkov bodkočiarku. Na rozdiel od súborov s príponou „.sass“, kde sa nepoužívajú zložené zátvorky, ale odsadenie a nie je potrebné písať bodkočiarky. Sass ponúka aj možnosť konvertovania týchto typov súborov medzi sebou.

```
// SCSS
.class {
    background-color: #fff;
    color: #000;
}
// SASS
.class
    background-color: #fff
    color: #000
```

2.3.1 Zanorenie selektorov

Možnosť zanorenia selektorov urýchľuje prácu, zvyšuje prehľadnosť a znižuje množstvo kódu, ktorý treba napísať oproti css. Zanorovanie je však dvojsečná zbraň. Odporúča sa zanorovať maximálne dvakrát, kvôli menšej špecifickosti. Viac levelov zanorenia môže spôsobiť chaotickú štruktúru kódu a problém „prebiť“ prioritu, ak je to neskôr potrebné.

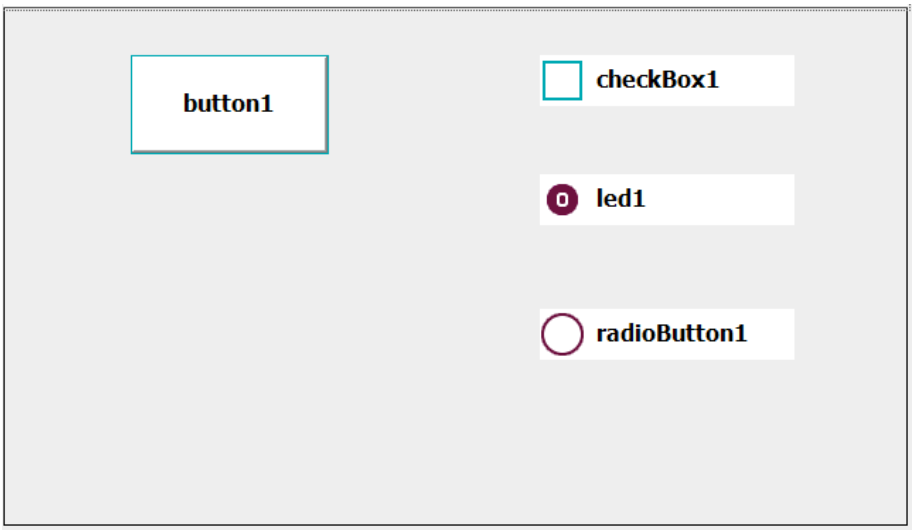
```
// SCSS
.section-header {
    margin-bottom: var(--mb);
    color: var(--secondary);

    &__icon {
        display: inline-block;
        background-color: var(--
secondary);
        width: em(34);
        height: em(34);
        @include border-radius-d-shape;
        margin-right: em(8);
        vertical-align: middle;

        svg {
            color: #fff;
            width: 52%;
            margin: auto;
            margin-right: 30%;
        }
    }
}
```

```
// Vygenerované CSS
.section-header {
    margin-bottom: var(--mb);
    color: var(--secondary);
}
.section-header__icon {
    display: inline-block;
    background-color: var(--secondary);
    width: 1.88889em;
    height: 1.88889em;
    border-top-right-radius: 100%;
    border-bottom-right-radius: 100%;
    overflow: hidden;
    margin-right: 0.44444em;
    vertical-align: middle;
}
.section-header__icon svg {
    color: #fff;
    width: 52%;
    margin: auto;
    margin-right: 30%;
}
```

Obrázok 7: Porovnanie SCSS a vygenerovaného CSS kódu

3 SCREENMAKER

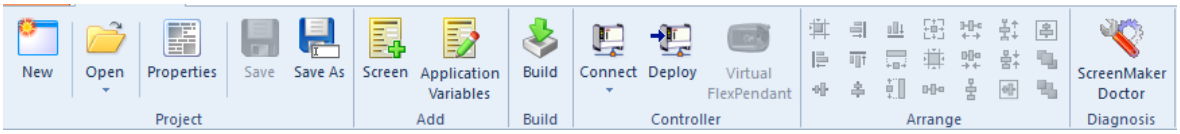
ScreenMaker je nástroj v aplikácii RobotStudio a slúži k vytváraniu aplikácií pre HMI. Vďaka tomuto nástroju je vytváranie aplikácií jednoduché a nie je potrebná znalosť inak potrebných programovacích jazykov. Užívateľ má možnosť si rozvrhnúť GUI, ľubovoľne rozložiť komponenty a priradiť im požadovanú funkcionalitu. [5]



Obrázok 8: IRC5 ScreenMaker GUI

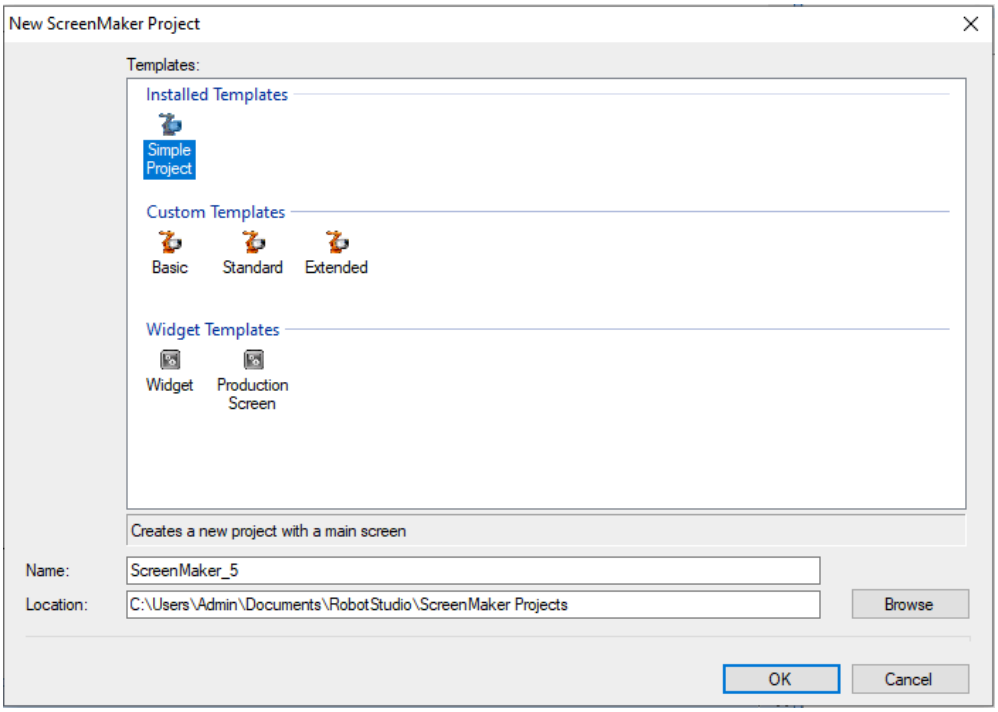
3.1 Postup vytvorenia aplikácie

Po spustení nástroja priamo z RobotStudia sa otvorí nové okno s množstvom ovládacích prvkov. V hornej lište „ribbon“ sa nachádza niekoľko akčných tlačidiel. Tlačidlo umiestnené hneď ako prvé zľava, slúži pre vytvorenie nového projektu.



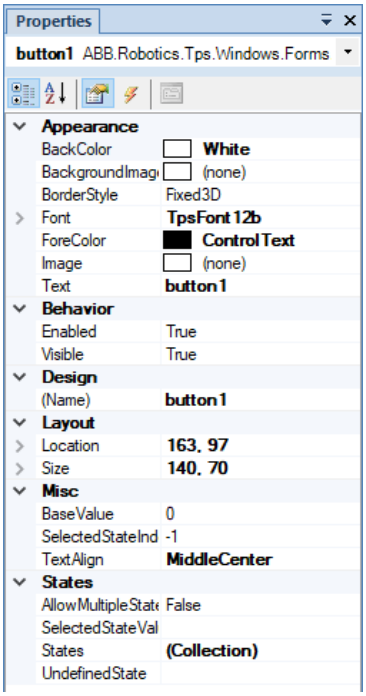
Obrázok 9: ScreenMaker ribbon

Po kliknutí na akčné tlačidlo sa zobrazí nové okno, v ktorom je výber viacero rôznych šablón. Na výber je viacero kategórií. Pre túto ukážku bol vybraný „Jednoduchý projekt“ z prvej ponúkanej kategórie.



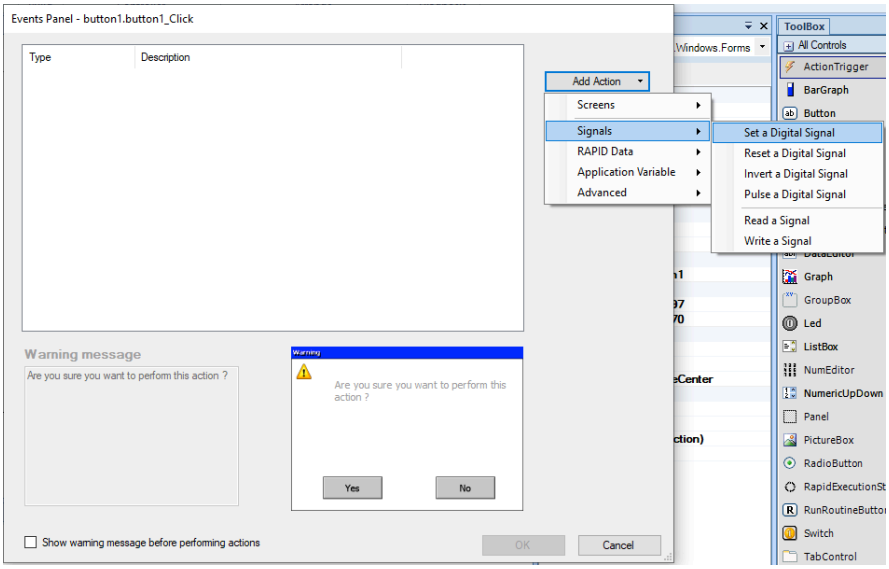
Obrázok 10: Výber šablóny

Následne je vytvorená obrazovka, do ktorej je možné vkladať komponenty z „Toolboxu“, zvyčajne umiestnenom na pravej strane aplikácie. Komponenty sa do obrazovky dajú vkladať jednoducho štýlom „Drag and drop“. Takto vložený komponent má v okne „Properties“ možnosti nastavenia vzhľadu.



Obrázok 11: Možnosti komponentov

Toto sú však len vizuálne nastavenia. Ďalej je potrebné komponentu nastaviť akciu, ktorú bude po kliknutí vykonávať. Dvojitým kliknutím na komponent sa zobrazí okno nazývané „Events Panel“, kde je možné pridať akciu a jej typ. Po kliknutí na „Add Action“ sa zobrazí kontextové menu, v ktorom vyberieme položku „Signals“ a následne klikneme na položku „Set a Digital Signal“.



Obrázok 12: Priradenie akcie

V ďalšom okne sú vypísané signály dostupné v kontroléri robota, z ktorých je možné vybrať jeden signál a výber potvrdiť kliknutím na tlačidlo „OK“. Podobným spôsobom je možné priradiť ďalšie komponenty z okna „ToolBox“ a priraďovať k ním požadované akcie. Aplikácia sa následne zostaví kliknutím na tlačidlo „Build“, ktoré sa nachádza v hornej lište. Pre overenie a vyskúšanie funkčnosti aplikácie je možnosť spustenia aplikácie vo virtuálnom FlexPendante (ovládacom paneli robota).

3.2 Porovnanie s OmniCore

S predošlou generáciou kontroléru IRC5 boli zaužívané určité postupy pre vytvorenie aplikácie pre HMI. Tieto postupy a dokonca aj používané technológie ABB úplne zmenilo. S novým OmniCore HMI spravili obrovský krok vpred a vydali sa úplne iným smerom. Prišli s možnosťou vytvárať rozhrania formou webových aplikácií za použitia základných webových technológií ako HTML, CSS a JavaScript.

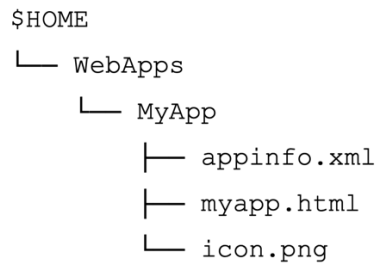
Takže prakticky každé rozhranie na OmniCore HMI, je webová stránka alebo teda webová aplikácia, bežiaca v stavanom webovom prehliadači. Tento prehliadač využíva jadro prehliadača Microsoft Edge.

Aplikácia však potrebuje s robotom aj nejakou cestou komunikovať. Na to sa využíva „Robot Web Services REST API“ (RWS), ktorá beží na strane kontróleru. RWS knižnica je dostupná v JavaScript súboroch, ktorú by mala obsahovať každá aplikácia.

Komponenty, ktoré je v aplikácii možné použiť prichádzajú tiež oddelene v JavaScript ale aj CSS súboroch, kde sú definované kaskádové štýly pre každý z komponentov. Do aplikácie sa teda dajú importovať len naozaj potrebné komponenty, ktoré sú v aplikácií použité. Tento postup vie následne ušetriť miesto, ktoré aplikácia bude zaberať. [10][11]

3.2.1 Štruktúra aplikácie pre OmniCore

Aplikácie musia spĺňať určité požiadavky aby ich Flexpendant mohol načítať a zobraziť na domovskej stránke. Aplikácia musí byť umiestnená v priečinku „WebApps“, kde kontrolér očakáva, že sa bude aplikácia nachádzať. Aplikácia je umiestnená v ľubovoľne nazvanom podpriečinku.



Obrázok 13: Štruktúra OmniCore aplikácie [11]

Každá aplikácia sa skladá z troch súborov. Súbor „myapp.html“ obsahuje HTML stránku aplikácie, ktorá slúži ako štartovací bod, do ktorého sa importujú ostatné konfiguračné súbory. „appinfo.xml“ už podľa názvu obsahuje základné informácie o aplikácii. Obsahuje najmä názov aplikácie, ktorý sa zobrazuje na domovskej stránske po otvorení Flexpendantu. Priečinok, v ktorom je aplikácia umiestnená sa však môže volať rozdielne. Kontrolér si vždy len prehľadáva podpriečinky „WebApps“ a hľadá práve tieto XML súbory. Ďalej odkazuje na ikonu aplikácie a na HTML súbor, ktorý sa má po spustení načítať. Súbor „icon.png“ je obrázok, ktorý je zobrazený ako ikona aplikácie na domovskej stránke.

```
<?xml version="1.0" encoding="UTF-8"?>

<WebApp>
    <name>MyWebApp</name>
    <icon>myicon.png</icon>
    <path>myapp1.html</path>
</WebApp>
```

Obrázok 14: Obsah súboru appinfo.xml [11]

Aplikácia samozrejme neobsahuje len tieto tri súbory. Ako už bolo spomenuté, je potrebné pridať knižnice, komponenty a JavaScript súbor, ktorý zaisťuje logiku a funkčnosť aplikácie.

II. PRAKTICKÁ ČÁST

4 OMNICORE SCREENMAKER

Aplikácia je vytvorená modernými webovými technológiami. Predošlá aplikácia bola súčasťou RobotStudia, a tak ju užívateľ dokázal spustiť len na operačnom systéme Windows. Nová aplikácia je postavená na frameworku Electron.js, ktorý zaisťuje, aby mohla bežať na rôznych operačných systémoch. O „user interface“ (UI) sa stará framework Vue.js, ktorý zaisťuje flexibilitu renderovania komponentov podľa „stavov“ aplikácie.

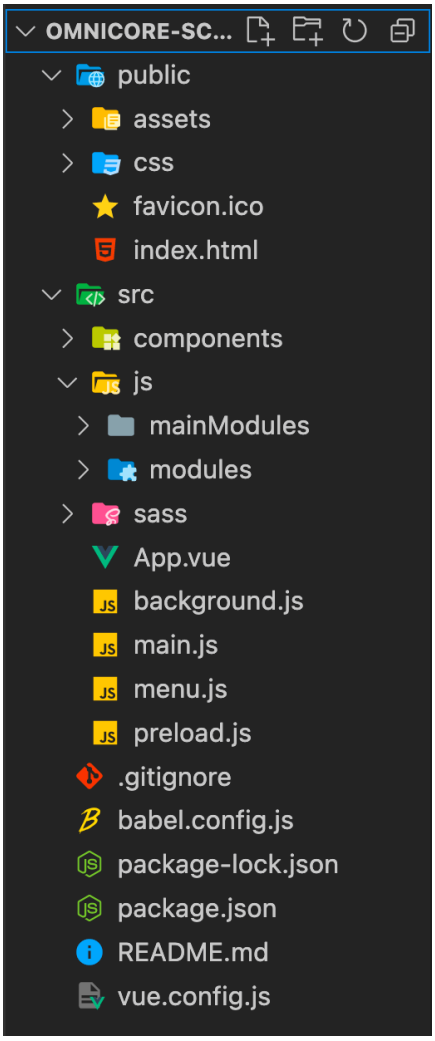
Tvorba tejto aplikácie bola inšpirovaná už existujúcou aplikáciou ScreenMaker pre kontroléry IRC5. Aplikácia teda disponuje podobnou funkcionalitou, len oveľa zjednodušenou formou.

4.1 Štruktúra projektu

Snahou bolo vytvoriť štruktúru projektu tak, aby bola prehľadná, dobre pochopiteľná a aby umiestnenie súborov a priečinkov dávalo čo najväčší zmysel. V samotnom koreňovom priečinku nájdeme len konfiguračné súbory použitých technológií a „read me“ súbor, v ktorom sú popísané kroky ako aplikáciu rozbehnúť, používať a zostaviť.

V priečinku „src“ sa nachádzajú ďalšie podpriečinky. V priečinku „components“ sa nachádzajú Vue.js komponenty. Priečinok „js“ je rozdelený na dva podpriečinky, kde priečinok „mainModules“ obsahuje JavaScript súbory, ktoré zabezpečujú funkcionalitu hlavného procesu a priečinok „modules“ zase naopak zabezpečuje funkcionalitu renderer procesu. V priečinku „sass“ sa nachádzajú súbory css preprocesoru, ktorý zabezpečuje UI aplikácie.

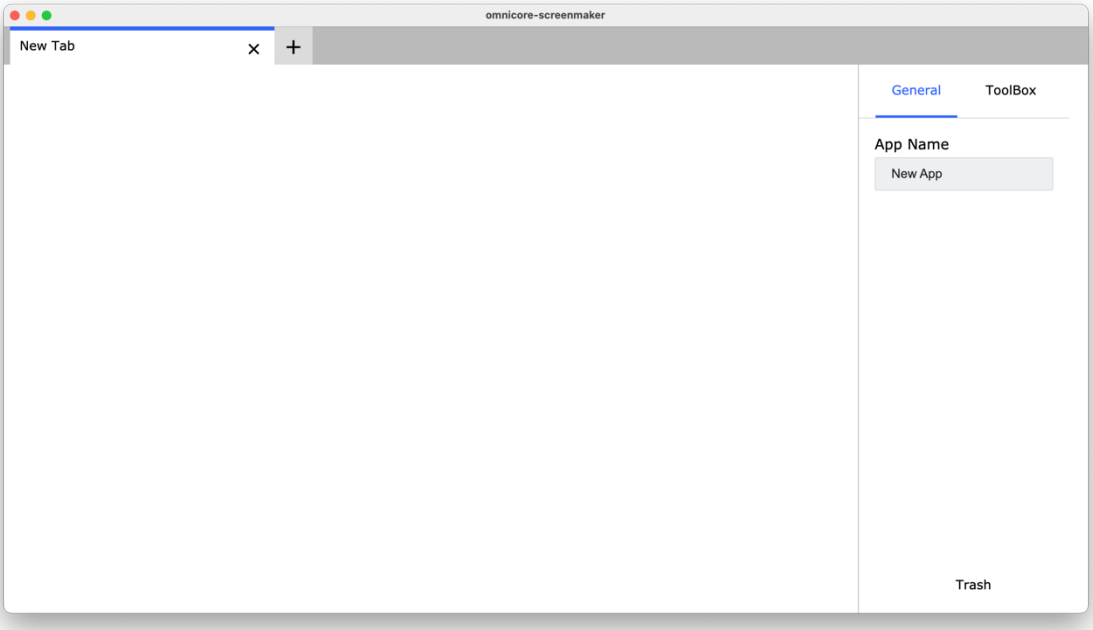
Priečinok s názvom „public“ obsahuje podpriečinok s názvom „assets“ kde sa nachádzajú knižnice RWS, všetky komponenty a ikona aplikácie z ABB SDK.



Obrázok 15: Štruktúra projektu

4.2 Layout aplikácie a jeho funkcionalita

Ako býva zvykom z mnohých editorov textových dokumentov, obrázkov, videí a obdobných typov, obsah okna aplikácie je tvorený z niekoľkých častí. V samotnom strede sa nachádza pracovná plocha, do ktorej je možné ľubovoľne vkladať komponenty. Vo vrchnej časti sa nachádza panel s kartami a na pravej časti je menu, ktoré zatiaľ ponúka dve kategórie nástrojov. Predvolená je otvorená záložka kategórie s názvom „General“. Na spodnej časti tohoto menu sa taktiež nachádza kôš.



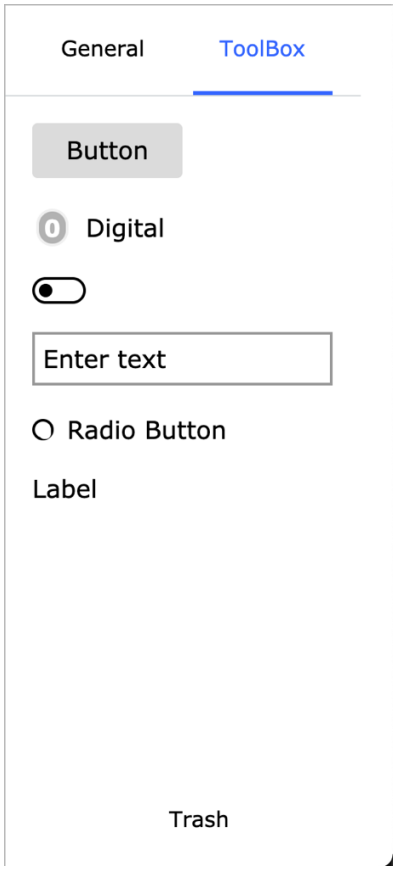
Obrázok 16: Okno aplikácie

4.2.1 Menu s nástrojmi

Ako už bolo spomenuté, toto menu využíva funkcionalitu záložiek pre zobrazenie odlišného obsahu. Na záložke General sa dá zatiaľ meniť len názov aplikácie, ktorý bude zobrazený aj v HMI na domovskej stránke.

V záložke ToolBox sa nachádzajú všetky komponenty, ktoré je možné použiť. Jednoducho stačí požadovaný komponent chytiť a potiahnuť na pracovnú plochu. Pri tejto akcii si je možné všimnúť trhaný pohyb kurzoru a komponenty. Táto funkcionalita sa nazýva „snapping“ a jej úlohou je užívateľovi pomôcť umiestniť komponenty na pracovnú plochu do pomyselnej mriežky, čo uľahčuje zarovnávanie pozície komponentov. Táto mriežka má nastavený nultý bod v ľavom hornom rohu pracovnej plochy. Jednotlivé bunky mriežky majú rozmer 40x40 pixelov.

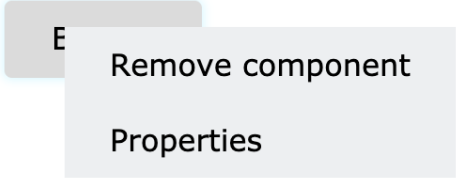
V spodnej časti sa nachádza text s nápisom „Trash“. V okolí nápisu, konkrétne na celej šírke menu a výške 200px sa nachádza priestor, ktorý slúži ako kôš. Na toto miesto je možné položiť ľubovoľný komponent umiestnený na ploche. Po položení komponentu sa komponent zničí a vymažú sa všetky jeho nastavenia.



Obrázok 17: ScreenMaker Toolbox

4.2.2 Pracovná plocha

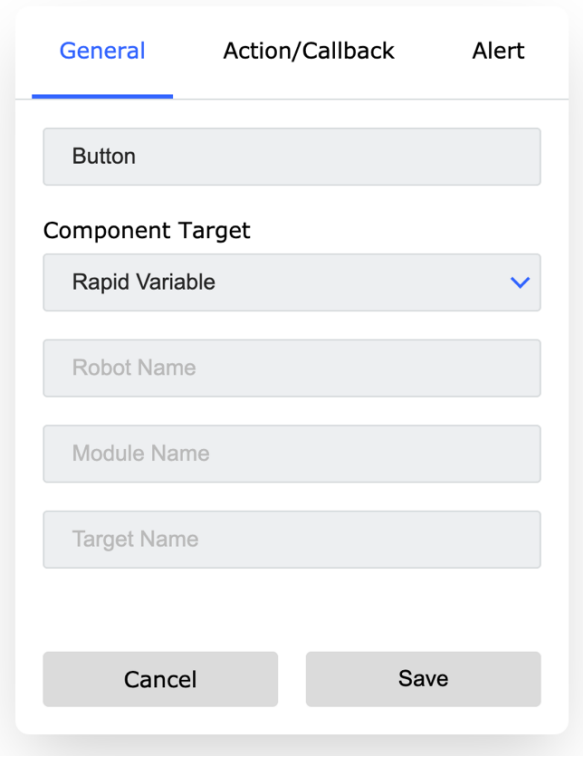
Pracovná plocha slúži pre umiestňovanie komponentov z ToolBoxu. Každý takýto komponent disponuje rôznymi akciami, ktoré sa dajú vykonávať. Napríklad kliknutím pravého tlačidla myši na komponent sa zobrazí kontextové menu s dvoma akciami. Prvá položka daný komponent vymaže. Táto akcia je ekvivalentom koša v pravom menu. Druhá položka zobrazí okno, v ktorom je možné nastaviť aktívny komponent podľa požiadaviek.



Obrázok 18: ScreenMaker kontextové menu

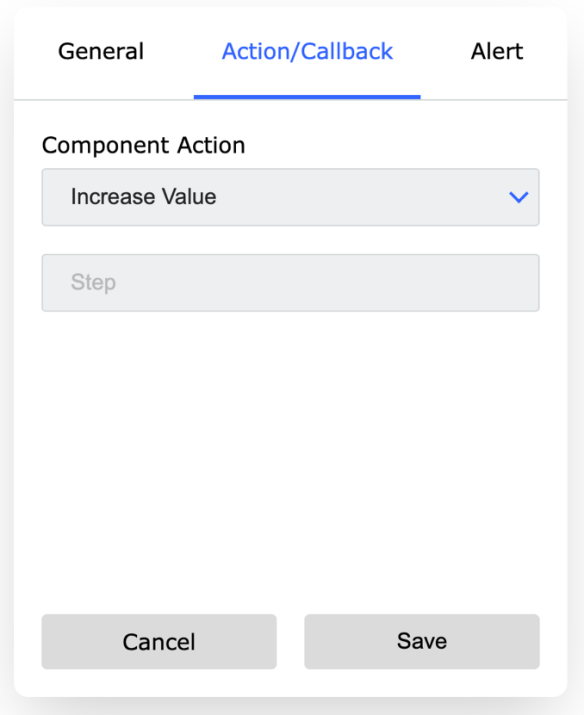
Nastavenie komponentov je možné pomocou popup okna. Obsah tohoto okna závisí na možnostiach nastavenia konkrétnych komponentov. Inými slovami, obsah sa renderuje v závislosti na možnostiach komponentov. Vo vrchnej časti okna sa nachádzajú záložky, ktoré pomáhajú rozdeliť možnosti nastavenia podľa kategórií.

Záložka s názvom General obsahuje prvky pre základné nastavenia komponentu. Prvé v záložke je textové pole, ktoré umožňuje zmenu textu komponentu, pokiaľ teda komponent takouto možnosťou disponuje. Následne sa v okne nachádza rozbaľovacie menu pre výber typu cieľu. Cieľom môže byť signál alebo rapid premenná robota. Pri výbere signálu sa zobrazí ďalšie textové pole, ktoré slúži pre zadávanie názvu signálu. V prípade, že je vybratá možnosť rapid premenná, tak sa zobrazia tri textové polia, do ktorých sa zadáva názov robota, modulu a nakoniec názov premennej.



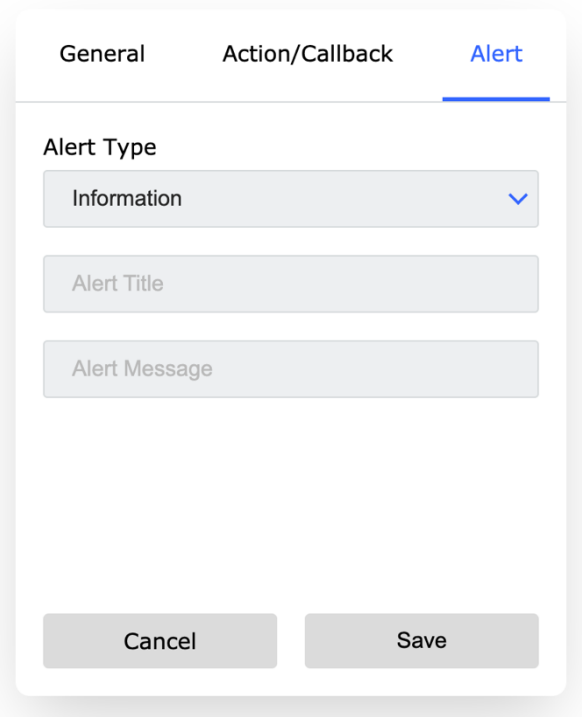
Obrázok 19: Property menu (General)

V záložke Action/Callback je umiestnené rozbaľovacie menu, v ktorom je možné priradiť akciu komponentu. Akcií je na výber niekoľko, avšak pokiaľ je vybratá akcia „increase“ alebo „decrease value“, pod rozbaľovacím menu sa zobrazí textové pole, do ktorého je možné pridať veľkosť kroku, o ktorý sa má hodnota zvyšovať alebo znižovať.



Obrázok 20: Property menu (Action/Callback)

Na tretej záložke „Alert“ sa nachádza rozbaľovacie menu, v ktorom je možné vyberať spomedzi viacerých typov upozornení. Sú to jednoducho popup okná, ktoré sa zobrazia pri interakcii s komponentom. Teda pokiaľ prebehne interakcia s komponentom, tak je užívateľ upozornený ľubovoľnou hláškou a má na výber stlačiť jedno z dvoch tlačidiel. Jedno z tlačidiel zamedzí uskutočneniu naviazanej akcie a druhé tlačidlo práve naopak, dovolí aplikácií pokračovať v danej akcii. Po vybratí typu upozornenia sa zobrazia ďalšie dve textové polia, vďaka ktorým je možné nastaviť titulok a popis upozornenia.

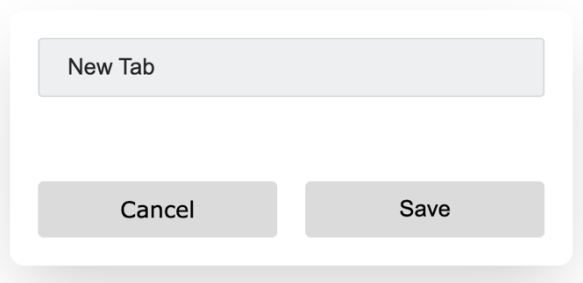


Obrázok 21: Property menu (Alert)

Po stlačení tlačidla „Save“ sa vybraté nastavenia uložia. Po znovu otvorení popup okna sa predošle uložené nastavenia načítajú a zobrazia, takže je umožnená aj úprava. Pokiaľ je stlačené tlačidlo „Cancel“, tak aktuálne nastavenia nebudú aplikované a okno sa zavrie.

4.2.3 Panel s kartami

Na tomto paneli je možné vytvárať nové karty a zatvárať staré. Karty sú naviazané na pracovnú plochu. Z užívateľského hľadiska fungujú podobne ako vo webových prehliadačoch, teda na každej karte je možné mať rozličný obsah. Celý panel s kartami je v podstate ďalší komponent, ktorý sa zobrazí aj v aplikácií v HMI. Každá karta má predvolený názov „New Tab“ ale dvojitým kliknutím na názov karty sa zobrazí popup okno, v ktorom je možné názov jednoducho zmeniť.



Obrázok 22: Zmena názvu karty

4.3 Dostupné komponenty

V aplikácii sú implementované najpoužívanejšie komponenty, ktoré umožňujú základné typy interakcie s robotom.

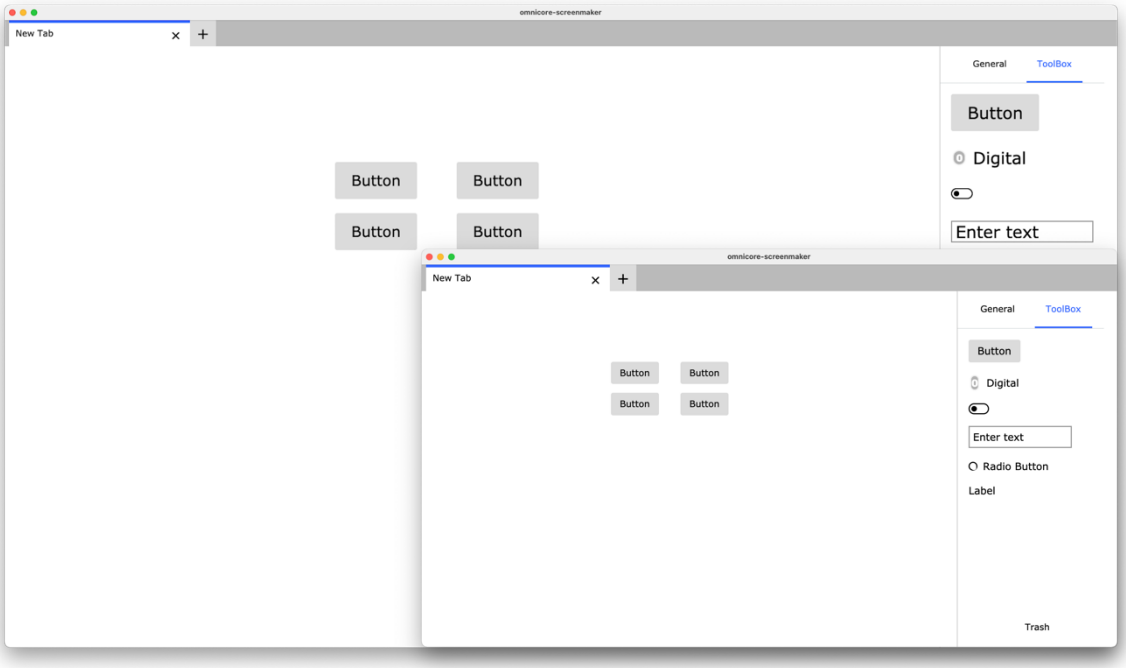
- Tlačidlo
  - komponent je možné naviazať na signály a premenné s numerickým alebo *boolean* dátovým typom
- Input
  - komponent slúži pre zmenu hodnôt premenných dátového typu *string*
  - aktuálna hodnota je vždy vypísaná vo vnútri komponentu
- Label
  - komponent obsahuje obyčajný text
  - najčastejšie používaný ako popis k inému komponentu, avšak komponent je možné naviazať aj na premennú dátového typu *string*
- Prepínač
  - slúži pre prepínanie hodnôt signálov a premenných typu *boolean*
- Digitálny prepínač
  - funkcionalita je zhodná s funkcionalitou obyčajného prepínača, avšak jeho vzhľad znázorňuje LED s ľubovoľne voliteľným popisom
  - komponent je možné naviazať len na signály
- Radio button
  - komponent je možné naviazať na premenné dátového typu *boolean* a signály
  - pre spoluprácu s ostatnými komponentami rovnakého typu sa priraďuje skupina, teda komponenty s rovnakou skupinou sú na seba naviazané

4.4 Funkcie

4.4.1 Dynamická zmena veľkosti

Jedna z kľúčových funkcionalít, ktorá zaručuje správne „vykresľovanie“ komponentov v HMI. Displej FlexPendantu má uhlopriečku 8 palcov s rozlíšením 1024x768 pixelov s pomerom strán 4:3, avšak samotná aplikácia má kvôli iným prvkom zobrazovací priestor na výšku presne 680px. [12]

Pokiaľ by mala pracovná plocha ScreenMakeru rovnaké rozmery ako práve zobrazovacia plocha v HMI, pozície komponentov na pracovnej ploche by sa bez problémov preniesli a po zostavení aplikácie by boli pozície komponentov zobrazené totožne aj v HMI. Avšak v dnešnej dobe je celkom nepredstaviteľné pracovať na tak malej ploche. Pre to má ScreenMaker užitočnú funkcionalitu, vďaka ktorej je možné ľubovoľne zväčšovať alebo zmenšovať okno aplikácie. Komponentom sa dynamicky menia rozmery v závislosti na výške a šírke okna tak, aby rozmery komponentov vždy odpovedali originálnej veľkosti a miestu na pracovnej ploche, ktoré by mali zaberať. Týmto je dosiahnuté vždy presné rozloženie komponentov aj v prípade, že okno aplikácie nezodpovedá pomeru strán HMI displeja. K zmene rozmerov komponentov dochádza vždy, keď nastane nejaká zmena v layoute a je možné, že sa rozmery pracovnej plochy zmenili.



Obrázok 23: Porovnanie rozloženia komponentov pri rôznych rozlíšeniach okna

4.4.2 Copy & Paste

Ľubovoľná interakcia s komponentom zapríčiní, že sa komponent stane aktívnym. Aktívny komponent má po svojom obvode žiaru odtieňa modrej farby. Ak je komponent aktívny, tak pri stlačení kombinácie tlačidiel na klávesnici *ctrl/cmd + c* sa komponent, uloží do clipboardu. Kombináciou tlačidiel *ctrl/cmd + v* je možné komponent vložiť na pracovnú plochu. Takto sa vytvorí na pohľad identicky nový komponent, ktorý v sebe bude mať rovnaké nastavenia ako komponent, ktorý bol skopírovaný.

Obrázok 24: Aktívny komponent

4.4.3 Komunikácia medzi hlavným procesom a oknom aplikácie

Takýto typ komunikácie je zabezpečený „ipcMain“ modulom, ktorý je súčasťou Electronu. Modul poskytuje obojstrannú asynchrónnu a synchrónnu komunikáciu, ktorá funguje na základe eventov. Na jednej strane musí byť event listener a na druhej strane spúšťač eventu. Každý event má svoj unikátny názov (kanál). Tento modul je v projekte používaný pre posielanie rôznych dát, ako napríklad informácie o komponentoch potrebných pre zostavenie aplikácie, import alebo export projektu.

```
window.api.send('build_fromRenderer', JSON.stringify(data));

ipcMain.on('build_fromRenderer', (event, components) => {
    AppBuilder.build(components);
});
```

4.4.4 Export a import projektu

Tieto funkcionality užívateľovi umožňujú uložiť aktuálny stav projektu a znovu neskôr načítať a pokračovať v práci. Tieto možnosti je možné nájsť v natívnom menu pod záložkou „File“. Pri exporte je užívateľ dotazovaný, kam si praje projekt uložiť. Po vybratí lokality a názvu súboru, sa potrebné nastavenia uložia vo forme *json* súboru. Podobne to funguje aj pri importe, kde je užívateľ takisto upozornený, avšak v tomto prípade je potrebné vybrať

súbor, z ktorého sa majú dáta načítať. Pri importe sa aktuálne otvorený projekt nahradzuje novým.

4.4.5 Zostavenie aplikácie

Zostavenie aplikácie je taktiež možné z natívneho menu v záložke „File“. Užívateľ je následne upozornený, kam sa má aplikácia zostaviť. Po vybratí lokality sa proces zostavenia začína.

Tento proces je riešený na báze šablón. Každý komponent má svoju vlastnú šablónu, v ktorej sa porovnávajú nastavenia komponentu a podľa nich sa volajú ďalšie funkcie, ktoré vložia dáta do šablóny. Šablóny sa medzi sebou dajú ľubovoľne kombinovať podľa požiadaviek.

```
const buttonInit = (component) => {
  const template = /*javascript*/ `
        ${component.id} = new FPComponents.Button_A();
        ${component.id}.attachToId("${component.id}");
        ${component.id}.text = "${component.text}";
        ${component.id}.onclick = async function () {
            ${
              component.alertType
                ? Config.alertPopup(component)
                : (component.action === "increase-value"
                    ? Config.increaseValue(component)
                    : "",
                  component.action === "decrease-value"
                    ? Config.decreaseValue(component)
                    : "",
                  component.action === "set-value"
                    ? Config.setValue(component)
                    : "",
                  component.action === "toggle-signal"
                    ? Config.toggleSignal(component)
                    : "")
            }
        };
        ${
          component.targetType === "signal"
            ? Config.subscribeSignal(component)
            : ""
        }
        ${
          component.targetType === "rapid"
            ? Config.subscribeVariable(component)
            : ""
        }
    `;
  return template;
};
```

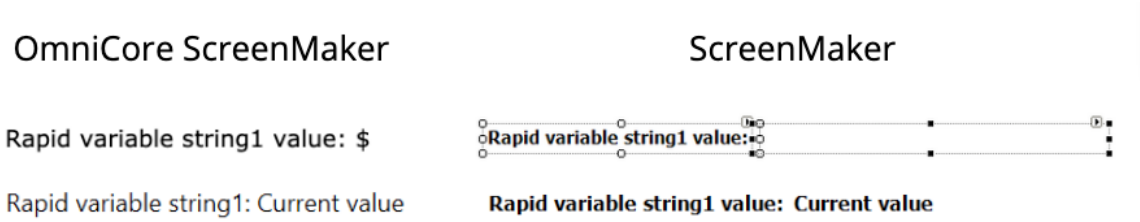
Takže každý jeden komponent z pracovnej plochy sa zoberie a pomocou šablón je zostavený kód, ktorý zodpovedá požadovanej funkcionalite. Takéto bloky sa pomaly zbierajú a nakoniec sa vložia do poslednej šablóny, do súboru app.js. Podobne zostavený je aj index.html súbor.

Ostatné časti aplikácie ako rws knižnica, samotné komponenty atď., sú do konečného priečinku len prekopírované z priečinku assets.

5 POROVNANIE APLIKÁCIÍ

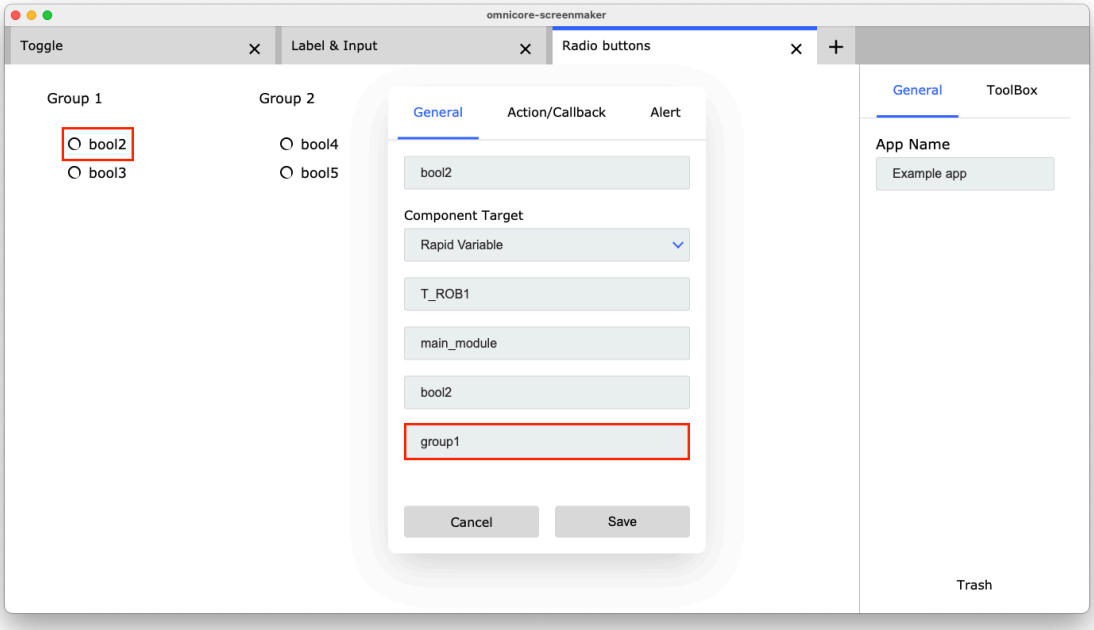
Pre porovnanie boli vytvorené dve podobné aplikácie. Ako prvá bola vytvorená aplikácia pomocou nového nástroja OmniCore ScreenMaker a následne pre tento istý účel bola použitá aj aplikácia ScreenMaker pre kontrólery typu IRC5. Každá aplikácia sa skladá z troch kariet, ktoré obsahujú niekoľko základných komponentov. Vďaka novým komponentom, ktoré OmniCore SDK poskytuje, aplikácia pôsobí rozmanitejšie a celkovo vizuálne krajšie. Proces vytvárania komponentov s pomocou OmniCore ScreenMaker je oveľa intuitívnejší. Poskytuje nové funkcie, ktoré zjednodušujú a urýchľujú prácu.

Jedným z rozdielov je prepájanie komponenty label s rapid premennými. Pokiaľ je potrebné použiť label s popisom a niekde vedľa popisu vypísať hodnotu rapid premennej, tak v prípade aplikácie ScreenMaker je potrebné priradiť dve label komponenty, kde jedna z nich obsahuje popis a druhá je prepojená s rapid premennou. Taktiež je potrebné predpokladať veľkosť plochy, ktorú bude hodnota premennej zaberať a podľa toho zmeniť veľkosť komponenty tak, aby bola hodnota správne zobrazená. V prípade aplikácie OmniCore ScreenMaker pre túto funkcionalitu stačí jeden label komponent. Komponent je len potrebné prepojiť s rapid premennou, z ktorej bude brať hodnotu a v popise na určité miesto vložiť znak „$“. V HMI sa tento znak automaticky nahradzuje hodnotou premennej.



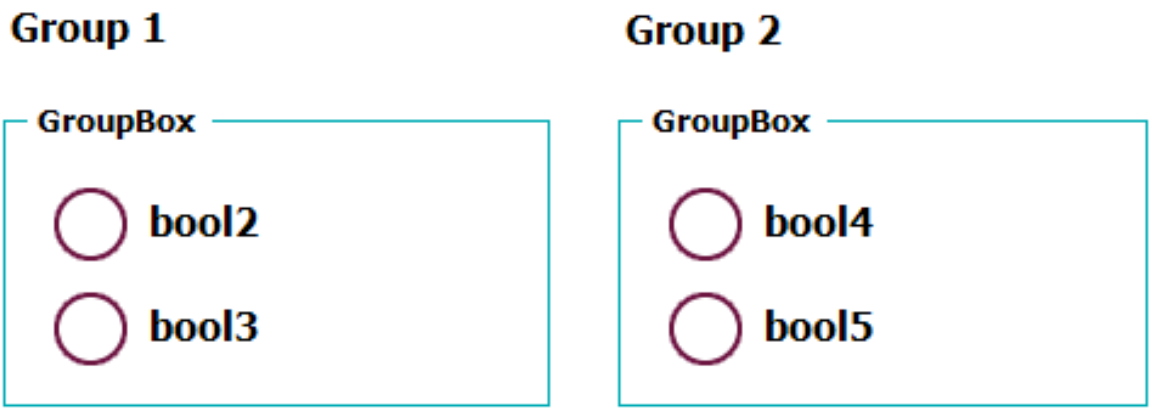
Obrázok 25: Porovnanie funkcionality label

Radio button komponenty fungujú podobne v obidvoch nástrojoch. Zásadný rozdiel je však v prípade, že je potrebné vytvoriť viacero skupín. V ukážkových aplikáciách sa nachádzajú štyri radio button komponenty. Tieto komponenty sú rozdelené do dvoch skupín. V aplikácií OmniCore ScreenMaker stačí každej komponente priradiť skupinu v property menu.



Obrázok 26: OmniCore ScreenMaker - nastavenie skupiny

V aplikácii ScreenMaker je pre dosiahnutie rovnakého výsledku potrebné vynaložiť o niečo viac práce. V ToolBoxe sa pre tento účel nachádza komponent GroupBox. Komponent vytvorí skupinu a všetky komponenty umiestnené vo vnútri, patria do tejto skupiny. Takýto postup vytvárania skupín obmedzuje užívateľa v rozmiestňovaní komponentov. V aplikácii OmniCore ScreenMaker je možné nastaviť skupinu aj v prípade, že sa radio button komponenty nachádzajú na iných kartách.



Obrázok 27: ScreenMaker - nastavenie skupiny

ZÁVER

Cieľom tejto práce bolo vytvoriť nový nástroj pre vývoj aplikácií pre HMI panely od firmy ABB. Pôvodná aplikácia ScreenMaker nepodporuje novú radu kontrolérov rodiny OmniCore, a preto jedinou cestou pre vytvorenie funkčnej aplikácie, ktorá dokáže s novou radou kontrolérov spolupracovať, je použiť oficiálny SDK od výrobcu.

V teoretickej časti sú popísané základné webové technológie, na ktorých je projekt stavaný. Následne sú rozobraté frameworky, ktoré tieto webové technológie obohacujú o novú funkcionalitu, a bez ktorých by len ťažko bolo možné dosiahnuť vyhovujúceho výsledku. OmniCore ScreenMaker je inšpirovaný najmä aplikáciou ScreenMaker pre IRC5 kontroléry, preto je popísaná jej základná funkcionalita a taktiež priblížená práca s ňou. Pre lepšie pochopenie riešenej problematiky je porovnaný aj proces vytvorenia aplikácie pre kontrolér typu IRC5 s jediným možným postupom pre vytvorenie aplikácie pre kontroléry OmniCore.

Praktická časť sa viac sústredí na samotný produkt tejto práce, kde sa striedajú pohľady vývojárskeho a užívateľského hľadiska. Popísaná je funkcionalita aplikácie, ktorá zachádza až do vysvetľovania na samotnom zdrojovom kóde. Neskôr boli vytvorené dve ukážkové aplikácie, každá iným nástrojom a následne bolo poukázané na zásadné rozdiely v procese vytvárania aplikácie pre HMI a porovnanie výsledku.

SEZNAM POUŽITÉ LITERATURY

[1] What Is Electron.js? [online]. Gliwice (Poľsko): Brainhub, 2020 [cit. 2022-02-23]. Dostupné z: https://brainhub.eu/library/what-is-electron-js/

[2] What Is Electron and Why Is It So Polarizing? [online]. USA: Kirupa, 2020 [cit. 2022-02-23]. Dostupné z: https://www.kirupa.com/apps/what_is_electron.htm

[3] 10 Most Popular Electron Apps of 2019 [online]. Dánsko: Wiredelta, 2019 [cit. 2022-02-24]. Dostupné z: https://wiredelta.com/10-most-popular-electron-apps-2019/

[4] *What is Vue.js, and Why is it Cool?* [online]. Sunnyvale (CA): Linuxhint, 2021 [cit. 2022-03-02]. Dostupné z: https://linuxhint.com/about_vue_js/

[5] *7 Reasons Why VueJS Is So Popular* [online]. Houston (TX): KofiGroup, 2021 [cit. 2022-03-02]. Dostupné z: https://www.kofi-group.com/7-reasons-why-vuejs-is-so-popular/

[6] *Application manual ScreenMaker* [online]. Zürich (Švajčiarsko): ABB, 2009 [cit. 2022-03-10]. Dostupné z: https://library.e.abb.com/public/b62731a5b2d5528bc125766d003a6228/Application_Manual_ScreenMaker.pdf

[7] *A Beginner's Guide to Sass* [online]. New York: codecademy, 2021 [cit. 2022-03-20]. Dostupné z: https://www.codecademy.com/resources/blog/what-is-sass/

[8] *DUCKETT, Jon. HTML and CSS: Design and Build Websites.* Indianapolis: John Wiley, 2011. ISBN 978-1-118-00818-8.

[9] *HAVERBEKE, Marijn. Eloquent Javascript: a modern introduction to programming.* San Francisco: No Starch Press, 2011. ISBN 978-1-59327-282-1.

[10] *Omnicore SDK: Robot Web Services.* ABB Deceloper Center [online]. ABB, 2021. Dostupné z: https://developercenter.robotstudio.com/api/RWS

[11] *Omnicore SDK: Application Manual.* ABB Deceloper Center [online]. ABB, 2021. Dostupné z: https://developercenter.robotstudio.com/omnicore-sdk

[12] *Omnicore SDK: Design Guideline.* ABB Deceloper Center [online]. ABB, 2021. Dostupné z: https://developercenter.robotstudio.com/omnicore-sdk

[13] *OmniCore™ Controller family* [online]. Västerås (Švédsko): ABB, 2022 [cit. 2022-03-22]. Dostupné z: https://new.abb.com/products/robotics/controllers/omnicore

[14] *What is the DOM? Document Object Model Meaning in JavaScript* [online]. USA: freeCodeCamp, 2021 [cit. 2022-04-16]. Dostupné z: https://www.freecodecamp.org/news/what-is-the-dom-document-object-model-meaning-in-javascript/

[15] *File:Electron Software Framework Logo.svg* [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2022-04-17]. Dostupné z: https://commons.wikimedia.org/wiki/File:Electron_Software_Framework_Logo.svg

[16] *File:Vue.js Logo 2.svg* [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2022-04-17]. Dostupné z: https://commons.wikimedia.org/wiki/File:Vue.js_Logo_2.svg

[17] *File:Sass Logo Color.svg* [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2022-04-17]. Dostupné z: https://commons.wikimedia.org/wiki/File:Sass_Logo_Color.svg

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

RWS Robot Web Services REST API

UI User interface

SDK Software development kit

DOM Document object model

LED Light-emitting diode

ZOZNAM OBRÁZKOV

SEZNAM PŘÍLOH

P1 Github repozitár obsahujúci kompletný projekt

P2 CD s bakalárskou prácou a manuálom pre vytvorenú aplikáciu

PŘÍLOHA P I: GITHUB REPOZITÁR OBSAHUJÚCI KOMPLETNÝ PROJEK

Github repozitár je možné nájsť na platforme github.

https://github.com/kpilat/omnicore-screenmaker.git