

# Generátory GUI pro mobilní a embedded aplikace

Bc. Václav Peredarjuk

---

Diplomová práce  
2014



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2013/2014

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Václav Peredarjuk**

Osobní číslo: **A12393**

Studijní program: **N3902 Inženýrská informatika**

Studijní obor: **Informační technologie**

Forma studia: **prezenční**

Téma práce: **Generátory GUI pro mobilní a embedded aplikace**

Zásady pro vypracování:

1. Analyzujte funkční i nefunkční požadavky na generátor GUI (Graphical User Interface) pro mobilní a embedded aplikace. Při analýze uvažujte jak návrh GUI modelu (mockup), tak možnost generování kódu pro cílovou platformu.
2. Pro pochopení všech funkčních požadavků implementujte pomocí technologie DHTML vzorový model GUI pro libovolnou embedded nebo mobilní aplikaci a analyzujte možnosti generování kódu pro cílovou embedded nebo mobilní platformu.
3. Prostudujte existující generátory grafických uživatelských rozhraní a srovnajte jejich funkce s požadavky dle bodu 1.
4. Pokud nenajdete žádný existující generátor, který vyhovuje větší části požadavků, proveďte analýzu možností implementace vlastního generátoru, včetně návrhu UML modelu a popřípadě i vlastní implementaci.
5. Pokud najdete vyhovující generátor, demonstруйте jeho možnosti na příkladu aplikace dle bodu 2.
6. Vlastní zdrojové kódy publikujte pod licencí GPLv2 nebo vyšší.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **BLIŽŇÁK, Michal, Tomáš DULÍK a Vladimír VAŠEK. WxShapeFramework: An Easy Way for Diagrams Manipulation in C++ Applications [online]. WSEAS TRANSACTIONS on COMPUTERS: WSEAS Press, 2010, roč. 9, č. 1 [cit. 2014-01-23]. ISSN 1109-2750. Dostupné z: <http://www.worldses.org/journals/computers/computers-2010.htm>**
2. **BLIŽŇÁK, Michal, Tomáš DULÍK a Roman JAŠEK. Production-Ready Source Code Round-Trip Engineering [online]. INTERNATIONAL JOURNAL OF COMPUTERS: NAUN Press, 2012 [cit. 2014-01-23]. ISSN 1998-4308. Dostupné z: <http://www.naun.org/wseas/cms.action?id=3036>**
3. **BLIŽŇÁK, Michal, Tomáš DULÍK, Roman JAŠEK a Pavel VAŘACHA. Optimized Production-Ready Source Code Generation Based on UML [online]. INTERNATIONAL JOURNAL OF SYSTEMS APPLICATIONS, ENGINEERING & DEVELOPMENT: NAUN Press, 2013, roč. 7, č. 1 [cit. 2014-01-23]. ISSN 2074-1308. Dostupné z: <http://www.naun.org/main/UPress/saed/16-498.pdf>**
4. **PRATA, Stephen. Mistrovství v C++. 3. aktualiz. vyd. Překlad Boris Sokol. Brno: Computer Press, 2007, 1119 s. ISBN 978-80-251-1749-1.**
5. **CORMEN, Thomas H. Introduction to algorithms. 3rd ed. Cambridge: MIT Press, 2009, xix, 1292 s. ISBN 978-0-262-03384-8.**
6. **KANISOVÁ, Hana a Miroslav MÜLLER. UML srozumitelně. 2. aktualiz. vyd. Brno: Computer Press, 2006, 176 s. ISBN 80-251-1083-4.**
7. **SMART, Julian. Cross-platform GUI programming with wxWidgets. Upper Saddle River: Prentice-Hall, 2006, xxxv, 700 s. ISBN 01-314-7381-6.**

Vedoucí diplomové práce:

**Ing. Michal Bližňák, Ph.D.**

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

**21. února 2014**

Termín odevzdání diplomové práce:

**20. května 2014**

Ve Zlíně dne 21. února 2014

prof. Ing. Vladimír Vašek, CSc.

*děkan*



doc. Mgr. Roman Jašek, Ph.D.

*ředitel ústavu*

## **ABSTRAKT**

Úkolem mé diplomové práce bylo prozkoumat GUI designery a generátory pro mobilní a embedded systémy. V teoretické části práce nastiňuji vznik GUI v době počítačů. Dále procházím historii vývoje GUI jednotlivých operačních systémů po současnost. V praktické části práce jsem zkoumal mnoho GUI softwarů a nejlepší vybrané jsem detailně popsal a demonstroval jejich možnosti na demu. Součástí práce je také individuální návrh GUI ovladače pro firmu Santech, který slouží jako předloha pro jeho realizaci. V závěru práce jsem zhodnotil výhody a nevýhody jednotlivých programů a mockupů.

Klíčová slova: Grafické uživatelské prostředí, modelář, návrhář, generátor kódu, vestavěný systém

## **ABSTRACT**

The goal of my magister thesis was to explore GUI designers and generator for mobile and embedded systems. In theoretical part I'm outlining the rise of GUI at computer times. Also I'm browsing through history of making GUI for operating systems. In practical part I've studied many GUI making software from which I've chosen few good ones that are described in detail and their functionality is shown in demo app. Part of my thesis is individual GUI draft for realization of Santech's control. In the thesis conclusion the pros and cons of individual softwares and mockups are evaluated.

Keywords: GUI, mockup, designer, code generator, embedded system

Zde bych chtěl tímto poděkovat svému vedoucímu diplomové práce panu Ing. Michalu Bližňákovi Ph.D., za rady a trpělivost a panu Ing. Tomáši Dulíkovi Ph.D. za přínosné informace.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/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 diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/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 – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/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 diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/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 diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD</b> .....	<b>9</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>10</b>
<b>1 VYSVĚTLENÍ POJMU GUI</b> .....	<b>11</b>
<b>2 EMBEDDED SYSTÉM</b> .....	<b>12</b>
2.1 CHARAKTERISTIKA EMBEDDED SYSTÉMU .....	12
2.2 HISTORIE VÝVOJE EMBEDDED SYSTÉMŮ .....	14
<b>3 HISTORIE VÝVOJE GUI</b> .....	<b>16</b>
<b>4 DŮVODY ZKOUMÁNÍ GUI GENERÁTORŮ</b> .....	<b>22</b>
<b>II PRAKTICKÁ ČÁST</b> .....	<b>23</b>
<b>5 POUŽITÝ SOFTWARE</b> .....	<b>24</b>
5.1 VMWARE PLAYER .....	24
5.2 GIMP VERZE 2.8.....	25
5.3 BRACKETS SPRINT.....	26
<b>6 ANALÝZA FUNKČNÍCH A NEFUNKČNÍCH POŽADAVKŮ</b> .....	<b>27</b>
6.1 GUI NÁVRH.....	28
6.2 GUI FUNKCIONALITA.....	31
6.3 SHRNUTÍ FUNKČNÍCH A NEFUNKČNÍCH POŽADAVKŮ .....	32
6.3.1 Funkční požadavky.....	32
6.3.2 Nefunkční požadavky.....	32
<b>7 INDIVIDUÁLNÍ ŘEŠENÍ POMOCÍ DHTML</b> .....	<b>33</b>
7.1 OBRAZOVKY .....	33
7.2 DIAGRAMY AKTIVIT.....	34
7.2.1 Funkce hydro.....	35
7.2.2 Funkce ovládání teploty .....	36
7.2.3 Funkce vybrání barvy .....	37
7.3 ZMĚNA OBRAZOVEK .....	38
7.4 DEFINICE CHOVÁNÍ APLIKACE .....	39
7.5 SPECIFICKÉ FUNKCE.....	39
7.6 VÝHODY A NEVÝHODY INDIVIDUÁLNÍHO ŘEŠENÍ.....	41
<b>8 KOMPONENTY PRO VÝVOJ EMBEDDED GUI</b> .....	<b>42</b>
<b>9 NALEZENÉ GUI DESIGNERY A GENERÁTORY</b> .....	<b>45</b>

9.1	FORE UI .....	45
9.2	PROTO.IO.....	46
9.3	GEM STUDIO PRO .....	47
9.4	CRANK STORYBOARD SUITE 3.2.....	48
<b>10</b>	<b>DEMONSTRACE APLIKACE V GUI GENERÁTORECH.....</b>	<b>51</b>
10.1	FORE UI .....	51
10.2	PROTO.IO.....	56
10.3	GEM STUDIO PRO .....	59
10.4	CRANK STORYBOARD SUITE 3.2.....	60
<b>11</b>	<b>SROVNÁNÍ GUI SOFTWARE .....</b>	<b>66</b>
	<b>ZÁVĚR .....</b>	<b>70</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>71</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>75</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>76</b>
	<b>SEZNAM TABULEK.....</b>	<b>78</b>
	<b>SEZNAM PŘÍLOH.....</b>	<b>79</b>

## ÚVOD

Pojem GUI je v dnešním světě samozřejmostí. Někteří lidé neví co se pod pojmem skrývá, i když ho používají téměř denně. Základní kámen všech GUI aplikací můžeme nalézt na PC ve formě desktopového operačního systému. Každý jedinec pojem GUI definuje jinak, avšak ve své podstatě se GUI skládá ze stejných prvků. Je to spojení textové a obrázkové formy. Obrázková forma je přítomna pro jednoduché a rychlé orientování v prostředí. Bývá reprezentována ikonami. V uživatelském prostředí platí pravidlo „obrázek je lepší než 1000 slov“ a tak jsou některé často používané texty nahrazeny těmito ikonami. Uživateli aplikace tak šetří drahocenný čas a umožňuje lepší orientaci.

GUI aplikace se dělí na několik typů s ohledem na jejich použití. GUI pro embedded systémy jsou velikým lákadlem a jeho vývoj neustále roste. Jako externí GUI lze považovat např. přístrojovou desku automobilu, ovládání LCD displeje navigace, ovládací panel pračky, panely automatů a mnohé další. Embedded GUI aplikace často zobrazují stavy reálných fyzikálních jevů, jako otáčky motoru, rychlost a další a umožňují tak sledovat a analyzovat funkčnost nebo chyby strojů. Ve své práci se zaměřuji zejména na GUI designery a generátory kódu pro embedded a mobilní systémy. Celá práce obsahuje seznámení s tím, jak GUI aplikace pro embedded a mobilní systémy fungují a jaké jsou její možnosti využití. V praktické části vytvořím vzorovou GUI aplikaci pomocí technologie DHTML a pomocí ní budu analyzovat jednotlivé požadavky z hlediska tvorby a použitelnosti aplikace pro jednotlivé typy systémů. V rámci analyzování vyhledám vytvořené softwary GUI zaměřující se na tvorbu těchto aplikací a vyberu nevhodnější kandidáty. Ve vybraných softwarech se budu snažit demonstrovat možnosti a vlastnosti aplikace na již vytvořeném demu.

## I. TEORETICKÁ ČÁST

## 1 VYSVĚTLENÍ POJMU GUI

Je zkratka z anglického slova „Graphics User Interface“ česky grafické uživatelské rozhraní. Jeho cílem je zjednodušit a zpřehlednit uživateli práci s programy namísto konzolových příkazů. Všechna GUI obsahují grafické prvky jako dialogová okna, tlačítka, ikony, menu nebo posuvníky namísto textových příkazů. [10]

GUI prvky jsou dostupná přes polohovací zařízení, jako je myš, touchpad nebo dotykové pero. GUI se v dnešní době vyskytuje všude – počítače, přenosné zařízení, MP3,MP4 přehrávače, domácí spotřebiče, kancelářské spotřebiče nebo průmyslové aplikace a zařízení. GUI představuje akce a informace, které jsou uživateli zobrazeny jako grafické ikony a vizuální indikátory. Různé akce jsou spuštěny po kontaktu s těmito grafickými prvky (tlačítka). Pomocí GUI lze provádět akce typu otevírání souborů, mazání souborů a mnohé další. S GUI se každý setkává denně v podobě operačního systému Windows, Mac OSX nebo Linux. [19]

## 2 EMBEDDED SYSTÉM

Embedded systém v češtině znamená „vestavěný systém“ nebo „zabudovaný systém“. Embedded systémy patří k nejrozšířenější variantě užívání počítačových systémů. Tyto systémy poskytují větším systémům, jichž jsou součástí, potřebnou inteligenci. Jedním z požadavků na tyto systémy je, že musí být tzv. „autonomní“ čili schopný plnit své funkce bez zásahu člověka v průběhu poměrně dlouhého časového intervalu. Jedná se většinou o jednoúčelový systém, ve kterém je řídicí počítač nebo mikročip zabudován do zařízení. Od osobních počítačů se tento systém liší z hlediska použití, embedded systém plní jednu požadovanou funkci na rozdíl od klasického stolního počítače. Embedded systém je tak snáze optimalizován pro konkrétní aplikaci a tudíž lze razantně snížit cenu výsledného výrobku. Embedded systémy bývají často sériově vyráběny a úspora bývá často několika násobně navýšena. [22]

### 2.1 Charakteristika embedded systému

S embedded systémy se dnes setkáváme běžně. Mezi nejznámější embedded systémy patří bankomaty, herní konzole, mobilní telefony, tiskárny, modemy, scannery, PDA, různé zdravotnické přístroje a domácí spotřebiče jako mikrovlnné trouby, televizory, pračky, navigační systémy letecké nebo cestovní navigace, telefonní ústředny a mnohé další. PDA, mobilní telefony (smartphone) jsou často označovány také jako embedded systémy vzhledem k vlastnostem hardware, i když po softwarové stránce nabízejí možnost rozšíření funkcionality, které se blíží spíše použití stolního PC. [11]



Obr.č. 1 – Ukázka embedded systému [26]

U systémů vyráběných ve velkých sériích, jako například MP3 přehrávačů nebo mobilních telefonů, je většinou jedním z primárních cílů ve fázi návrhu nízká cena a možnost masivní výroby. Software, který řídí embedded systémy je často označován jako firmware a bývá uložen v paměti ROM nebo Flash a tím se liší od stolního PC, kde jsou programy uloženy na pevném disku. Z toho vyplývá i to, že software psaný pro embedded systém musí být, co se týče velikosti, co nejmenší a musí obsahovat jen to nejnútnejší pro správnou funkci systému. [12]

Řídicí systémy jsou vestavěny do zařízení, od nichž se očekává, že budou schopny pracovat léta bez chyb a v některých případech schopny zotavit se z poruchy. Proto bývají programy většinou vyvíjeny a testovány pečlivěji než programy pro osobní počítače a v návrhu se nepoužívají nespolehlivé mechanické součástky jako disky, přepínače a tlačítka. Zotavení z poruchy může využít například techniky watchdog timeru, který resetuje počítač, pokud program po jistou dobu neupozornil watchdog, že je v pořádku. [11]



Obr.č. 2 – Ukázka GUI embedded systému [27]

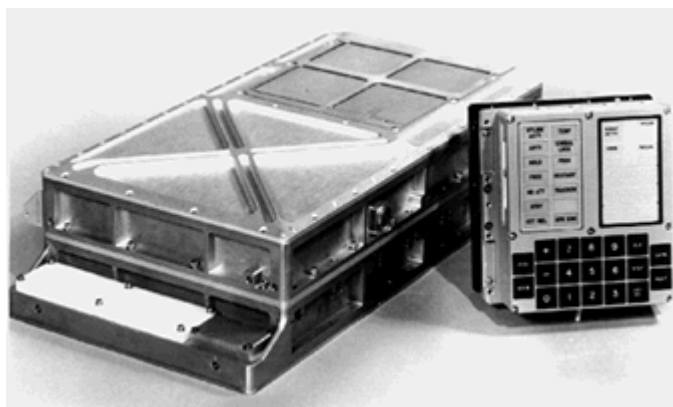
Uživatelská rozhraní u vestavěných systémů jsou velice variabilní a liší se od nulového uživatelského rozhraní až po plné rozhraní podobné systému stolního PC. Systémy mohou obsahovat pouze ovládací prvky jako tlačítka nebo i zobrazovací prvky v podobě displeje různé velikosti. Při použití větších displejů bývají použity dotykové displeje nebo displeje s ovládacími klávesami (soft buttons), které poskytují dostatečnou flexibilitu při

minimalizaci použitého prostoru. Výhodou tohoto řešení je, že funkce tlačítek se změni nápisem na displeji a také přirozené ovládání. V dnešní době mobilní telefony a PDA již disponují velkým dotykovým displejem, který zcela nahradil hardwarovou klávesnici. Rozvoj webu dává tvůrcům vestavěných zařízení další zcela novou možnost ovládání pomocí webové stránky dostupné prostřednictvím síťového připojení. Toto řešení eliminuje potřebu složitého displeje, a přesto poskytuje možnost složitého nastavení a zobrazení v případě potřeby na vzdáleném počítači. Toto řešení je úspěšně používáno pro ovládání vzdálených pevně instalovaných zařízení. [22]

## 2.2 Historie vývoje embedded systémů

Historie embedded systému se datuje kolem let 1930-1940, kdy počítače byly předurčeny pro vykonávání jedné specifické úlohy, byly však po fyzické stránce moc velké a po finanční stránce moc drahé. Postupem času se však koncept programovatelných kontrolerů vyvinul z tradičních elektromechanických zařízení k využití výpočetní techniky. [16]

Prvním z moderních vestavěných systémů byl systém „Apollo Guidance Computer“, který byl postaven Charlesem Starkem Draperem v laboratoři univerzity MIT. Na počátku projektu byl Apollo guidance computer považován jako nejrizikovější položka, protože využil nově vytvořené monolitické integrované obvody z důvodu redukce velikosti a váhy.



Obr.č. 3 – Apollo guidance computer [28]

Brzo masově vyráběný embedded systém byl „Autonetics D-17“, který byl vyvinut v roce 1961 a sloužil jako podpůrný modul pro „Minuteman“ raketu. Tento vestavěný systém byl postaven z tranzistorové logiky a disponoval pevným diskem pro hlavní paměť. Když v roce 1966 byl vyvinut Minuteman II, D-17 byl nahrazen novým počítačem s vysokým počtem integrovaných obvodů. Tento program způsobil snížení nákladů na jedno 4-

vstupové hradlo z 1000\$ na 3\$ za kus a umožnil jejich použití v komerčních produktech. Tyto události způsobily výrazný vzestup výpočetního výkonu a funkčnosti. [21]

První mikroprocesor, který byl navržen pro embedded systémy byl INTEL 4004. Byl využit zejména pro kalkulačky a jiné menší systémy, ale stále vyžadoval mnoho paměťových a podpůrných čipů. Ceny mikroprocesorů a mikrokontrolerů spadly, protože se stalo schůdným nahradit drahé analogové komponenty jako třeba potenciometry nebo měnitelné kondenzátory s tlačítky nahoru/dolů. V roce 1974 byl uveden na trh 8-bitový mikroprocesor Intel 8008 a 8080 a zároveň uvedla firma Motorola na trh mikroprocesor MC6800. [16]

V polovině osmdesátých let většina běžných součástí dříve vnějšího systému byla integrována do stejného čipu jako procesor a tato moderní podoba mikrokontroléru dovolila široké využití, které do konce desetiletí bylo spíše pravidlem než výjimkou téměř všech elektronických zařízení. Okolo roku 1989 byly vyvinuty tiskárny, které se řadí mezi embedded systémy a zanedlouho po nich přišly na trh mobilní telefony, které navíc komunikovaly bezdrátově. Pro sekci embedded systémů měl velký vliv na rozvoj vyvinutí dotykového displeje, který můžeme najít v mnoha zařízeních – mobilní telefony, PDA, ovládací prvky domácích spotřebičů, bankomaty atd. [25]

V dnešní době se embedded systémy vyskytují všude, můžeme je najít jako řídicí elektroniku novějších aut, nemocniční zařízení a přístroje, hodinky, mikroskopy a mnohé další. [25]

### 3 HISTORIE VÝVOJE GUI

Historie GUI se datuje do roku, kdy ještě neexistovaly ani počítače. Myšlenky GUI počítače byly ještě dlouho předtím, než byly vynalezeny technologie na její realizaci. Jako první člověk popisující tyto metody byl Vannevar Bush. Okolo roku 1930 poprvé publikoval popis zařízení zvané „Memex“, který vypadal jako stůl s dvěma dotykovými displeji a klávesnicí. To by umožňovalo přistupovat k lidským znalostem pomocí připojení podobné tomu, kterým se vyznačují dnešní hypertextové odkazy. V této době však nebyl počítač vynalezen a tak nebyl způsob jak toto zařízení uvést do práce. Proto Bushovy nápady nebyly zkoumány. [9]

V roce 1945 se opět ozval Vannevar Bush, který publikoval článek s názvem „As we may think“ v magazínu „Atlantic Monthly“. Tento článek byl motivační nástroj pro mladého Douglase Englebart, který chtěl vyzkoušet postavit takovýto stroj. V roce 1948 dokončil studium v elektrotechnickém oboru a našel si práci v NACA institutu, který je předchůdce NASA. Bushova esej mu nedala spát a pořád přemýšlel o tom, jak by tuto myšlenku uvedl v realizaci. Proto začal hledat investora, který by investoval do jeho vizí. V roce 1955 si udělal doktorát a dostal pozici u Stanford Research Institute, kde získal mnoho patentů na zmenšování počítačových komponent. V roce 1959 již měl velký respekt, a proto dostal grant od leectva spojených států na vývoj GUI zařízení. V roce 1962 Douglas Englebart publikoval esej s názvem „Rozšiřování lidského intelektu“. V práci argumentoval, že digitální počítače by mohly poskytnout nejrychlejší způsob jak zobrazit složitou situaci a vyvodit její řešení. Jako první hypotetický příklad byl návrh budovy pomocí architekta, který se měl podobat dnešním moderním grafickým CAD programům. Douglas a jeho spolupracovníci po mnoho let rozvíjeli jeho nápady a technologii, které nakonec vyústili ve veřejnou demonstraci před více než tisícem počítačových odborníků v roce 1968. Englebartova demonstrace byla multimediálně fantastická. Nový systém byl nazván NLS nebo taky oN-Line systém, protože byl připojen k několika počítačům. [10]



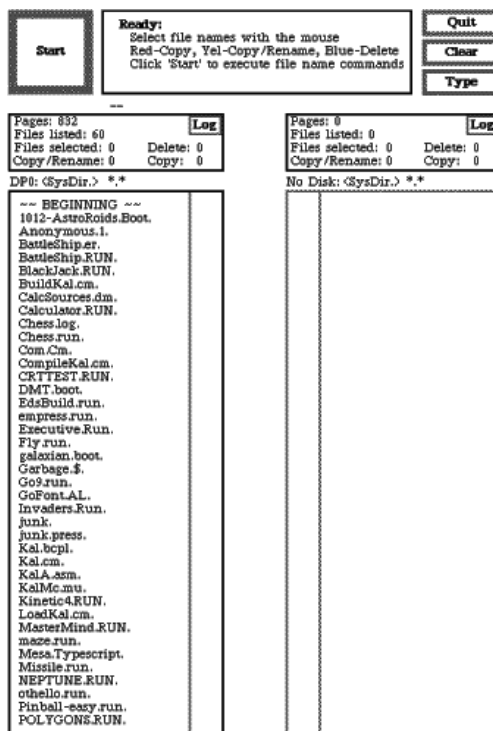
Obr.č. 4 – oN-Line systém, klávesnice, myš [23]

Systém displeje byl založen na vektorové grafice a byl schopen zobrazovat text i jednoduché čáry. Douglasovy ruce obsluhovaly tři vstupní zařízení, klasický psací stroj ve stylu klávesnice, pěti tlačítkovou klávesnici a malý box se třemi tlačítky na vrchu připojené k počítači pomocí dlouhého kabelu. Jednalo se o myš, kterou vynalezl Douglas a postavil jeden z jeho inženýrů. Nikdo neví, kdy vzniklo označení myš, ale ujal se a zůstalo dodnes.

S vynálezem myši přišel vynález ukazatele myši. V té době měl tvar hůlky o velikosti jednoho znaku. Douglasův tým ho pojmenoval „bug“ v českém překladu brouk, nicméně tento název se do současnosti nezachoval. Douglasova demonstrace systému NLS ohromila mnoho lidí. To jim otevřelo oči a zamysleli se, co by bylo možné v budoucnosti, kde by pracovali lidé na celém světě na elektronických dokumentech na obrazovce svého počítače. Tato budoucnost ovšem nenasvědčuje nic dobrého pro firmu Xerox, která se živí prodejem kopírek. [8]

Firma Xerox se bála o svůj zánik v „bezpapírové budoucnosti“ a rozhodla se, že bude nejlepší ovládnout novou technologii. Proto vytvořili Palo Alto Research Center (PARC) v roce 1970. Jako první z věcí, které vynalezli, byla laserová tiskárna. Ale takováto tiskárna vyžadovala větší grafickou přípravu dokumentu k tisknutí. S faktem, že žádné takové počítače v té době nebyly, vynalezli svůj vlastní v roce 1973 a pojmenovali ho Xerox Alto. Nejnápadnějším rysem byl jeho displej, který byl orientován na výšku a měl stejné rozměry jako tištěné stránky. Displej představoval plnou rastrovou bázi o 606px na 808px. Každý pixel mohl být samostatně zapnut a vypnut na rozdíl od typických terminálových. Měl klávesnici a modernizovanou Engelbartovu myš. Kurzor myši se nyní

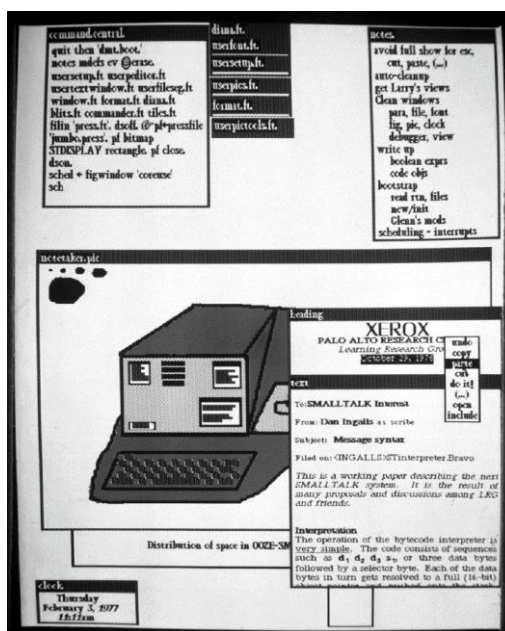
stal bitmapovým obrazem, který zobrazoval diagonální šipku, kterou známe dnes a stejně jako morphing do jiných tvarů v závislosti na vykonávané činnosti. [24]



Obr.č. 5 – GUI počítače Xerox Alto [20]

První software, který byl napsán pro Xerox Alto, byl hrubý a byl jen mírně grafický. Grafický textový procesor tzv. „Bravo“ byl vyvinut a mohl zobrazovat různá písma a velikosti textu na obrazovce současně, ale měl jiné uživatelské rozhraní, které bylo umístěno naspodu místo navrchu. Výzkumníci PARC si uvědomili, že je potřeba konzistentní uživatelské rozhraní pro nové aplikace a aby se tak stalo, muselo být vyvinuto úplně nové vizuální prostředí. A tak se zrodil SmallTalk první moderní GUI rozhraní. [19]

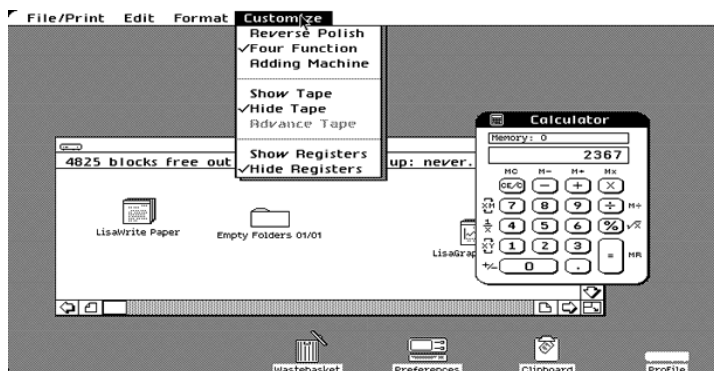
SmallTalk byl koncipován jako programovací jazyk a vývojové prostředí tak, aby byl snadno použitelný a v mnoha ohledech byl úspěšný. Vývojové prostředí SmallTalk obsahovalo také grafické uživatelské rozhraní, na němž SmallTalk běžel a v mnoha ohledech zavedl moderní GUI pojetí. Jednotlivé okna ve SmallTalku obsahovala ohraničení a šedý vzor pro pozadí. Každé okno mělo záhlaví na horním řádku, které bylo použito pro identifikaci a přesouvání po obrazovce. Okna mohla překrývat jiná okna a vybrané okno se přesouvalo do popředí obrazovky. Ve SmallTalku byl také poprvé uveden pojem „ikona“ – malé ikonické zobrazení programů a dokumentů. [19]



Obr.č. 6 – GUI SmallTalku [9]

V době SmallTalku se poprvé také objevují popup menu, kdy po kliku myši na tlačítko se rozevřelo hierarchické menu na poslední pozici kurzoru. Dále se prvně objevily posuvníky, přepínače a dialogová okna. [20]

Nejdůležitější z průkopníků a vývojářů GUI byla malá firma založená v garáži nazvaná Apple Computer. Byla založena v roce 1976 Stevem Jobsem a Stevem Wozniakem. Apple postavil svůj majetek na Apple II, který byl velmi populární a zobrazoval text i grafiku, ale obsahoval také příkazový řádek. Mnoho bývalých inženýrů z Xerox PARC chtěli pracovat pro Apple. V Applu začala práce na příštím počítači nazvaném Lisa, který zprvu začal na textové bázi, ale po příchodu pracovníku z PARC byl jejich vlivem změněn na grafický. Steve Jobs se stal GUI vizionářem. Lisa se stal grafickým počítačem, avšak grafické rozhraní ještě nebylo zcela hotové. [19]

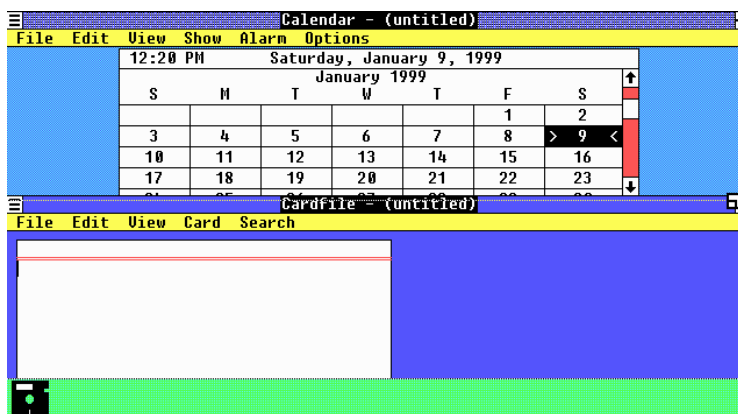


Obr.č. 7 – GUI počítače Apple Lisa [20]

Tým, který pracoval na počítači Lisa, uznal jako nejlepší řešení ikonový systém, kde každá ikona zobrazovala dokument nebo aplikaci a vyvinul první vysouvací menu, které obsahovalo všechny nabídky a bylo umístěno na horním řádku displeje. Tým Lisy změnil tehdejší třílačítkový design myši na dvoutlačítkový a dále definovali aktivování dvojklikem. Poklepání se později stalo plně standartizovaným úkonem pro všechny GUI pro spuštění programu a to i pro myši s více tlačítky. Uživatelské rozhraní Lisy definovalo GUI pojmy, které používáme dodnes. Přišli s myšlenkou, že ikony mohou zastupovat všechny soubory v souborovém systému, které by pak uživatel mohl procházet pomocí hierarchického adresáře. Dále myšlenky „drag and drop“, to znamená vybrání více souborů pomocí myši a přesunutí do jiné složky za pomoci tahu. Poprvé se také objevila myšlenka tříd, která poskytovala, že každý soubor by mohl být spuštěn v přiřazené aplikaci. [9]

U nového GUI počítače Macintosh měla operační paměť velikost pouze 128KB, nebyl přítomen žádný multitasking ani přepínání mezi programy. Uživatelské prostředí Macintoshe je postaveno na GUI z Lisy, ale operační systém byl přepsán od nuly, aby splnil nároky na operační paměť. [23]

V dalších letech se mnoho lidí snažilo vymyslet nový GUI operační systém. Mezi tyto GUI systémy se řadí VisiOn, Tandy DeskMate, GEM, Amiga workbench, které ovšem díru do světa neudělaly. Avšak nejznámějším byl Windows v té době verze 1.0. Obsahoval veškeré GUI prvky, jako jsou posuvníky, menu, dialogová okna a widgety. Lišil se oproti Lise nebo Macintoshi v tom, že každá aplikace měla svoji vlastní ovládací lištu a menu k ní připojené. Po velké oblibě uživatelů vyšla v roce 1987 aktualizovaná verze Windows 2.0. Opouštěl kachličkový systém a přesedlal na překrývající se model. Tato verze spolu s vylepšeným softwarem byla prodávána firmou HP s názvem „NewWave“. [23]



Obr.č. 8 – GUI operačního systému Windows 1.0 [9]

Dalším milníkem ve vývoji GUI bylo uvedení nového operačního systému Next v roce 1988 vyvinutého firmou Steva Jobse. Next představil ostrý 3D design GUI prostředí, poprvé zavedl symbol X pro zavírání aplikace nebo widgetu. Také přišel s myšlenkou visulé nabídky s pruhem v levém horním rohu, aby uživatel mohl opustit nabídky v libovolném bodě na obrazovce. Next obsahoval také panel, který byl pevně umístěn na pravé straně, který v dnešní době používal například systém Windows Vista. [24]



Obr.č. 9 – GUI systému Next [23]

Na přelomu roku 1990 ostatním počítačovým systémům klesala oblíbenost, zůstali na poli GUI pouze Microsoft a Apple. Digital Research zastavil prodej systému GEM v roce 1989. Atari přestala prodávat ST v roce 1993. V roce 1993 se firma Next rozhodla přestat prodávat hardware a vrhla se výhradně na operační systém. [24]

Firma Microsoft dosáhla nebývalé popularity současně s vydáním verze Windows 3.0 v roce 1990 a 3.1 v roce 1992. Zatímco co Macintoshi chybí ještě hodně funkcí, přišel na trh Windows 95 vydaný v roce 1995 a ten stmelil Microsoftu náskok v oblasti prodeje GUI a stal se jedním z neprodávanějších programů. GUI prostředí se dále vyvíjelo a vyvíjí se stále. Po roku 2000 následovaly aktualizované verze systémů Mac OS a Windows, jak je známe dnes. Objevilo se také nové GUI prostředí pro operační systém Linux. I toto prostředí se aktualizovalo a vznikaly nové verze. Nelze jednoznačně říci, kdo vynalezl GUI. Každý kdo se podílel na vývoji, má zásluhu. [20]

## 4 DŮVODY ZKOUMÁNÍ GUI GENERÁTORŮ

Téma diplomové práce mi vrtalo už delší dobu hlavou. Poté se naskytla možnost vytvořit GUI mockup ovladače pro firmu Santech. Zákazník požadoval prezentaci logiky GUI aplikace s její interakcí. Cílem bylo nalezení optimálního ovládání a vyřešení přístupu a nastavitelnosti jednotlivých funkcí. Pro tvorbu prezentace GUI aplikace byl dodán grafický návrh s náhledy jednotlivých obrazovek. Po několika schůzkách, kde se prezentovala hotová aplikace, se vždy logika ovládání měnila pro nalezení snadnějšího a intuitivnějšího přístupu k funkcím GUI aplikace. Tyto časté úpravy mě vedly k myšlence, jestli neexistuje nějaký software, pomocí kterého bych mohl jednoduše měnit požadované vlastnosti, prvky a chování GUI aplikace do budoucna. Hlavní požadavek byl uspořít čas, který výrazně narůstal s každou změnou.

Bylo tedy zapotřebí najít hotové softwary, které umožňují vytvářet tyto GUI mockupy bez nutnosti něco programovat nebo hlubšího nastudování jejich fungování. Ze začátku se jednalo pouze o prezentaci GUI aplikace. Téma však bylo rozšířeno o oblast embedded a mobilních systémů, které jsou v dnešní době velice populární a výdělečné. Hledal jsem tedy i softwary, které umožňují z vytvořené GUI aplikace vygenerovat kód pro cílové zařízení nebo podpůrné vývojové desky, kde by byla GUI aplikace prezentována.

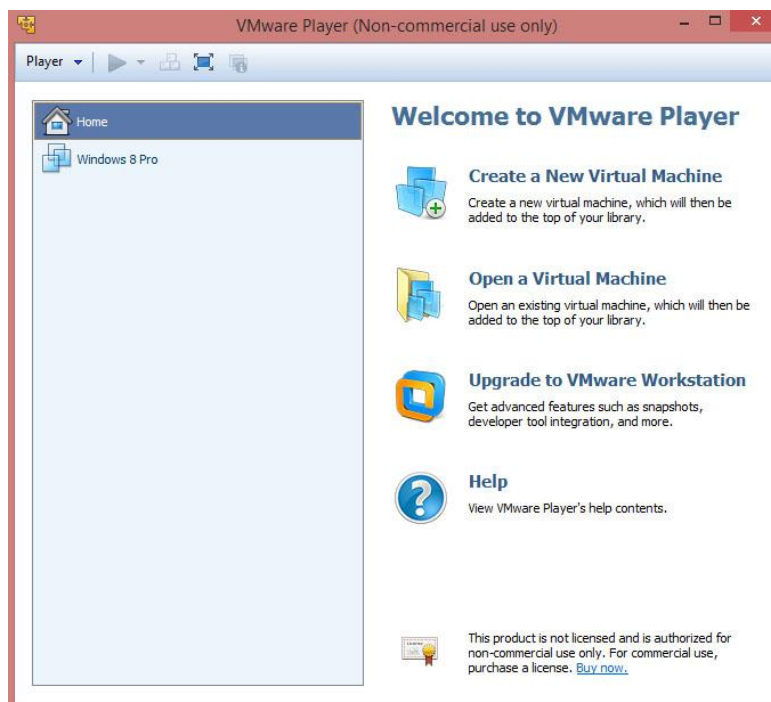
## **II. PRAKTICKÁ ČÁST**

## 5 POUŽITÝ SOFTWARE

Zde se nachází přehled software, který jsem použil pro vytvoření individuální návrhu GUI a k prozkoumání možností nalezených GUI designerů a generátorů.

### 5.1 Vmware Player

Vmware Player je virtualizační nástroj pro testování software bez rizika poškození operačního systému a je opensource pro nekomerční účely. Umožňuje vytvořit libovolný virtuální stroj s požadovaným systémem (Windows, Linux atd) a nastavit mu jednotlivé parametry. Software je dostupný ke stažení z domovských stránek výrobce v sekci Downloads/Vmware Player. [34]



Obr.č. 10 – Vmware Player

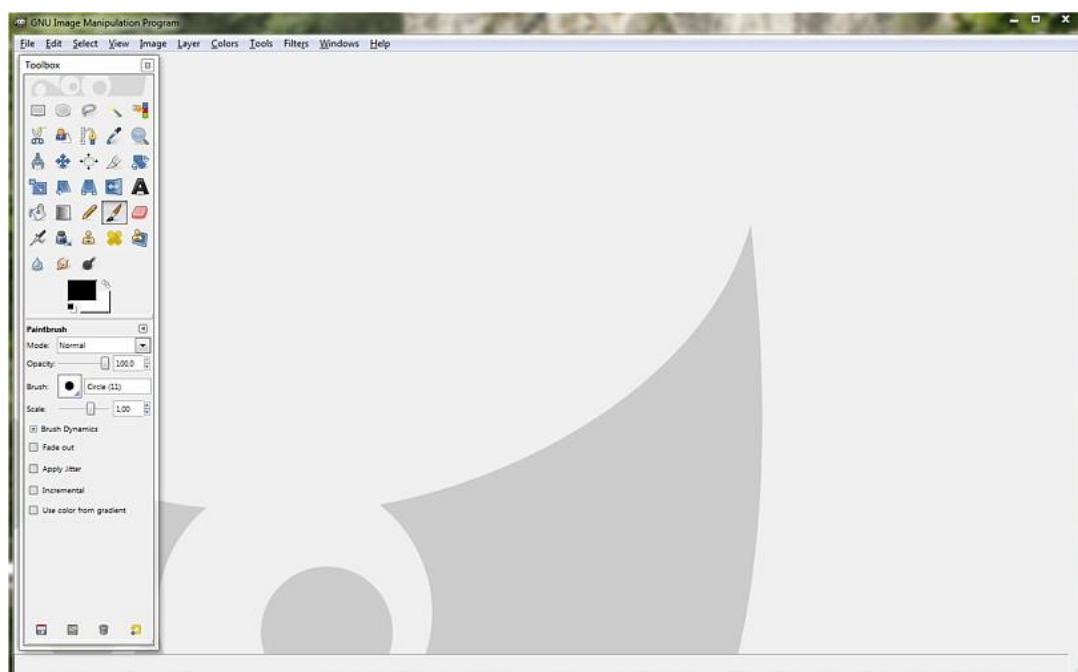
Program Vmware Player jsem použil z důvodu zkoumání jednotlivých GUI softwarů. Řešil jsem to tímto způsobem, protože prozkoumání obsahovalo nainstalování softwaru, testování funkčnosti a v některých případech i odinstalování z důvodu nesplnění požadavků. Hlavními důvody bylo zachování bezpečnosti a integrity vlastního systému a také aby funkčnost jednotlivých GUI softwarů nebyla ovlivněna programy, které jsou již nainstalované na mém systému.

Jako virtuální stroj jsem zvolil jednotku s operačním systémem Windows 8. Instalace probíhala úplně stejně jako u klasického systému s jediným rozdílem, že jsem systém instaloval z obrazu disku. Při instalaci jsem změnil parametry z důvodu rychlosti práce ve virtuálním stroji. Vytvořené a vygenerované výstupy jednotlivých softwarů jsem jednoduše přenesl přes USB jednotku, kterou jsem připojil k virtuálnímu stroji a překopíroval požadované soubory.

## 5.2 Gimp verze 2.8

Jedná se o open source bitmapový editor. Název je zkratkou „GNU Image Manipulation Program“, autoři programu jsou Peter Mattis a Spencer Kimball, kteří studovali univerzitu v Berkeley. V současné době jsou aktualizace a vývoj programu na bedrech dobrovolníků. Program je lokalizován do češtiny, zvolil jsem verzi normální, protože není nutná práce na více počítačích. Ještě je k dispozici verze „Portable“, která jde spustit kdekoli (nevyžaduje instalaci) a může se přenášet na přenosných discích. GIMP má podporu veškerých klasických bitmapových formátů (JPG,BMP,PNG,TIF,GIF a další). [5]

Software je dostupný ke stažení z domovských stránek výrobce v sekci downloads. [35]



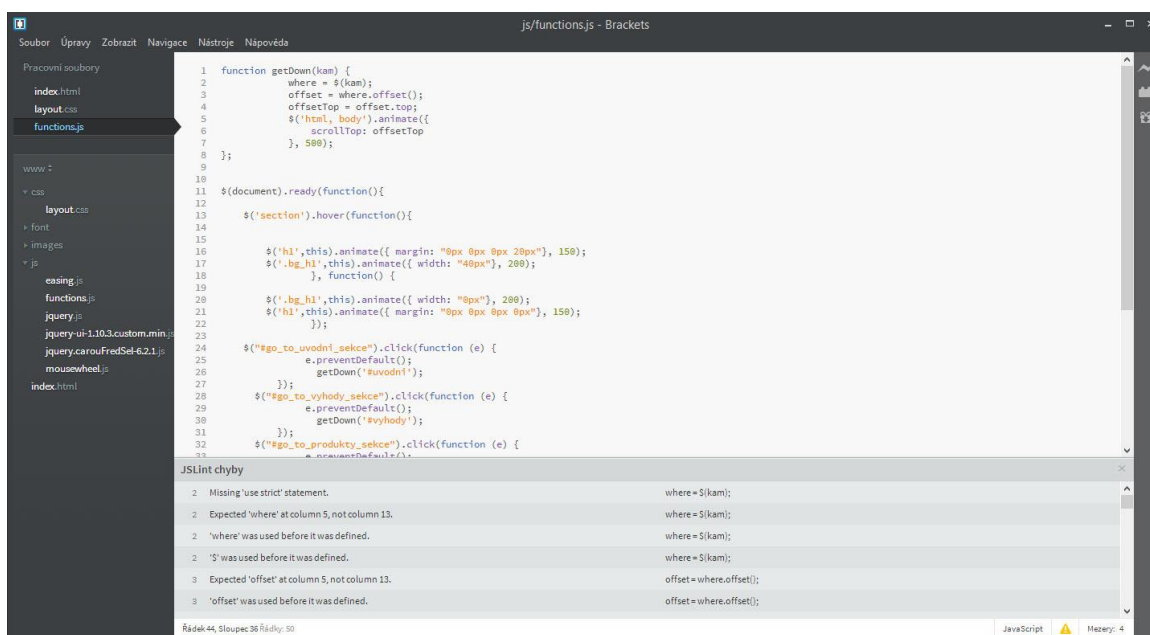
Obr.č. 11 – Uživatelské rozhraní programu GIMP

Program GIMP byl použit pro úpravy obrázků pro diplomovou práci a také výhradně pro rozřezání dodané grafiky ve formě jednotlivých obrazovek. Jednotlivé elementy obrazovek jsem ukládal ve formě PNG obrázkových souborů z důvodu úspory velikosti.

### 5.3 Brackets Sprint

Brackets je open source program pro úpravu CSS, HTML a JavaScript kódu. Brackets představuje sbírku vývojářských nástrojů, které vám umožní upravit jednoduše pravidla kódu či různé skripty a potom analyzovat výkon webové stránky. Aplikace nabízí uživatelsky příjemné prostředí, ve kterém se dobře pracuje a navíc obsahuje podporu češtiny. Brackets obsahuje zvýraznění syntaxe textu pro snadnější úpravu kódu. Je multiplatformní a běží na systémech Windows XP, 7, 8 a Mac OSX.

Software je dostupný ke stažení z domovských stránek výrobce. [36]



Obr.č. 12 – Uživatelské prostředí Brackets Sprint

Program Brackets Sprint jsem využil zejména pro druhý bod diplomové práce, kdy jsem tvořil individuální návrh GUI aplikace pomocí technologie DHTML a upravoval jsem v něm HTML, CSS, JS a Jquery kódy.

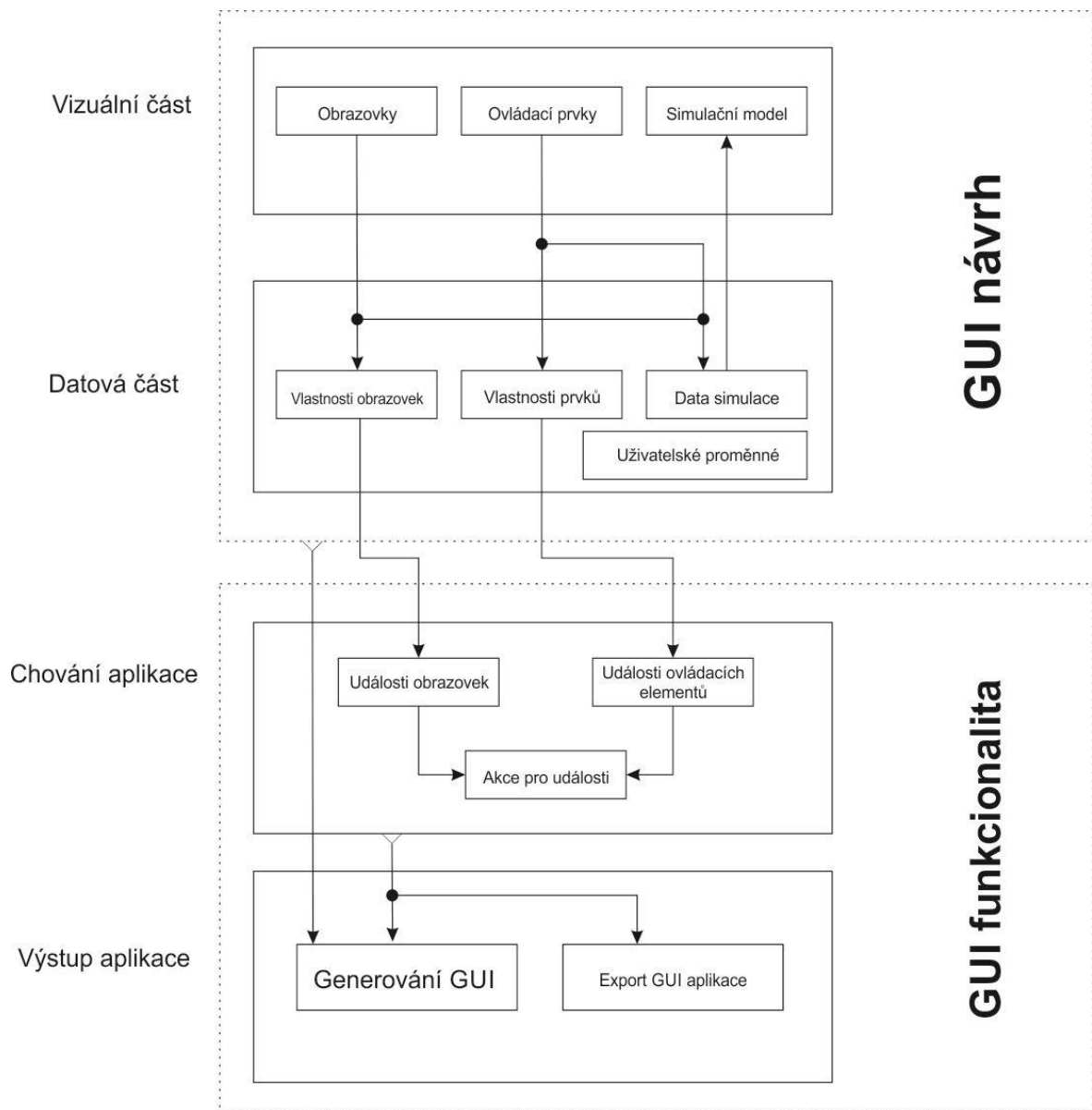
## 6 ANALÝZA FUNKČNÍCH A NEFUNKČNÍCH POŽADAVKŮ

Prvním úkolem mé diplomové práce je analyzovat a zpracovat požadavky na generátory GUI pro mobilní a embedded systémy. Abych tyto požadavky mohl lépe formulovat, měl jsem vytvořit demo GUI aplikace, ze kterého jsem měl vysledovat jednotlivé fáze a nezbytné prvky pro tvorbu. Zároveň jsem měl definovat požadavky na generaci kódu a exportu aplikace.

Po vytvoření dema GUI jsem analyzoval funkční požadavky, tzn. ty požadavky, co se musí udělat (aktivity, vizuální tvorba, akce). Jako nefunkční požadavky jsou interpretovány různá omezení softwaru, co se týče výkonnosti, spolehlivosti, oblast generování jazyka a pro jaké cílové zařízení je aplikace vhodná. Pomocí analýzy jsem rozdělil tvorbu GUI do 4 základních oblastí.

- Vizuální sekce
- Datová sekce
- Sekce chování aplikace
- Sekce generování / exportu

Tyto sekce představují oddělené aktivity při tvorbě, avšak pro GUI software jsou naprosto nezbytné. Níže na obrázku je vyobrazeno schéma požadavků na GUI software i s jeho vazbami na dané sekce a jak jsou propojeny.



Obr.č. 13 – Schéma požadavků na GUI software

## 6.1 GUI návrh

Základní požadavky na software jsou, aby obsahoval WYSIWYG editor a měl logicky uspořádané ovládací panely bez nutnosti přepínání.

Sekce GUI návrh se dělí na 2 základní oblasti – vizuální a datová oblast. V těchto sekcích se tvoří GUI aplikace z pohledu testovacího a logického členění s jednotlivými parametry. Tvoří se zde posloupnosti obrazovek. Chování a výstup aplikace se řadí do sekce GUI funkcionality, kde definujeme, co má daná aplikace dělat a pro jaký výstup je preferována.

Dalším důležitým požadavkem je možnost importovat zdrojové soubory pro tvorbu GUI aplikace. Software tak musí umožňovat nahrát např. obrázky do vytvořeného projektu a dále s nimi pracovat.

Vizuální část návrhu je základní kámen pro sestavení GUI. Software musí obsahovat veškeré standartní ovládací prvky, s kterými se můžeme při tvorbě GUI setkat:

- Tlačítko
- Obrázkové tlačítko
- Textarea
- Input
- Text
- Výplň

Tyto prvky musí být editovatelné. Výhodou softwaru by bylo, kdyby si uživatel mohl nadefinovat vlastní ovládací prvek a vytvořit jej jako šablonu nebo předlohu pro další použití, aniž by musel znova definovat parametry.

U vizuální stránky GUI softwaru musí být přítomny nástroje, které obsahují grafické programy jako všechny typy zarovnání a editace objektů do popředí nebo pozadí. Protože každý návrh GUI aplikace je založen na posloupnosti obrazovek, musí GUI software obsahovat panel nebo jakýkoli nástroj s možností správy jednotlivých obrazovek GUI aplikace. Tento nástroj musí umožnit vytvářet libovolný počet obrazovek a mazat obrazovky. Výhodou softwaru bude přítomnost funkce duplikování nebo kopírování obrazovek, protože mnoho návrhu GUI vychází ze stejného základu obrazovek. Vytvořené obrazovky musí mít možnost úpravy jednotlivých údajů a jejich velikostí. Musí se dát tvořit posloupnosti obrazovek a musí obsahovat příznak, kterým uživatel zajistí, že daná obrazovka bude nastavena jako výchozí bod GUI aplikace a od ní se bude návrh dále odvíjet. Pokud nebude obsahovat příznak startovní obrazovky, musí být GUI software založen na vrstveném modelu, kde první obrazovka bude startovní.

Software by měl také obsahovat simulační běh GUI aplikace, který spadá také do vizuální části návrhu. Hlavní požadavek na simulátor je věrné zobrazení a funkčnost GUI aplikace i v případě, že nebude embedded systém připojený. Toto by byl ideální stav, který ovšem ne

vždy bude reálný. Simulátor musí mít přístup k datům a funkcím GUI aplikace kdykoli, aby uživatel mohl testovat funkci během tvorby.

Datová část GUI návrhu musí obsahovat veškeré informace a parametry o vložených obrazovkách a ovládacích prvcích. Tyto informace mají hodně společného. Základní parametry pro všechny ovládací prvky musí obsahovat:

- Pozice x, y na obrazovce
- Výška a šířka ovládacího prvku
- Identifikátor (název nebo ID)
- Pozice z-index (popředí, pozadí)

Parametry u obrazovek musí splňovat základní požadavky:

- Velikost obrazovky (šířka, výška)
- Id nebo název obrazovky
- Příznak startovní obrazovky

U vyjmenovaných ovládacích prvků se mohou vyskytovat také jedinečné parametry v závislosti na tom, o jaký ovládací prvek se jedná. Jako příklad uvedu Button a Image button. U klasického tlačítka jsou hlavní rozměry jako šířka, výška, pozice x,y, tyto parametry samozřejmě obsahuje i obrázkové tlačítko, avšak má svůj jedinečný parametr a to odkaz na obrázek, který by měl být ve formě adresy na zdrojový soubor ve složce, kterou zvolí uživatel, nebo přímo možnost vybrat obrázek vložený do projektu. Obrázkové tlačítko by mělo podporovat standartní formáty bitmapových souborů, jako jsou BMP, JPEG, PNG. U textových ovládacích prvků musí být možné měnit text, stylovat font co se týče velikost a barvy textu. Nadstandartní funkcí by bylo měnit také samotný typ fontu.

Datová část GUI návrhu musí obsahovat také nástroje pro tvorbu uživatelských proměnných. Tyto proměnné mohou sloužit následně jako větvení chování GUI aplikace. Uživatelské proměnné by měly obsahovat klasické datové typy, které jsou známé z různých programovacích jazyků:

- Int
- Float
- String

- Boolean (true, false)

Počet uživatelských proměnných by neměl být omezen a přístup k těmto proměnným by měl být povolen. To znamená, že uživateli bude umožněno měnit hodnoty těchto proměnných pomocí jednotlivých akcí a nastavovat tak dynamicky jejich hodnotu.

Součástí datové části je příprava dat pro simulaci GUI aplikace. Tedy data potřebné k simulaci musí být přístupny bez nutnosti generovat výsledný kód pro cílovou platformu nebo exportovat aplikaci.

## 6.2 GUI funkcionalita

Samostatná kapitola požadavků je vztažena na definování funkčnosti a výstupu GUI aplikace. Software musí obsahovat nástroje nebo funkce pro správu chování GUI aplikace z pohledu ovládacích prvků, které se budou na obrazovce vyskytovat tak i na správu chování celé obrazovky a jejich událostí. V jednotlivých událostech musí být možnost vytvářet konkrétní akce, které budou vykonávány, pokud nastane zvolená událost.

Tyto nástroje pro správu chování musí umožňovat vytvářet základní události, které budou vyvolány uživatelem aplikace po interakci na ovládací prvek. Jelikož se v praxi GUI návrhy dělají přímo pro embedded zařízení, bude zařízení testováno nebo vyvíjeno na nějakém dotykovém displeji nebo displeji spojeném s externími klávesami, proto jediným požadavkem je, aby software umožňoval zvolit tyto události:

- Událost po kliku myší na element
- Událost po dotyku na element

Dále by měl obsahovat automatické události obrazovek:

- Událost po načtení obrazovky
- Událost po změně obrazovky

Výhodou by byla možnost zavedení zpoždění mezi jednotlivými událostmi a definování délky trvání. Všechny události pouze zachycují to, co se děje s GUI aplikací, ale software musí obsahovat také jednotlivé akce, které se budou vykonávat, pokud se událost spustí. Tyto akce musí zahrnovat alespoň některé operace s elementy na obrazovce.

- Skrytí a zobrazení elementu

- Změna parametru elementu (např. zdroj obrázku)
- Změna datových proměnných
- Změna pozice elementu (x,y)
- Změna rozměrů elementů (šířka, výška)

Pro účely prezentování výstupu musí GUI software obsahovat nástroje pro export nebo generování aplikace. Pokud se jedná o embedded systém musí obsahovat kompilátor a generování kódu pro cílovou platformu. Výhodou by byla možnost vygenerovat výstup pro mobilní systémy ve formě standardizovaných instalačních balíčků. Pro designery je nutné, aby obsahovaly export do HTML, včetně všech potřebných skriptů a dalších souborů.

### **6.3 Shrnutí funkčních a nefunkčních požadavků**

#### **6.3.1 Funkční požadavky**

- Software bude umožňovat vkládat všechny standartní ovládací prvky
- Ovládací prvky budou obsahovat název nebo ID, pozice x,y a rozměry prvku
- Software bude umožňovat tvořit uživatelské proměnné (int, float, string, boolean)
- Software musí obsahovat správce obrazovek a vestavěný simulátor běhu
- U obrazovek musí být umožněno měnit rozměry, název
- Správce obrazovek musí mít vrstvený model nebo příznak startovní obrazovky
- Obrázkové ovládací prvky musí podporovat alespoň jeden z těchto bitmapových formátů (JPG, PNG, BMP, GIF)
- Software musí umožňovat nahrát zdrojové soubory (obrázky)

#### **6.3.2 Nefunkční požadavky**

- Software musí umožnit generovat výstupní kód pro embedded systém nebo exportovat GUI aplikaci do HTML
- Software musí být vizuální WYSIWYG editor
- Omezení vývojové platformy není kladeno (C,C++,Java)

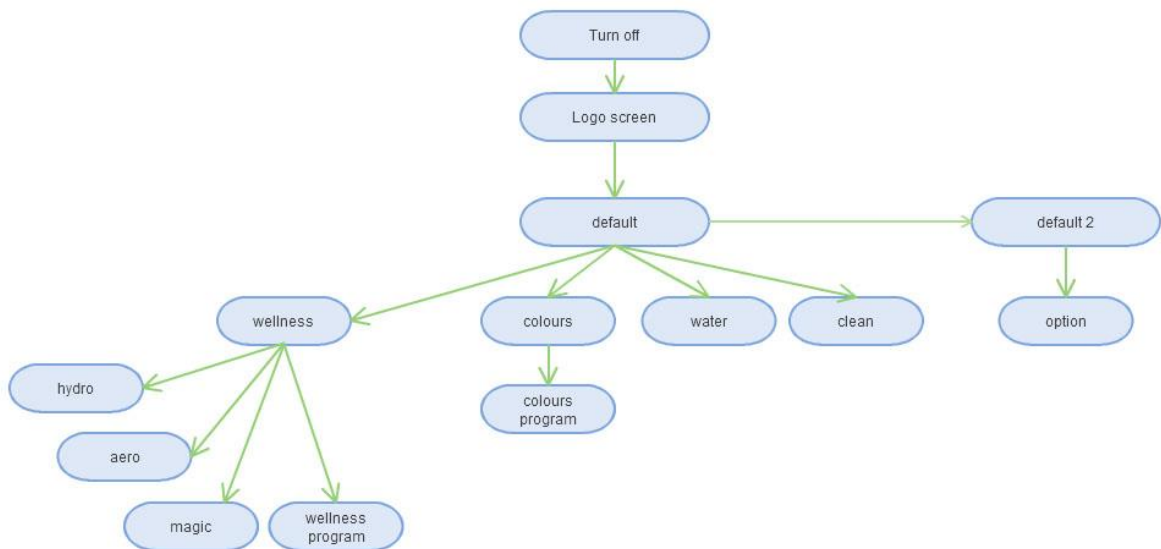
## 7 INDIVIDUÁLNÍ ŘEŠENÍ POMOCÍ DHTML

Jako vzorovou GUI aplikaci jsem vytvořil návrh logického ovládání GUI ovladače van pro firmu Santech. Návrh jsem implementoval pomocí technologie DHTML. DHTML je soubor několika technologií pomoci, kterých může stránka měnit obsah i po jejím načtení. První krok bylo rozřezání dodané grafiky na elementy, které budou následně použity pro vyvolání interaktivnosti. Rozřezání elementů jsem prováděl v bitmapovém editoru GIMP. Obrázky jsem ukládal do formátu PNG jednak z důvodu velikosti a zadruhé z důvodu podpory jednotlivých GUI softwaru. Rozřezané elementy byly uloženy hierarchicky do složek odpovídající jednotlivým obrazovkám, aby se zvýšila přehlednost.

Jako editor HTML, CSS a Javascript kódu jsem použil program Brackets Sprint. V programu Brackets jsem si vytvořil prázdný HTML dokument. Do něj jsem vkládal div tagy, kterým jsem přiřazoval jednotlivé třídy a ID elementů. Třídy jsem přidával kvůli stylování elementu a jednotlivé ID bylo přidáno kvůli vyvolání událostí s reakcí na tyto prvky. Celá aplikace ovladače se skládá z jednoho HTML souboru „index.html“ a několika php souborů odpovídající každé obrazovce. Propojení jednotlivých skriptů a CSS souborů bylo provedeno formou odkazů v záhlaví dokumentu. Individuální návrh je dostupný online na adrese <http://santech-comfort.solansky.cz>.

### 7.1 Obrazovky

Individuální návrh obsahuje celkem 14 obrazovek. Funkčních obrazovek je pouze 12. První obrazovka „turnoff“ obsahuje pouze ovládací prvek zapnutí, který načte další obrazovku „logo screen“ s logem výrobce. Po uplynutí vteřiny zpoždění se GUI aplikace dostává do základního menu, které obsahuje 2 obrazovky, mezi kterými jde rolovat pomocí ovládacího prvku. Na Obr.č.14 je naznačena hierarchická posloupnost obrazovek.



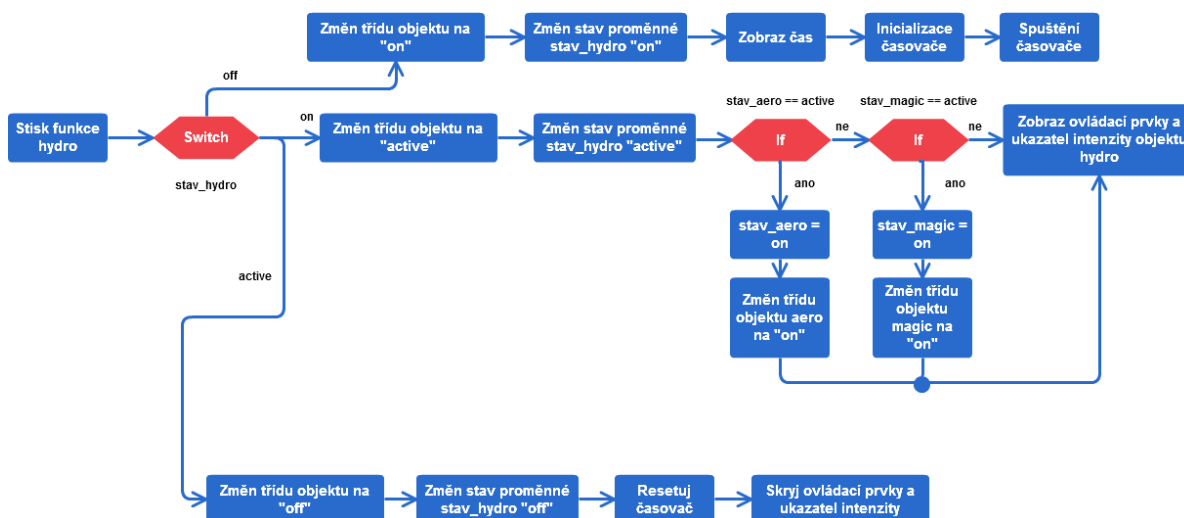
Obr.č. 14 – Hierarchické schéma obrazovek GUI návrhu

Ze schématu obrazovek je patrné, že GUI návrh obsahuje 3 úrovně obrazovek. První úroveň je základní menu GUI aplikace, které obsahuje obrazovky default a default2. Druhá úroveň jsou funkční menu, které obsahuje obrazovky wellness, colours, water, clean a option. Tyto prvky slouží jako rozcestník pro další obrazovky GUI návrhu. V menu wellness se nastavují jednotlivé programy a jejich intenzita. Menu wellness dále obsahuje linky na další obrazovky hydro, aero, magic a program. V menu colours se nastavují a zapínají barevné tóny. Obrazovka water slouží pro nastavení teploty vody. Menu clean slouží pro nastavení metody hygieny. Menu option nabízí nastavení délky trvání jednotlivých programů a délku trvání masáže. Třetí úroveň obrazovek jsou dodatečné nastavení programů v menu wellness a spouštění přednastavených programů.

## 7.2 Diagramy aktivit

Vytvořil jsem pár diagramů aktivit zachycující akce, které se provádějí, když je spuštěna příslušná událost. Vybral jsem funkci hydro na obrazovce „wellness“, ovládání teploty na obrazovce „water“ a proces aktivace výběru barvu v menu „colours“.

### 7.2.1 Funkce hydro

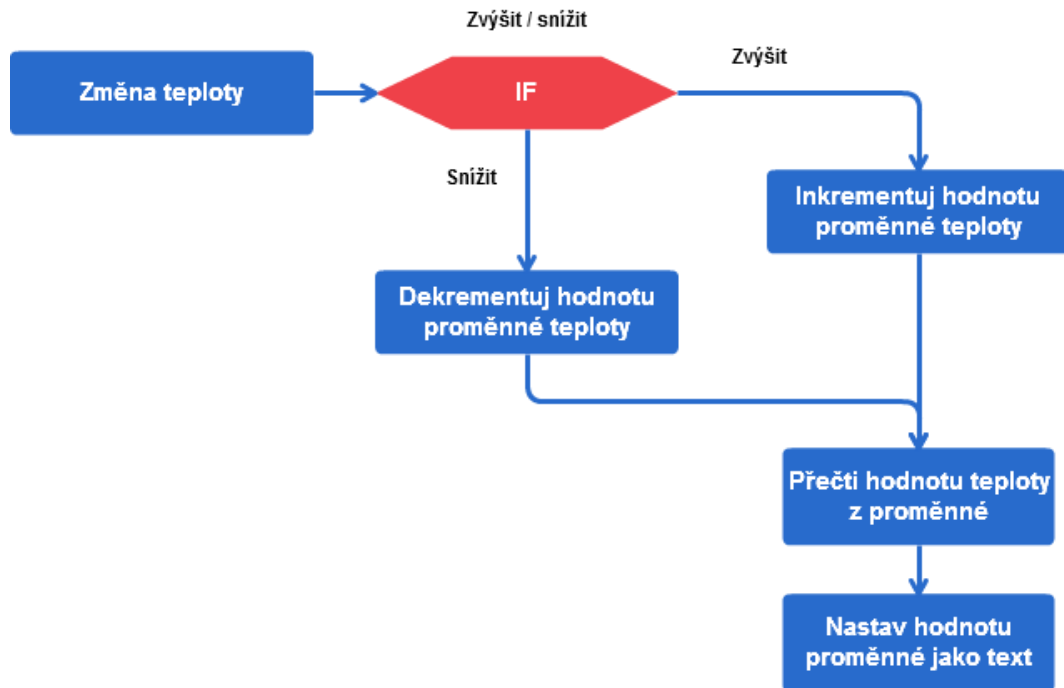


Obr.č. 15 – Diagram aktivity funkce hydro

Při stisku funkce hydro jsem na začátku funkce větvil tok pomocí přepínače. Jako podmínku jsem dal vytvořenou proměnnou „stav\_hydro“, která uchovává stav funkce hydro. Pokud je podmínka splněna provede se příslušná větev uživatelského kódu. Ve větvi pro stav „off“ změním hodnotu stavové proměnné a pomocí jquery funkce .addClass() přidám objektu hydroButton třídu „on“. Zároveň zobrazím odpočítávání času pro program hydro. Zobrazení času je provedeno jquery funkcí show() a délka odpočítávání je inicializována v časovači, kde tato hodnota se bere z globální proměnné „cas“. Tato proměnná je dále nastavitelná v obrazovce „option“. Pro větev stavu „on“ je to podobné pouze zde přibývají 2 podmínky, kdy testuji zda-li není aktivní jiný program. Pokud ano tak ho nastavím pouze do stavu „on“ a změním třídu daného objektu programu na „on“. Dále nahraji ukazatel intenzity do informační lišty pro daný aktivní program. Jako poslední krok je zobrazení aktivního ovládání funkce. Ovládací prvky budou označeny červeně oproti stavu „off“, kde jsou tmavě šedé. Poslední větev pro stav „active“ je deaktivace funkce. Změním třídu objektu na „off“ a také stavovou proměnnou na „off“ a resetuji časovač. Součástí je také skrytí ovládacích prvků a ukazatele intenzity.

Pro programy aero a magic byly funkce při stisku ovládacího prvku realizovány stejně jako u funkce hydro, pouze s rozdílem větve, kdy je stav „on“. Zde se testují vždy zbylé dva programy, pro zajištění ovládání jednoho programu ovládacími tlačítky + a -.

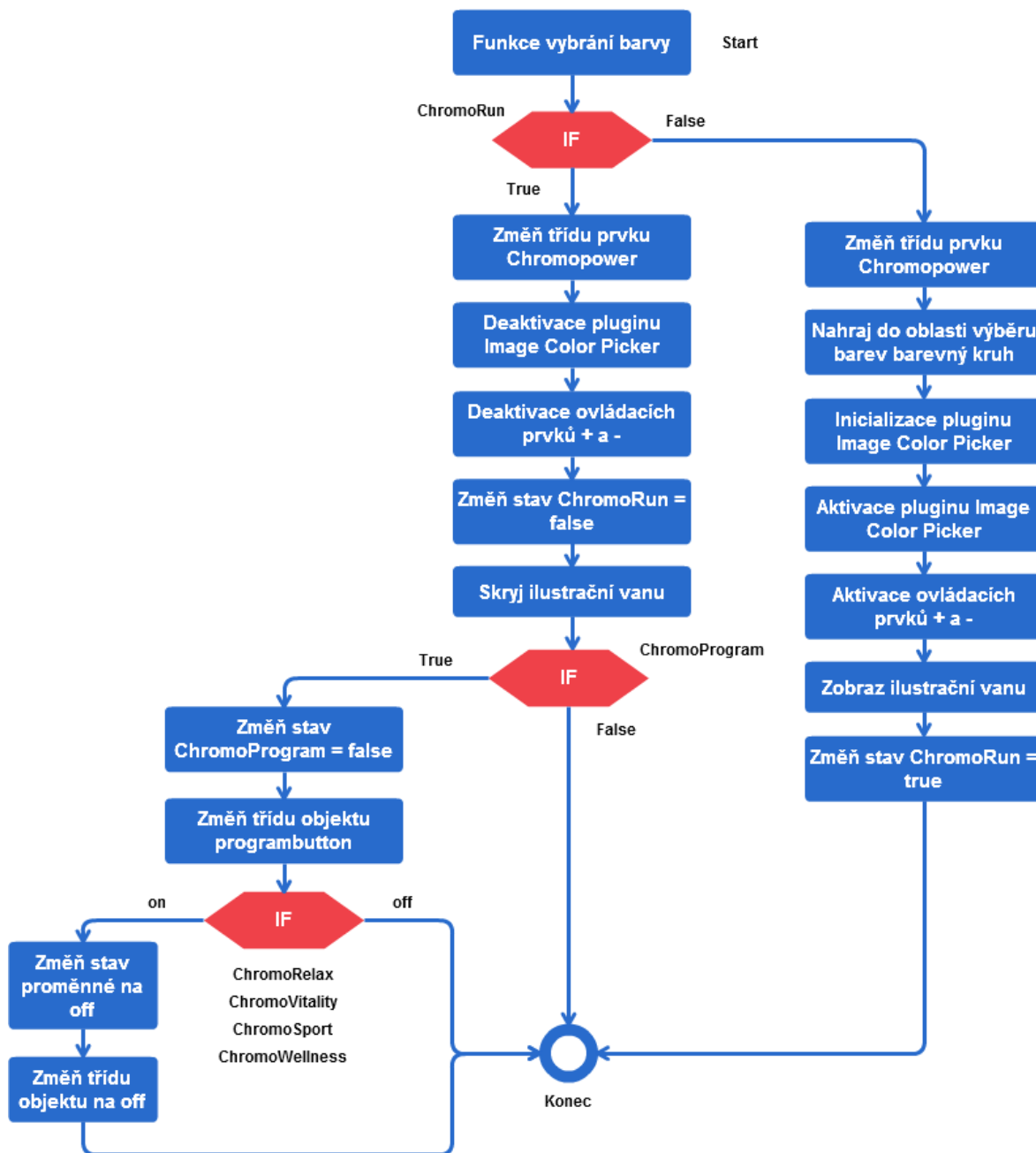
### 7.2.2 Funkce ovládání teploty



Obr.č. 16 – Diagram aktivit změny teploty

Změna teploty byla realizována velice jednoduše pomocí jedné funkce, která byla použita pro stisknutí ovládacích prvků + a -. Jako parametr funkce jsem vkládal znaménko, které signalizovalo zvýšení nebo snížení teploty. Poté jsem hodnotu proměnné teploty inkrementoval nebo dekrementoval podle zachycené události. Na konci funkce jsem přečetl aktuální hodnotu z proměnné teploty a nastavil ji jako text objektu, který zobrazuje teplotu na obrazovce „water“, pomocí JQuery funkce `.text(,hodnota‘)` objektu podle ID.

## 7.2.3 Funkce vybrání barvy



Obr.č. 17 – Diagram aktivit funkce výběru barvy

Funkce vybrání barvy je trochu složitější, ale opět byla provedena pomocí jedné javascript funkce. Tato funkce zajišťuje zapnutí a vypnutí, vybrání barvy a inicializaci barevného pluginu.

Na začátku funkce testuji proměnnou ChromoRun. Na základě testu se další akce větví do dvou sekcí. ChromoRun signalizuje stav, kdy je vybrání barvy neaktivní nebo aktivní hodnotami false nebo true. Pokud je neaktivní, podle ID objektu změním třídu elementu

zapínání na aktivní. Neaktivní zašedlá kružnice barev je nahrazena za jiný zdroj obrázku a při této operaci je provedena inicializace a spuštění pluginu Image Color Picker, který je zaměřen přímo na nově nahraný zdroj. Zároveň aktivuji ovládací prvky pro regulaci intenzity barvy. Poslední krok je změna stavové proměnné ChromoRun na hodnotu true. Pokud je proměnná ChromoRun rovná hodnotě false, znamená to, že výběr je aktivní a uživatel chce funkci vypnout. V tomto případě je aplikován opačný postup pouze s jedním rozdílem. Na konci větve se testuje, zda-li jsou vypnuté i přednastavené programy. Pokud ne, tak se nic neděje, ale pokud jsou programy zapnuté, jsou postupně vypnuty a jejich stavové proměnné změněny na hodnotu off.

### 7.3 Změna obrazovek

Změna obrazovek byla provedena pomocí technologií Javascript a JQuery. GUI návrh ovladače byl rozdělen do několika logických oblastí. První je informační lišta nahoře, část nabídek a menu uprostřed a dolní oblast je ovládací sekce. Informační lišta zůstává fixní až na výjimky (nastavení intenzity programů). Oblast menu se mění v závislosti na vyvolaném požadavku. V projektu jsem vytvořil složku menu a do ní jsem vložil php soubory s daným názvem obrazovky. Při vyvolání akce kliknutí na ikonu jsem pomocí technologie JQuery konkrétně funkce .load() nahrál do oblasti menu nový php soubor s příslušným obsahem. Přechody obrazovek jsou převážně řízeny 2 tlačítky – tlačítko zpět a tlačítko domů. Tlačítko domů vrací z kterékoli obrazovky do hlavního menu (obrazovka default). Tlačítko zpět vrací pouze o jeden krok. V aplikaci je to vyřešeno stavovou proměnnou, kdy vždy po změně obrazovky se uloží název předešlé obrazovky do proměnné. Vracení o krok zpět je zachyceno funkcí a dynamickým nahráním menu podle hodnoty proměnné.



Obr.č. 18 – Rozdělení logiky GUI ovladače na 3 zóny

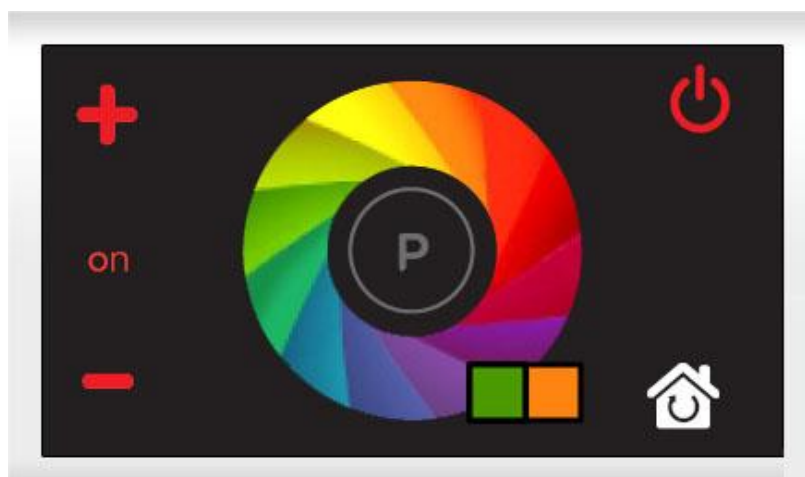
## 7.4 Definice chování aplikace

Chování GUI aplikace je řízeno pomocí technologie Javascript. Jsou přítomny 2 soubory (init.js a functions.js). V init.js se nachází definice chování při prvotním spuštění aplikace tzv. inicializace. V druhé js souboru jsou definovány veškeré funkce a události, které mohou nastat. Aplikace byla dělána jako jeden dynamicky měnitelný html soubor. Má to však své nevýhody. Po načtení nového menu již nelze přistupovat k chování elementů přes identifikátor, a proto jsem využil u nově načtených elementů funkce onclick. Při interakci elementu se spustí daná funkce vyvolaná onclick událostí, kde jsem definoval název a parametry funkce, která má být provedena. Při psaní funkcí jsem se snažil o univerzálnost, avšak ne vždy toho šlo docílit. Pro potřeby definice chování aplikace jsem vytvořil mnoho stavových proměnných. Stavové proměnné byly vytvořeny globálně. Proměnné jsem následně v průběhu ovládání aplikace měnil a podle nich větvil chování GUI aplikace. V jednotlivých menu php souborech jsem musel při načítání zohlednit stavové proměnné. To znamená, že pokud daná funkce běží, menu se musí importovat s danou třídou pro požadovaný element. Toho jsem docílil pomocí javascriptu, kde jsem testoval stav proměnné a podle toho přiřazoval elementu vhodnou třídu ještě před tím, než se nahraje nová obrazovka.

## 7.5 Specifické funkce

Individuální řešení také obsahovalo pár velice specifických funkcí, které nešly vyřešit klasickými javascript funkcemi. V menu „colours“ se jednalo o výběr barvy z kružnice barevných tónů, dále zvýšení nebo snížení barevné intenzity vybrané barvy a v menu „wellness“ animace intenzity jednotlivých programů.

Mezi tyto funkce patřilo vybrání barvy z barevného kruhu v menu „colours“. Javascript neobsahuje žádné takové funkce, a proto jsem hledal free pluginy pro technologii JQuery. Našel jsem plugin Image Color Picker, který je dostupný zde. Pro správné fungování pluginu ovšem nestačí pouze knihovna JQuery, ale musí se includovat současně i pomocný Framework pro tvorbu efektů JQuery UI. Plugin jsem testoval z důvodu pochopení funkce a následně importoval do mého projektu. Plugin potřeboval pár úprav z hlediska CSS stylů. Aktivaci pluginu jsem importoval do souboru functions.js.



Obr.č. 19 – ImageColorPicker v akci

Musel jsem ošetřit také stav, kdy funkce chromo není zapnuta a tedy výběr barvy není možný. Po vybrání barvy plugin uloží do zvoleného objektu její HEX hodnotu. Toho jsem využil pro korekci barevného tónu. Po kliku na ovládací prvky + nebo – byla vyvolána událost, kdy jsem předal funkci na korekci barvy hodnotu, která byla vybrána z barevné kružnice a ta mi vrátila výsledek nové upravené barvy. Tuto barvu jsem následně uložil do proměnné a použil pro animaci barevného tónu v ilustrační vaně. Ilustrační vana obsahuje na pozadí objekt, kterému jsem pomocí JQuery funkce `.animate()` měnil parametr `background`.

Intenzita jednotlivých programů byla animována pomocí frameworku JQuery UI, který byl použit také pro Image Color Picker plugin. Pro potřeby interakce byly vytvořeny 3 proměnné, které uchovávají hodnotu nastavení intenzity dané funkce (hydro, aero, magic) a byly nastaveny na defaultní úroveň. Při výběru požadované funkce se objeví v informační liště intenzita pro daný program (Obr.č. 20) a zároveň je aktivováno ovládání intenzity pro daný program. Po kliku na + nebo – je Javascriptem zachycena událost a změněna hodnota proměnné daného programu. Po změně hodnoty proměnné je intenzita animována pomocí JQuery funkcí `.animate()`. Jako parametr šířku funkci `.animate()` vkládám hodnotu dané proměnné a tato hodnota zastupuje hodnotu šířky ve stylech CSS.



Obr.č. 20 – Nastavení intenzity hydro

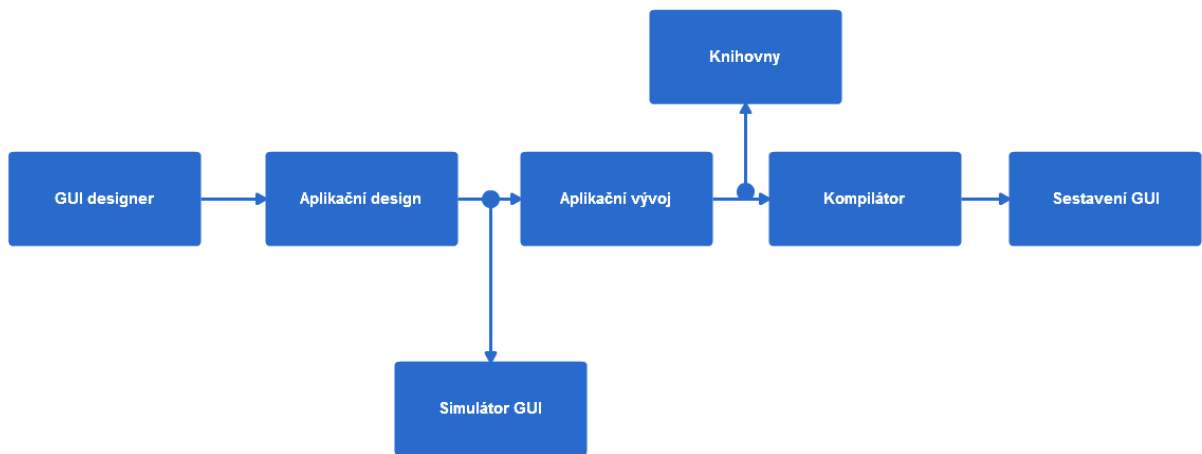
## 7.6 Výhody a nevýhody individuálního řešení

Největší nevýhoda individuálního řešení GUI je jeho časová náročnost. Ve fázi, kdy se GUI prostředí aplikace navrhuje, se často provádějí elementární změny z pohledu funkčnosti i rozmístění daných prvků. Tyto změny se jeví na pohled, že jsou velice jednoduché, ale jejich proveditelnost na daném modelu DHTML byla časově náročná. Ať už se jedná pouze o změnu pozic daných elementů nebo zvýšení počtu stavů, do kterých se daná funkce elementu může dostat. Pro přesné definování funkce GUI aplikace je zapotřebí GUI designer, který udělá tzv. frontend aplikace, ale i programátor, který následně naprogramuje požadovanou funkcionalitu aplikace na embedded systému.

Naopak mezi výhody individuálního řešení patří velká variabilita GUI aplikace. V tomto směru je možné udělat naprosto vše, což demonstruje vytvoření specifických funkcí.

## 8 KOMPONENTY PRO VÝVOJ EMBEDDED GUI

Z analýzy požadavků na tvorbu GUI pro embedded systémy a jednotlivých software jsem stanovil jednoduchý koncept komponent, které jsou nezbytné pro generátory GUI. Pro vývoj embedded GUI aplikací je tedy zapotřebí několik základních komponent. Když tyto komponenty zahrneme do jednoho software, můžeme mluvit o software pro tvorbu GUI řešení vestavěných systémů. Skládá se z GUI designeru, aplikačního designu, aplikačního vývoje, knihoven pro tvorbu GUI, kompilátoru, GUI simulátoru a fyzického embedded systému což může být modul nějakého systému, vývojový kit, mobilní telefon nebo MP3 přehrávač.



Obr.č. 21 – Komponenty aplikace pro tvorbu embedded GUI rozhraní

Mezi základní stavební kámen těchto softwarů patří GUI designery. Jedná se o komponentu, která umožňuje rychlý vývoj aplikace z grafického pohledu. V tomto editoru jsou vytvořeny jednotlivé obrazovky GUI aplikace, do kterých se vkládají ovládací prvky aplikace a to bez nějakého programování. Aplikace automaticky generuje kód, který popisuje typ zobrazení a umístění elementů na každé obrazovce vytvořeného designu. Každý GUI generátor zpracovává vizuální stránku jinak, avšak většina je založena na generování XML souboru nebo HTML a CSS souborů. Editor obsahuje funkce, které svou činností mají hodně společného s grafickými editory, typicky zarovnání objektů, vrstvení objektů a další. Hlavním úkolem GUI designeru je minimalizovat časové nároky na tvorbu aplikace a maximálně navýšit efektivitu při pozměnění logiky ovládání nebo zobrazení dané GUI aplikace. GUI designer pracuje také jako správce všech objektů vložených do GUI designu, což mohou být obrázky, tlačítka nebo další různé elementy jako např. uživatelské formuláře, grafy, dialogová okna a další.

Aplikační design je forma vývoje aplikace, kde se popisuje základní chování z pohledu přechodu obrazovek GUI aplikace a jednotlivých událostí při interakci na elementy, které se nacházejí na dané obrazovce GUI designu. Tato funkcionality je generována automaticky nebo s přičiněním uživatele. Jelikož se většinou jedná o model embedded GUI, kde jsou informace nebo uživatelské rozhraní zobrazeny na displeji nebo se jedná o mobilní systémy, nacházejí se zde události jako interakce po stlačení nebo kliku na ovládací element. U více rozvinutých systémů je možné měnit datové parametry jednotlivých proměnných a řídit tak pokročilý chod aplikace. Pro pokročilý chod aplikace a vytvoření dynamického GUI mockupu je zapotřebí nějaký skriptovací jazyk (GEM script, Javascript, Lua script).

Aplikační vývoj je programová sekce, kde dochází k programování samotných funkcionalit. Tato definice se u každého embedded systému liší a proto není možné automatické generování. Zde dochází k potřebě programátora, který pro jednotlivé události vzniklé při interakci v aplikačním designu napíše co má GUI aplikace provádět. Může se jednat v podstatě o cokoli od ovládnání motoru po signalizaci stavů aplikace pro nemocniční účely, automobilové závody nebo dotykové ovládnání domovních a spotřebních zařízení.

GUI simulátor je velice důležitým aspektem pro vývoj GUI na embedded systému. Umožňuje pohlížet na aplikace v reálném čase na cílovém zařízení nebo jen v simulátoru programu pokud cílové zařízení není ještě dokončeno a připraveno plnit požadovanou funkci.

Knihovny pro tvorbu GUI obsahují zejména potřebné ovladače, které spolupracují např. s displejem embedded systému, nebo ovladače jednotlivých řadičů, které se vyskytují na desce. Tyto knihovny jsou většinou přítomny ve formě dynamicky linkovaných knihoven s příponou „.dll“.

Kompilátor je sekce, která se stará o zkompilování a generování kódu pro embedded systém. Každý software má různý typ generování kódu v závislosti na výstupu. Některé systémy generují výsledný kód ve formě ANSI C, jiné C++ a ostatní jsou zase postaveny na platformě Java. Kompilátor úzce spolupracuje se sekcí knihoven, daty aplikace, ovladači periférií a uživatelským kódem. Po zkompilování daného projektu pošle výstupní kód do připraveného cílového zařízení. Některé aplikace obsahují také generování výstupního

souboru ve formě pro mobilní systémy, nejčastěji ve formě instalačního balíčku pro operační systém Android (.apk) nebo pro operační systém iOS.

Vyjmenované komponenty se můžou trochu lišit u každého softwaru, ale ve své podstatě je každý GUI generátor postaven na tomto konceptu. Odlišnosti se můžou vyskytovat například v knihovnách nebo kompilátoru, protože každá aplikace je postavena na jiném programovacím jazyce.

## 9 NALEZENÉ GUI DESIGNERY A GENERÁTORY

Další částí mé diplomové práce bylo nalézt vhodné GUI generátory, které by splňovaly analyzované požadavky. Při hledání těchto GUI softwarů jsem vždy navštívil domovské stránky výrobce a zde jsem zkoumal jejich funkce, které byly představeny samotným výrobcem. Pokud splňoval alespoň z větší části požadavky, které jsem analyzoval, tak jsem ho stáhl a vyzkoušel podrobně.

Několikrát se mi stalo, že jsem našel GUI generátor, který splňoval předpoklady a po nainstalování jsem zjistil, že tomu tak vůbec není. Z tohoto důvodu, který jsem předvídal, jsem veškeré práce s GUI generátory, které se instalovaly fyzicky na disk, dělal v programu VMware Player ve vytvořeném virtuálním stroji s operačním systémem Windows 8. V některých případech stažení nebylo nutné, protože GUI software byl přístupný online. Oblast GUI designerů a GUI generátorů je dnes velký byznys a tak není náhodou, že při rozsáhlém hledání jsem nenarazil ani na jeden dobrý open source. Co se týče licencí, pro mé potřeby mi vždy stačila verze trial, která nabízela zkušební dobu od 15 nebo 30 dnů.

Z několika GUI generátorů jsem vybral pouze 4. Do užšího výběru se nedostaly softwary jako Pencil, Wakanda Studio, Mockup builder, GUI design studio nebo Pidoco. Tyto softwary nabízejí vytvoření GUI mockupů, ale spíše na webové bázi, kde jde pomocí nich prezentovat formuláře a dynamické návrhy webových stránek. Nesplňují však analyzované požadavky z hlediska vizuální tvorby ani vytvoření funkcionality nebo generování a export výstupu. GUI design studio bylo vhodné pro vývoj GUI a také obsahovalo vestavěný simulátor. Jenomže nástroje pro definování chování aplikace zahrnovaly pouze změny jednotlivých obrazovek a generování nebo exportování aplikace zde vůbec nebylo. Podobné to bylo i u webové aplikace Pidoco, která se zaměřuje na mobilní systémy a tvorbu GUI přímo pro android a iOS. Pidoco obsahovalo vestavěný simulátor i umožňovalo export do HTML, avšak zde chyběly naprosto nástroje, kterými by bylo možné vytvořit dynamický GUI návrh (obsahovalo pouze linkování mezi obrazovkami).

### 9.1 Fore UI

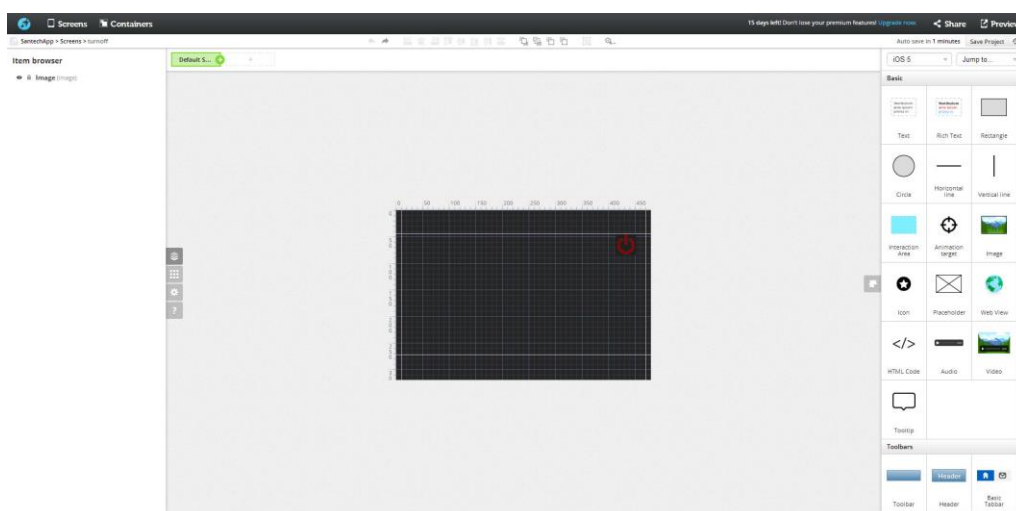
Fore UI je modelovací nástroj pro tvorbu GUI mockupů. Je plně skinovatelný i s jeho prvky, je tedy možné přepínat vzhledy jednotlivých elementů ( tlačítek, posuvníků, oken, dialogových oken atd.) přímo do dané platformy, pro kterou ji tvoříme. Obsahuje pokročilé



netbooku nebo tabletu. Proto.io se zaměřuje zejména na mobilní systémy a tudíž obsahuje veškeré prvky (status bar atd.), které jsou potřeba pro vývoj mobilních GUI nacházející se v knihovnách UI. Aplikace disponuje manažerem jednotlivých obrazovek a funkcí, které zajistí funkcionalitu GUI mockupu. Dají se zde vytvářet vlastní elementy, které jdou pak jednoduše importovat do vytvořených obrazovek. [17]

Instalace zde není, je provedena pomocí zaregistrování a přidělení webového uložení a projektu. Adresu projektu si volí uživatel sám zvolením názvu projektu, ke kterému se přiřadí koncovka „proto.io“. Proto.io obsahuje také export vytvořené GUI aplikace do HTML5. Aplikace Proto.io splňuje z větší části požadavky na tvorbu GUI, dobře je zde vyřešená možnost správy uživatelských proměnných a možnosti simulování projektu.

Proto.io je placená služba, avšak nabízí trial verzi k vyzkoušení trvající po dobu 15 dní. Licence je rozdělena do několika oblastí freelancer (24\$), startup (40\$), agency (80\$), corporate (160\$). [17]



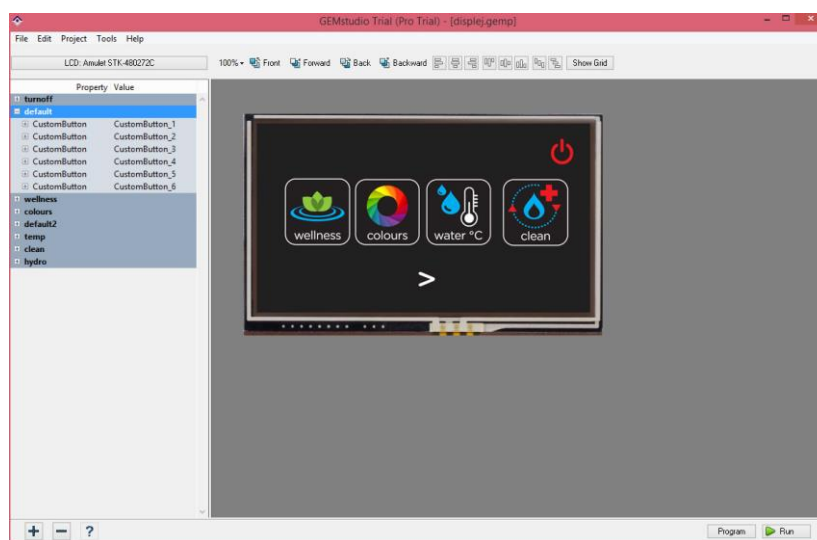
Obr.č. 23 – Aplikace Proto.io

### 9.3 GEM studio Pro

GEM studio Pro je skutečný WYSIWYG GUI Design softwarový nástroj, který nevyžaduje žádné znalosti kódování. Ve srovnání s jinými řešení GUI designu, které často vyžadují další vývojářské nástroje s C++ znalostí je tak snadno použitelné. Program obsahuje také svůj vlastní skriptovací jazyk GEMscript, který umožňuje vytvářet specifické prvky uživatelského rozhraní. Na levé straně se nachází globální panel, který umožňuje vytvářet

nové obrazovky i s přidáváním elementů. Jednotlivé události a funkce GUI aplikace jsou řízeny jazykem GEM script. Výrobce nabízí k programu také vývojové kity a desky, na kterých se nachází Amulet OS, který zajišťuje komunikaci s programem a poskytuje realtime prezentaci GUI aplikace přímo na embedded zařízení. Samozřejmostí je také vestavěný simulátor, který umožňuje testovat vytvořenou aplikaci bez nutnosti připojení embedded zařízení. Program GEM studio nabízí v základu několik nastavení displejů, včetně jeho rozlišení a typů podpůrných desek a modulů. Program je placený, nabízí však trial verzi po dobu 30 dní. [13]

Trial verze GEM studio Pro je dostupná ke stažení z domovských stránek výrobce v sekci Products/GEM studio Pro. [32]



Obr.č. 24 – GEM studio Pro

GEM studio Pro splňuje veškeré analyzované požadavky. Jedinou nevýhodou je definice pokročilého chování aplikace, která zde není moc dobře uživatelsky vytvořená. Výhodou je generování kódu pro cílové zařízení a správa a nastavení jednotlivých embedded zařízení.

## 9.4 Crank Storyboard Suite 3.2

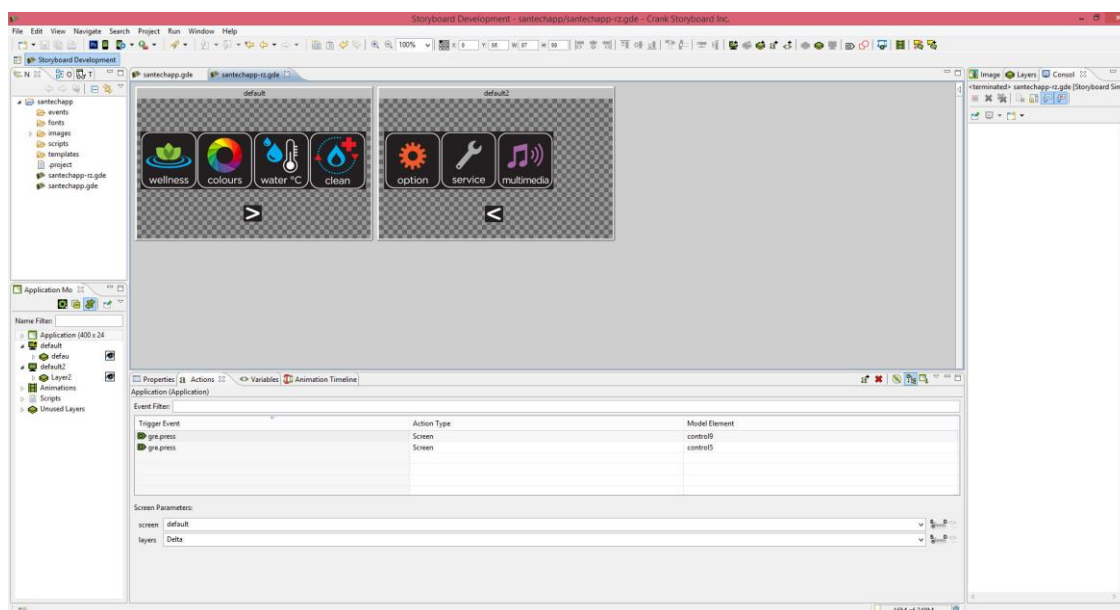
Crank Storyboard je profesionální nástroj používaný pro vývoj GUI návrhu pro embedded systémy. Jak je z uživatelského prostředí patrné, je založen na jádře vývojového prostředí eclipse. Po stránce uživatelského rozhraní je velmi intuitivní a přehledný. Nabízí mnoho výjimečných funkcí. Jednou z nich je import grafiky přímo z PSD souboru i s import názvů

jednotlivých obrázků. Obsahuje také přehledného správce obrazovek. Je založen na vrstveném modelu, tzn. že pro každou obrazovku musí být vytvořena alespoň jedna vrstva. Obsahuje pokročilé funkce pro animování elementů a 3D objektů a umožňuje tak tvořit zajímavé GUI aplikace. Generování aplikace je umožněno jak pro mobilní systémy tak embedded systémy. Pro mobilní systémy generuje výstupní aplikaci jako instalační balíček pro android (.apk) a také pro systém iOS. Pro embedded systémy generuje tzv. storyboard embedded engine (.see). [14]

Definice chování GUI aplikace je možné tvořit pomocí dialogového okna, kde si uživatel může vybrat událost, akci a parametry akce, které se mají provést. Samozřejmostí je simulátor v dialogovém okně, který nabízí pohled na vytvořenou aplikaci včetně její funkcionality. Pro vytvoření dynamické GUI aplikace s přičiněním uživatele je zapotřebí vytvoření LUA scriptů, které se poté volají jako funkce v parametrech akce. Program umožňuje vytvoření také lokalizací GUI aplikace skrze LUA skripty.

Vytvoření animací je zde velice podobné jako u Adobe Aftereffects. Animace se provádějí v záložce „animation timeline“. Při animaci jde operovat s více elementy zároveň.

Crank Storyboard splňuje celkově všechny požadavky kladené na GUI generátor. Obsahuje exporty na cílové zařízení a uživatelské prostředí a správa chování aplikace je zde velice dobře vyřešena.



Obr.č. 25 – Uživatelské prostředí Crank Storyboard

GUI generátor je placený software a jedna uživatelská licence je velice drahá, cena se pohybuje okolo 6000\$. I když je cena softwaru velice vysoká pro začínající firmy zabývající se vývojem GUI by to mohla být dobrá investice, protože program toho nabízí opravdu hodně. Software je také dostupný v podobě trial verze trvající 15 dnů. [14]

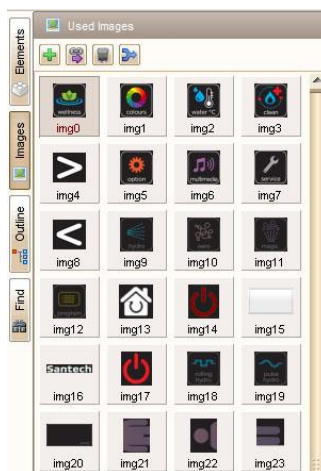
Trial verze Crank Storyboard Suite je dostupná ke stažení z domovských stránek výrobce. Nejdříve se vyplní registrační formulář a automaticky vygenerovaným emailem je poslán odkaz ke stažení. [33]

## 10 DEMONSTRACE APLIKACE V GUI GENERÁTORECH

Jelikož jsem našel vyhovující generátory GUI, zaměřím se pouze na to demonstrovat jejich funkce a možnosti na vytvořené DHTML aplikaci. V této sekci popíši, jak jsem vytvářel dema v jednotlivých GUI generátorech a poukážu na problémy při tvorbě a jejich individuální řešení.

### 10.1 Fore UI

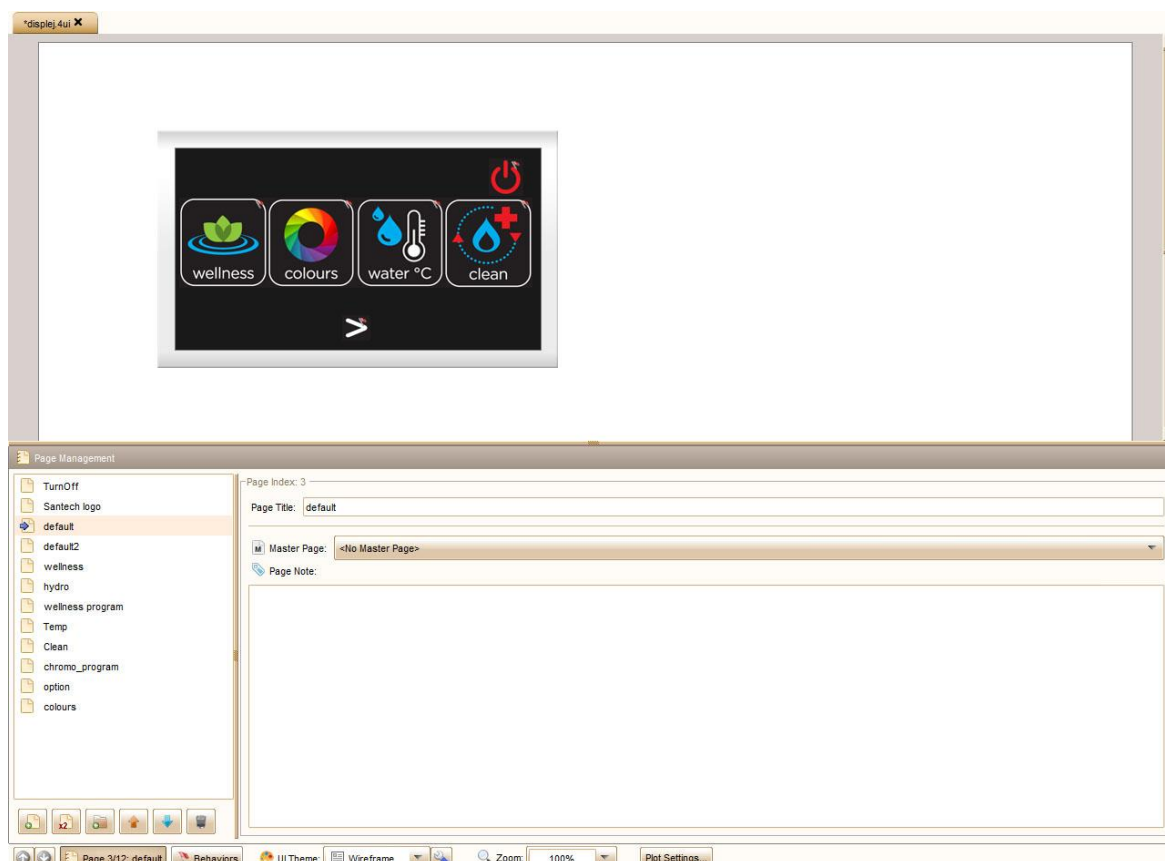
Prvotní část tvorby byla dostat nařezanou grafiku z individuálního řešení do projektu. Na pravé straně jsem zvolil panel obrázky, který umožňuje import obrázku do projektu. Po kliku na přidání obrázku se zobrazilo dialogové okno s adresářovou strukturou. Vybral jsem tak jednoduše veškeré obrázky, které jsem potřeboval a importoval je. Program podporuje všechny známé formáty obrázku od BMP, PNG, JPEG a GIF a proto s importem nebyl žádný problém. Užitečnou funkcí je optimalizování využití obrázků. Funkce zajišťuje odstranění duplicit a nevyužitých obrázků.



Obr.č. 26 – Import obrázku do projektu

Z galerie importovaných obrázků jsem poté jednoduše obrázky přenášel pomocí myši do jednotlivých obrazovek metodou „drag and drop“.

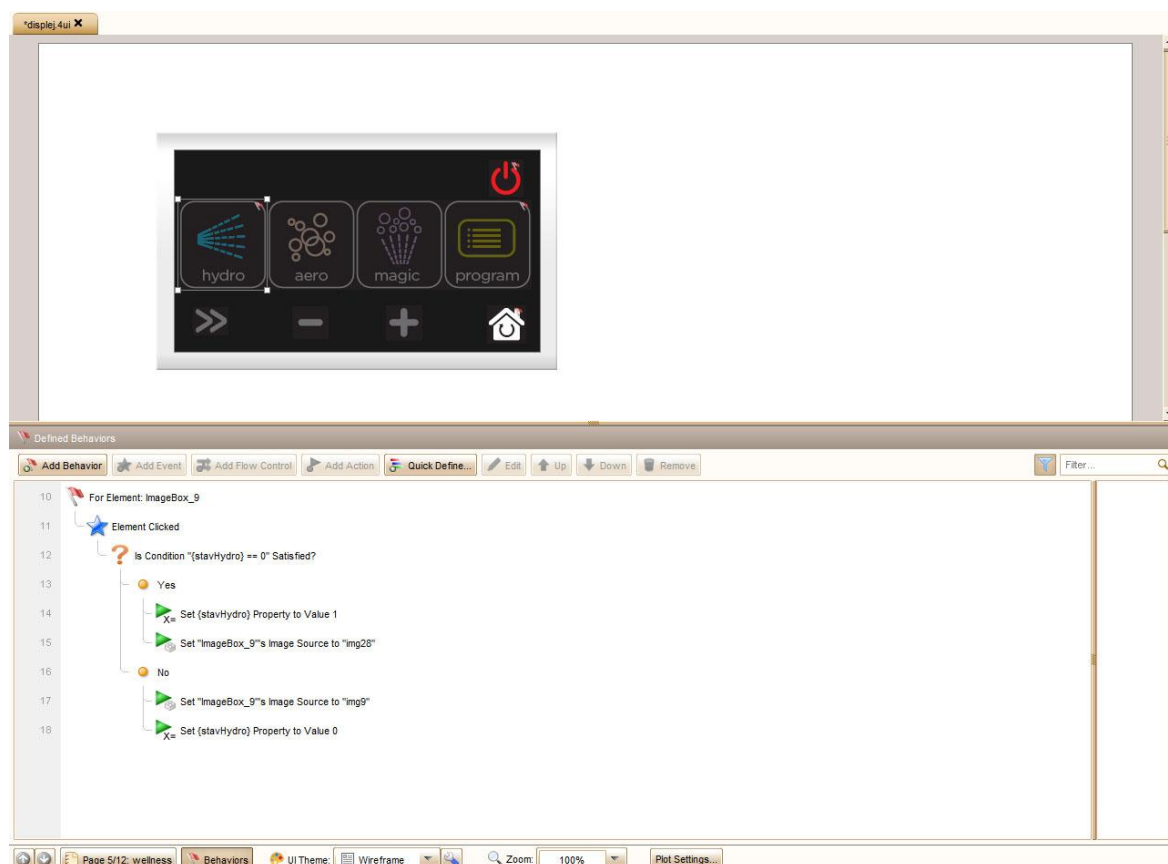
Základ ovladače tvoří obrazovky. Tvorba obrazovek ve Fore UI probíhala jednoduše. Vytvořil jsem projekt a následně jsem pomocí ovládacího panelu „Page management“ vytvořil jednotlivé obrazovky. Pro přehlednost jsem obrazovky rovnou pojmenoval. Počet obrazovek není nijak omezen.



Obr.č. 27 – Tvorba obrazovek

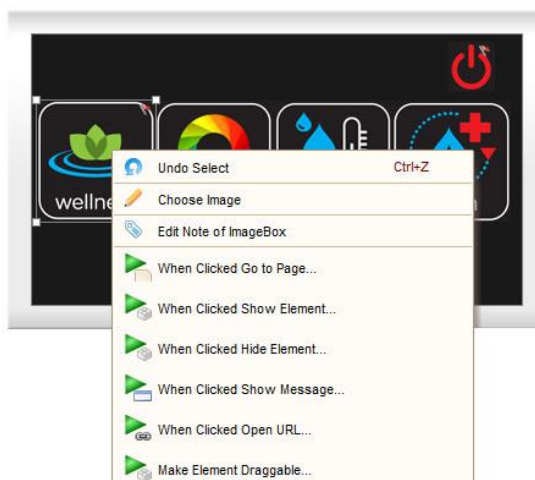
Fore UI neobsahuje žádný příznak obrazovky, od které se má začít, a proto pro správné zobrazení aplikace je důležité, aby startovní obrazovka byla na prvním místě. Chtěl bych vyzdvihnout chytrou funkci duplikování obrazovky, protože při duplikování jsou zachovány všechny události a akce elementů, které se na obrazovce nacházejí a tak výrazně šetří čas. Podobné obrazovky ovladače jsem jednoduše zkopíroval a změnil pouze elementy, ve kterých se lišily nebo přidal chybějící.

Po vytvoření všech potřebných obrazovek jsem se vrhl na chování aplikace. Pro úpravu chování aplikace slouží panel „Behaviours“. Ten obsahuje funkce, které jsou vizuální obdobou klasických javascript funkcí. Definice chování spočívala ve výběru daného elementu a přiřazení události, která má být udělaná při interakci s tímto objektem. Program obsahuje klasické události jako klik, dvojklik, klik pravým tlačítkem myši, přejetí myši nad objektem a poté z objektu pryč a dále umožňuje vytvářet vlastní události. Pro potřeby prezentace mi vystačili klasické události. Tímto způsobem jsem definoval události a akce pro každou obrazovku.



Obr.č. 28 – Definice chování elementu při kliku

Přechody mezi obrazovkami byly vcelku jednoduše ošetřeny událostmi při kliku na element. Pravým tlačítkem myši jsem vyvolal popup menu, kde jsem zvolil při kliku přejít na obrazovku. Program zobrazil seznam dostupných obrazovek a tak jsem jednoduše bez psaní zvolil požadovanou obrazovku. Po zvolení obrazovky se automaticky do panelu chování přidala stromová struktura s informací o jaký element, událost a akci se jedná. Nebylo tak nutné ručně naklikávat akce. Ne vždy to tak šlo, pro komplexnější chování s využitím podmínek jsem musel tyto parametry sám vyplňovat.

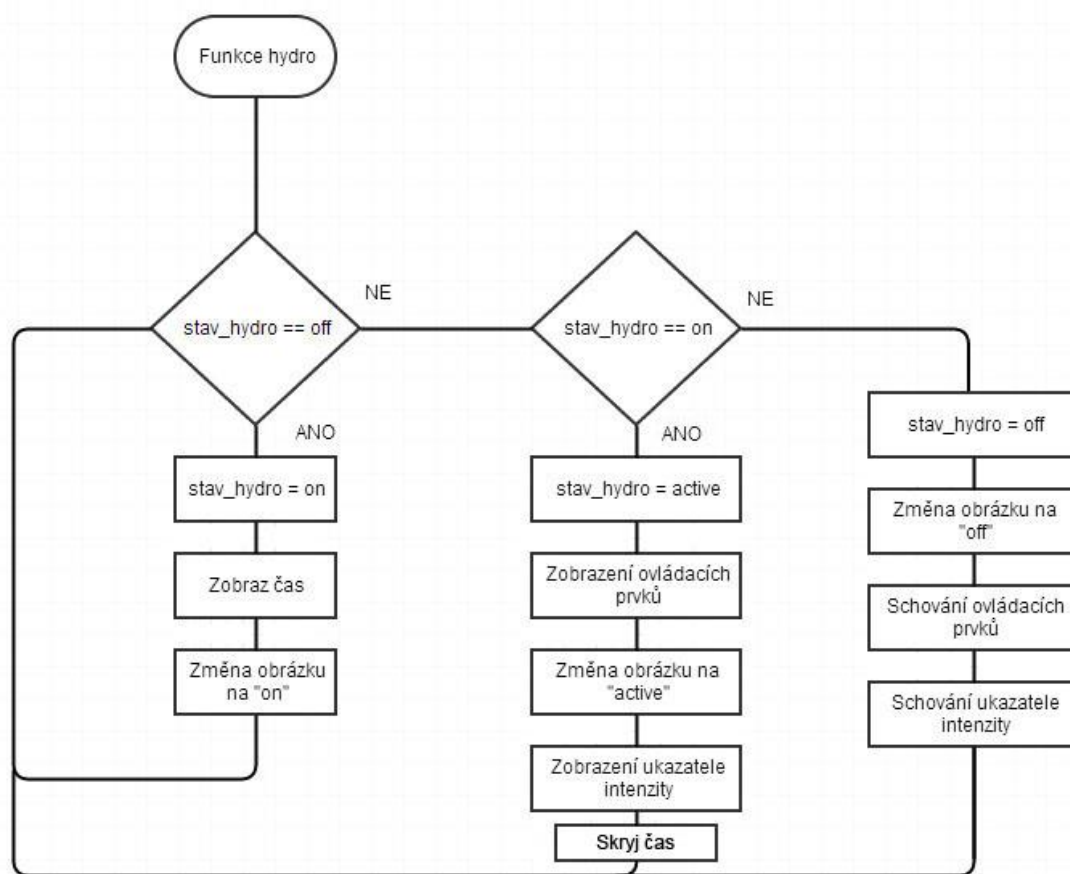


Obr.č. 29 – Přechod mezi obrazovkami

Stavové proměnné jsem dělal již při individuálním řešení a nevyhnul jsem se tomu ani tady, protože komplexnější funkce jako menu wellness a jeho programy si to vyžadovalo. Obdobně jako u individuálního návrhu jsem vytvořil 3 stavové proměnné pro funkci hydro, aero a magic a nastavil jim výchozí hodnotu na „off“. Tyto proměnné jsem vytvořil v postranním panelu „custom property“. Nejprve jsem chtěl pro přehlednost vytvořit proměnnou jako string, ale ukázal se problém v simulaci. Větvení pomocí stringu nefungovalo korektně, a proto jsem vytvořil proměnnou typu integer a stavy jsem nastavil následovně:

- Off = 0
- On = 1
- Active =2

V definování chování GUI aplikace jsem využil větvení pomocí funkce akce řízení toku chování aplikace. Zde jsem vybral klasickou podmínku if. Když byla splněna podmínka, tak jsem přidal akci nastavení hodnoty do vytvořené stavové proměnné (set custom property). Zároveň jsem přidal akci „operate an element“ pomocí, které jsem změnil zdroj obrázku pro tlačítko. Pokud podmínka nebyla splněna, následovala poslední podmínka a v závislosti na ní jsem přiřadil opět akce pro úpravu proměnné a úpravu elementu. Zároveň jsem zobrazil ovládací prvky a ukazatel intenzity pro danou funkci. Pro lepší pochopení jsem si vytvořil vývojový diagram, který řeší ovládání elementu hydro.



Obr.č. 30 – Vývojový diagram pro funkci hydro

Ovládání elementů aero a magic bylo uděláno skoro identicky, pouze se zobrazovaly ovládací tlačítka a ukazatel intenzity pro jinou funkci.

Ovládání teploty jsem vyřešil za pomoci vytvoření proměnné, která uchovávala nastavenou teplotu. Jako typ proměnné jsem zvolil integer. Dále jsem vytvořil funkce, které zachycovaly událost při kliku na tlačítka + a -. Jediným účelem těchto funkcí bylo zvýšit nebo snížit hodnotu proměnné teplota a v dalším kroku nastavit aktuální hodnotu proměnné jako text teploty.

Program Fore UI má vestavěný simulátor a nebylo tedy nutné s každou provedenou změnou exportovat celý projekt do HTML. Stačilo spustit simulaci a v defaultním webovém prohlížeči se automaticky spustila simulace s vytvořeným mockupem nebo GUI aplikací. Načítání simulace je velice rychlé a srovnatelné s načítáním webové stránky. Při tvorbě každé obrazovky včetně její funkcionality jsem často využíval simulátor pro odzkoušení správného chodu.

Při tvorbě se mi nepodařilo udělat odpočet času při spuštění programu. Tato funkce je nepodstatná a zobrazil jsem tedy pouze informační čas jako statický text. Další funkce, kterou nebylo možné v programu Fore UI udělat, byl výběr barvy z barevné kružnice.

Po vytvoření kompletní funkční GUI aplikace jsem přistoupil k exportu aplikace. Export vytvořené GUI aplikace byl velice jednoduchý, využil jsem funkci export do HTML. Vyskočilo dialogové okno s dotazem na cestu. Po zadání cesty byl celý projekt exportován do dané složky. Program sám vytvořil adresářovou strukturu, HTML soubor, CSS styly, které obsahují pozice a rozměry daných elementů a dále Javascript a JQuery soubory, které popisují chování GUI aplikace.

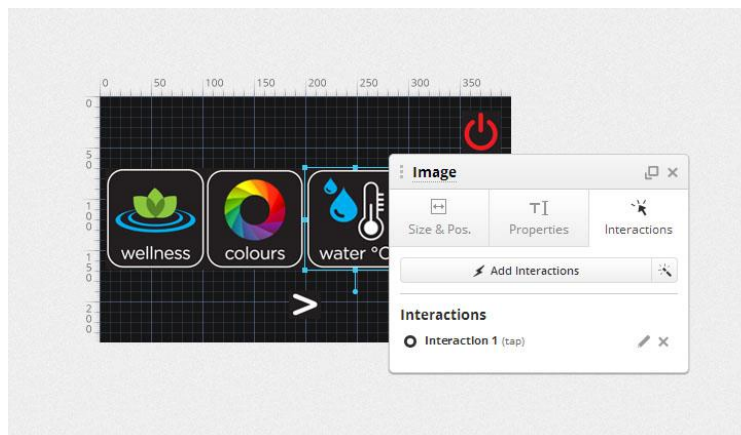
Když bych porovnal tvorbu aplikace v programu Fore UI a individuální návrh, zde byla doba potřebná k sestavení aplikace výrazně nižší. Pokud bych vzal v potaz ještě čas, který byl potřebný k zorientování v softwaru a vyzkoušení funkcí, tak úspora byla více než  $\frac{3}{4}$  doby strávené na individuálním návrhu. V případě dalších úprav logiky by zde nebyl nejmenší problém a byla by to otázka několika minut.

## 10.2 Proto.io

Po zaregistrování trial verze jsem si vytvořil prázdný projekt. Software nabízí i vytvoření z několika předefinovaných šablon, pro mé účely mi postačil prázdný. Před vytvořením projektu jsem si definoval svoje zařízení na rozměry GUI aplikace 400x240px a pojmenoval jsem ho „bathcontrol“. Následně při vytváření projektu jsem vybral vytvořené zařízení. Po vytvoření projektu jsem si nahrál veškeré zdrojové obrázky do složky images, kterou jsem si vytvořil. Nahrání probíhá skrze upload dialogové okno. Po nahrání všech potřebných obrázků jsem začal s vizuální stránkou aplikace. Přes tlačítko „screens“ jsem si vytvořil základní obrazovku s definovaným pozadím. Následně jsem veškeré obrazovky vytvářel pomocí funkce duplikování aktuální obrazovky a pouze jsem změnil název a elementy, které se na stránce vyskytovaly. Ušetřil jsem tak spoustu času, který bych strávil na pozicování fixních elementů.

Propojení obrazovek bylo realizováno pomocí funkce „interactions“. Tato funkce byla dostupná při kliku na element, kdy se zobrazí popup okno s rozměry elementu, popřípadě zdrojem obrázků a interakcemi. Tlačítkem „add interactions“ jsem přidal události a akci

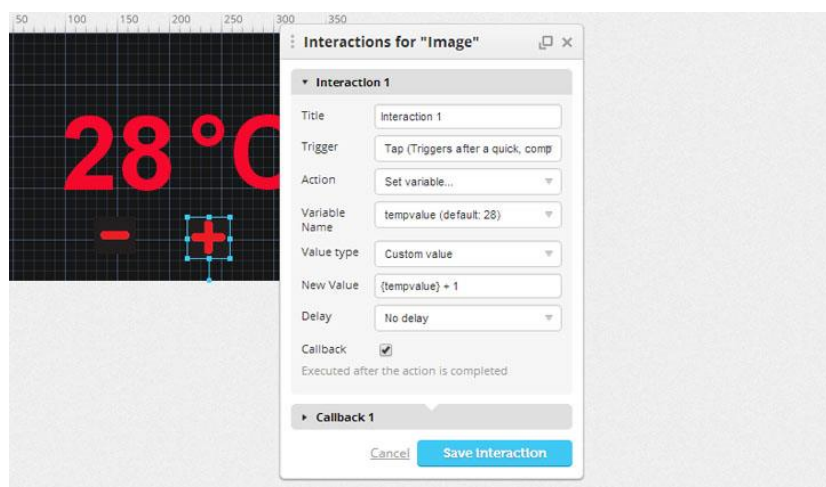
přechodu obrazovky, kde jsem následně vybral obrazovku, na kterou se má přejít. Přidal jsem také nějaký efekt přechodu jako prolnutí nebo slide.



Obr.č. 31 – Vlastnosti elementu aplikace proto.io

Pro potřeby vytvoření dynamické části GUI aplikace v obrazovce temp jsem využil „variable manager“ a vytvořil uživatelské proměnné. Pro obrazovku temp mi vystačila pouze jedna a to hodnota teploty. Vytvořil jsem tedy proměnnou s počáteční hodnotou 28 a dal jsem jí typ numeric. Důležitým aspektem bylo zatrhnutí checkboxu „evaluate“. Ten zajišťoval, že jsem mohl měnit její hodnotu při jednotlivých akcích.

Pro tlačítka + a – jsem vytvořil interakci. Jako událost jsem zvolil „tap“, ta nahrazuje funkci onclick. Akci jsem vybral nastavit proměnnou a z roletkového menu jsem vybral vytvořenou proměnnou tempValue. Pro změnu hodnoty jsem název této proměnné dal do složených závorek a zvýšil její hodnotu o 1.

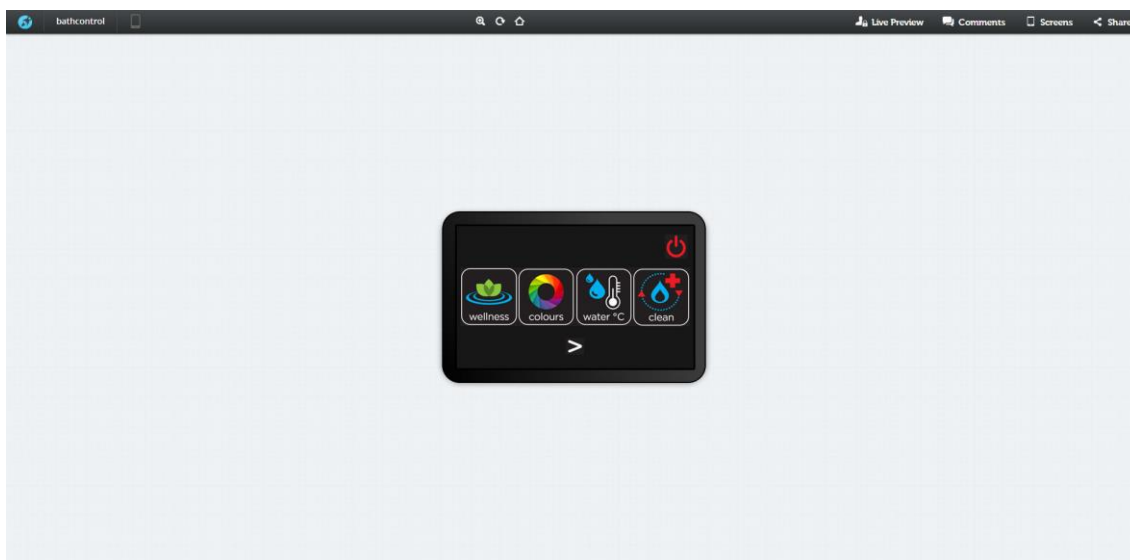


Obr.č. 32 – Úprava hodnoty proměnné

Po vykonání této akce bylo zapotřebí ještě nastavit textu aktuální hodnotu proměnné. Využil jsem funkce callback, která zajišťovala provedení po úspěšném nastavení proměnné. V callback interakci jsem vybral jako akci nastavení vlastnosti textu, který zobrazoval hodnotu teploty a z roletkového menu jsem vybral možnost číst z proměnné a zvolil nově nastavenou proměnnou.

Pro realizaci programů v menu wellness jsem vytvořil vždy 3 elementy, které reprezentovaly stavy off, on a active. Stav „on“ a „active“ jsem vždy schoval, aby ve výchozím menu byly pouze stavy „off“. Po kliku na „off“ element jsem zobrazil ostatní elementy pomocí funkcí „show/hide items“. Při ovládání intenzity programů jsem opět využil uživatelské proměnné jako v případě obrazovky temp, avšak zde jsem nastavil proměnnou jako délku elementu obdélníku daného programu. Pro tuto akci jsem vybral možnost animace programu 500ms zpožděním.

Testování funkčnosti aplikace jsem provedl pomocí simulátoru, který je dostupný v pravém horním rohu pod tlačítkem „preview“. Pro korektní funkčnost jsem musel vždy uložit projekt tlačítkem „save“.



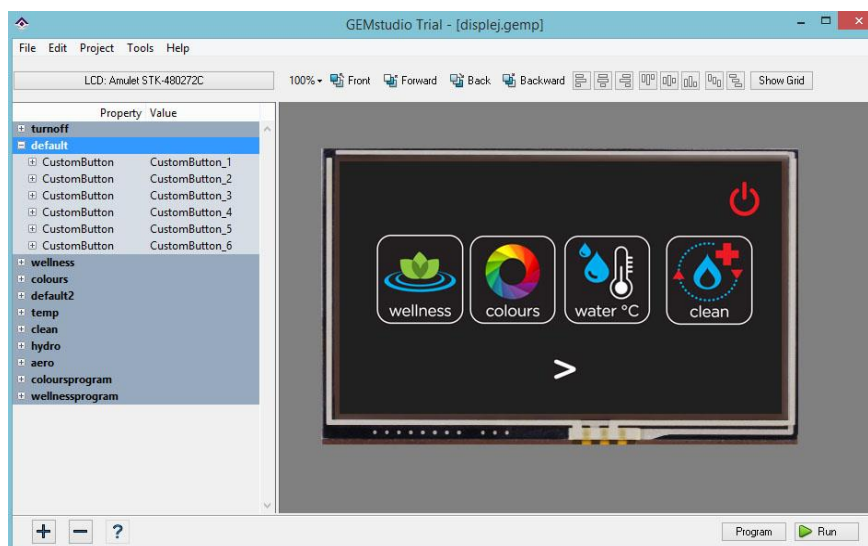
Obr.č. 33 – Simulátor aplikace Proto.io

Po dokončení všech funkcí GUI aplikace jsem použil funkci exportu do HTML. Tato funkce je dostupná v dropdown menu pod tlačítkem „share“. V dialogovém okně jsem potvrdil a zanedlouho byl připraven balíček ve formátu zip s vygenerovaným výstupem ke stažení.

Oproti individuálnímu návrhu byla tvorba v Proto.io radost. Tvorba vizuální stránky zde byla velice rychlá a neměla žádnou chybu. Definování chování zde bylo také dobře vyřešeno a uspořilo mnoho času, jedinou nevýhodou bylo, že jsem stejné akce musel definovat vícekrát a nešlo tyto akce duplikovat.

### 10.3 GEM studio Pro

Tvorba obrazovek byla provedena v levém panelu. Tlačítka + a – jsem přidával nebo mazal jednotlivé obrazovky. Po vytvoření obrazovky jsem ji rovnou pojmenoval. Zde se nenachází žádný příznak startovní obrazovky a tak jsem musel jako ve vrstvách v grafickém programu startovní obrazovku přetáhnout na první pozici v panelu obrazovek. Podobné obrazovky jsem duplikoval pomocí kontextové nabídky a funkce „duplikace this page“.



Obr.č. 34 – Tvorba obrazovek GEM studio Pro

Obsah obrazovek jsem doplnil pomocí elementu „Custom button“, který umožňoval vložit jako speciální parametr obrázek. Veškerá funkcionalita se prováděla přes parametr href objektu „custom button“. Po kliku vyskočilo prázdné okno s akcemi. Prostudoval jsem GEM script a pomocí jednoduchých definicí jsem podobně jako ve ForeUI a Proto.io vytvořil vždy 2 objekty pro požadované stavy a poté jsem je zobrazil nebo schoval. S tvorbou obrazovek problém nebyl. V programu GEM studio Pro se mi nepodařilo udělat dynamický ukazatel intenzity pro jednotlivé programy ani ovládání teploty. Funkce GEM scriptu mi přijdou chaotické a mohly být vyřešeny mnohem lépe. S textem je to stejné.

Jediný text je zde ve formě widgetu „stringfield“, avšak manipulace ani prvotní nastavení textu není snadné. Zkoušel jsem realizovat ovládání teploty, avšak pokaždé při kompilování GUI návrhu se vyskytla chyba u tohoto objektu. Přechody jednotlivých obrazovek jsem také prováděl v parametru „href“ u ovládacího prvku. Z roletkového menu stačilo vybrat název obrazovky a přidat parametr „.open()“, který zajistil načtení nové obrazovky.

## 10.4 Crank Storyboard Suite 3.2

Po seznámení s prostředím jsem vytvořil nový projekt/aplikaci. Zde jsem zvolil rozměry displeje 400x240px. Zajistil jsem tak, že každá nová obrazovka a vrstvy budou mít požadované rozměry.

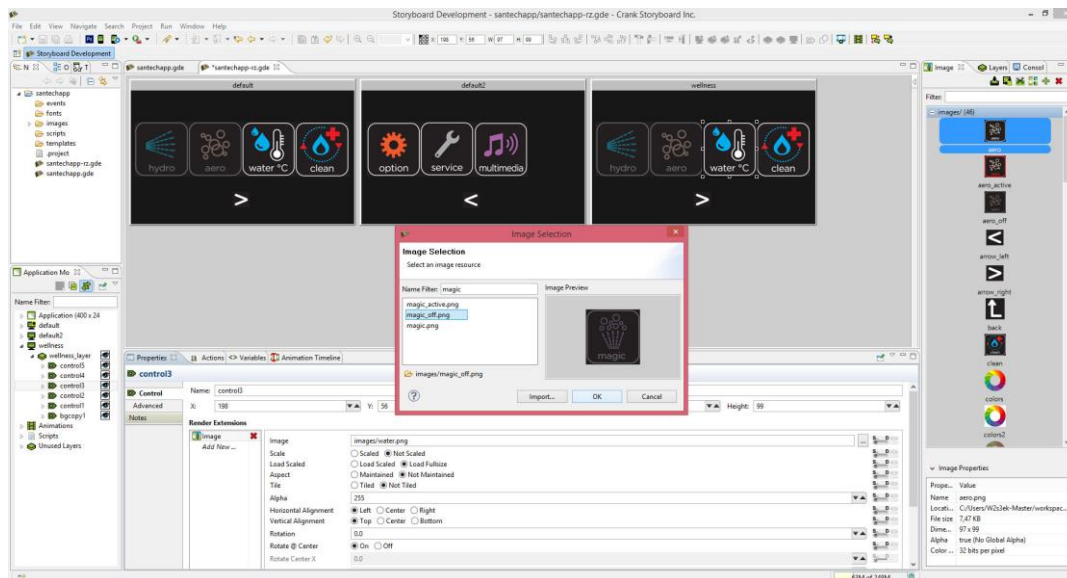
Import obrázku do projektu byl proveden přes pravý ovládací panel. Zvolil jsem záložku „Image“ a přes ikonku plus jsem importoval veškeré zdrojové obrázky do projektu. Po importu je k dispozici náhled ke každému obrázku a jejich umístění v jednotlivých obrazovkách jsem provedl přetažením na danou obrazovku. Ušetří se spousta času vytvářením elementů přes dialogové nabídky. V průběhu projektu se dá jednoduše obrázek změnit přes záložku „properties“ ve středovém panelu, kdy najede dialogové okno s výběrem obrázku i s náhledem.



Obr.č. 35 – Panel image

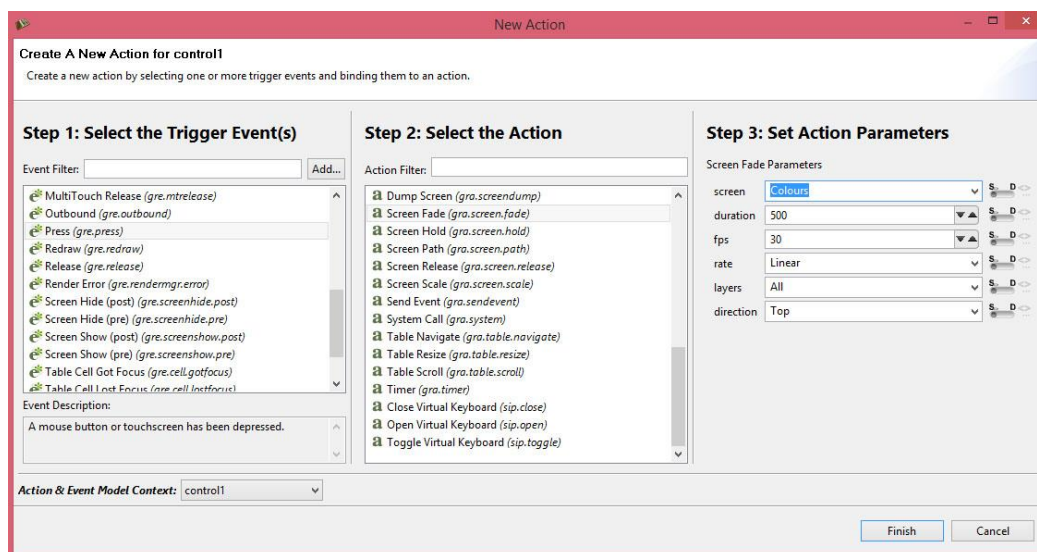
Tvorbu obrazovek jsem provedl v levém dolním panelu nazvaném „application model“. Vytvořil jsem obrazovky (screens) a rovnou je pojmenoval. Abych mohl vkládat jednotlivé elementy do obrazovky, byl jsem nucen vytvořit i vrstvu pro každou obrazovku. Název vrstvy není důležitý, tak jsem nechal implicitní, ale při vytvoření více vrstev v rámci jedné obrazovky je pojmenování nutné pro snadnější výběr akcí. Tvorba aplikace je založena na vrstveném systému, tudíž není problém s překrýváním objektů. Při tvorbě první obrazovky

jsem zatrhl možnost „starting screen“ a definoval tak startovní obrazovku, od které se má začít.



Obr.č. 36 – Tvorba obrazovek v Crank Storyboard

Přechody obrazovek jsem realizoval pomocí definování akcí a událostí. Tyto akce a události se dají vytvářet pomocí kontextové nabídky vyvolané pravým tlačítkem myši nebo v prostředním panelu v záložce actions. Obě metody mají společné dialogové okno, které obsahuje 3 kroky, které jsem musel provést. Definovat událost, definovat akce a zvolit parametry dané akce. Pro potřeby změny obrazovky a interakci s elementy jsem vždy vybral událost „Press“, která napodobuje událost „onclick“ u javascriptu. Dále jsem vybral akci „Screen fade“. Jako parametry akce jsem nastavil obrazovku, na kterou se má přejít, další parametry jako dobu trvání jsem ponechal nezměněné.

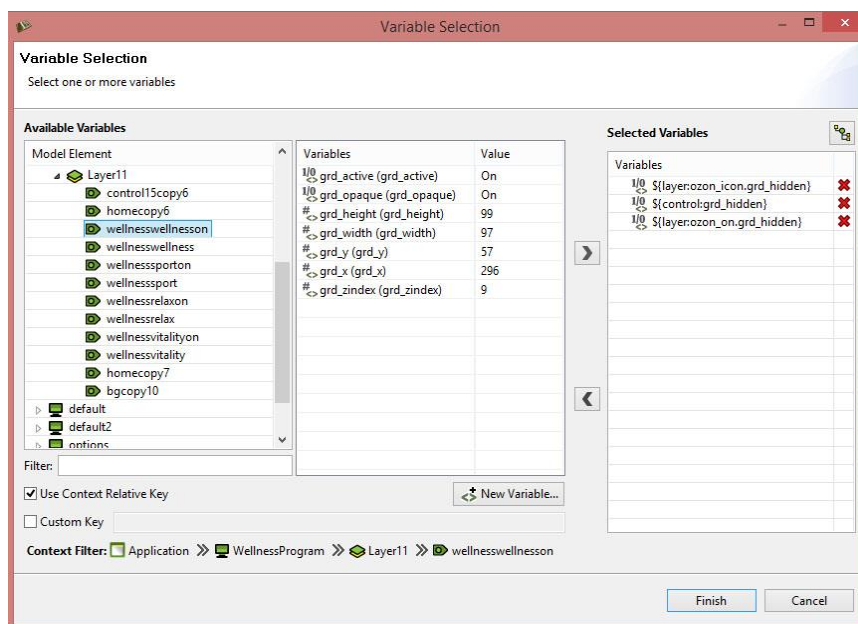


Obr.č. 37 – Přejít mezi obrazovkami

Vytvořené akce jsem následně libovolně upravoval přes středový ovládací panel v záložce „actions“. Tímto způsobem jsem provázal logiku ovládání jednotlivých obrazovek a elementů.

Jednotlivé funkce ovladače jsem pojnal trochu jinak, než tomu bylo u programu Fore UI, který nabízel větvení. Zde jsem vytvořil pro každý stav programu hydro, aero a magic nový ovládací element, který jsem pojmenoval podle stavu a elementy, které se nemají zprvu zobrazovat, rovnou schoval. Funkci jsem následně provedl po kliku na element a vždy jsem element aktuálního stavu skryl a zobrazil nový. Nebyla tedy vytvořena jedna akce pro všechny stavy, ale naopak pro všechny stavy byla nastavena událost s jednotlivými akcemi a změnou parametrů.

Pro jednoduchou změnu dvou stavů jsem využil možnosti vytvoření dynamické proměnné. Touto proměnnou může být v podstatě cokoli. V mém případě jsem vytvořil dynamickou proměnnou zdroj obrázku, abych jej mohl měnit v nastavení parametrech akce při zvolení metody „data change“. Tohoto jsem využil v menu „Clean“ pro správu programů auto a manual. Po kliku na manual jsem vyměnil zdroj obrázku na „active“ a elementu auto jsem změnil zdroj obrázku na „off“. U programu auto to bylo zase obráceně. Tím jsem docílil jednoduchého přepínání mezi auto a manual. Ostatní menu byly vytvořeny identicky s výše popsanými postupy.



Obr.č. 38 – Příklad akce data change

Pro dynamické ovládání teploty na obrazovce „water“ jsem využil funkce lua script. Vytvořil jsem ve složce script nový lua script a zde jsem si zadefinoval nové proměnné, se kterými budu pracovat. Definice proměnných je provedena pomocí syntaxe „local název\_proměnné = hodnota“. Ve vizuálním editoru jsem vytvořil pro můj text zobrazující teplotu dynamickou proměnnou s typem string a ponechal jsem ji implicitní hodnotu (28). Ve vytvořeném lua scriptu jsem vytvořil funkci pro zvýšení a snížení teploty. V těle funkce jsem provedl navýšení nebo snížení teploty a pomocí kontextové nabídky jsem zkopíroval ve vizuálním editoru cestu dané proměnné a vložil do parametru, který upravuje data. Ve vizuálním editoru jsem pak jednoduše nabíndoval funkce na tlačítka + a – pomocí události „press“ a výběr lua scriptu. V parametrech lua scriptu jsem vybral z roletkového menu vytvořenou funkci.

Lua skripty jsem využil i v obrazovce „wellness“ pro animování velikosti ukazatele intenzity programů. Postup byl obdobný jako u teploty, zde jsem však musel pro parametr upravující data přidat text k cestě objektu. Přidal jsem parametr „grd\_width“, kterým jsem zajistil, že budu dynamicky zvětšovat velikost ukazatele intenzity. Ve funkci jsem ošetřil případy, aby ukazatel intenzity nepřetékal pozadí a nebyl nastaven na 0. Nabíndování funkcí bylo provedeno stejně jako v předešlém případě. Samozřejmostí bylo vytvoření nových funkcí.

```

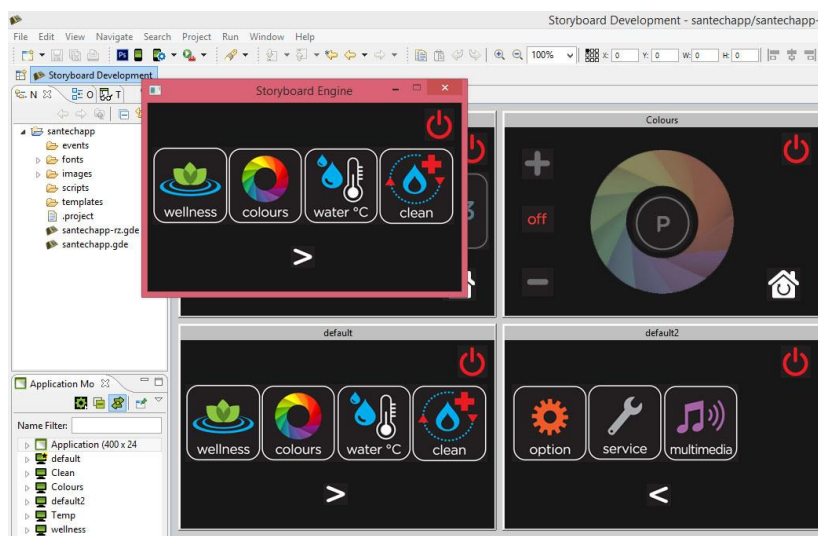
function increaseMagic(cur_width)
    local data = {}
    local width = 0
    if (intmagic < 270) then intmagic = intmagic + 10
    width = intmagic
    else width=270 end
    data["Intensity_controls_magic.Intensitycopy2.grd_width"] = width
    gre.set_data(data)
end

```

Obr.č. 39 – Příklad lua scriptu animace ukazatele intenzity

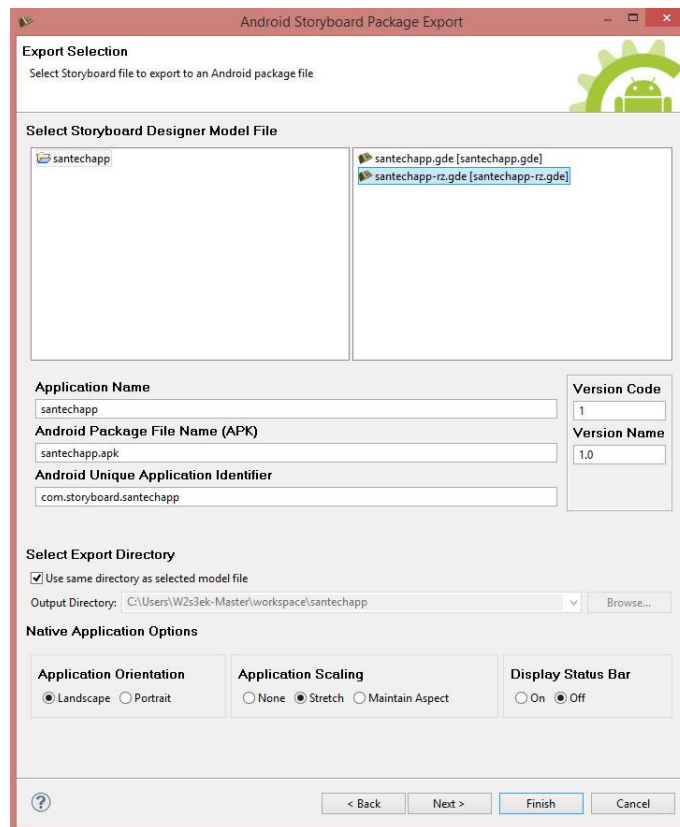
Pro odzkoušení funkčnosti aplikace jsem používal vestavěný simulátor, který běží na Storyboard engine. Je přístupný v horní ovládací liště ve formě ikonky mobilu. Výbornou funkcí je detekce změn provedených v navrhované aplikaci a tudíž simulátor se nespustí, dokud nejsou uloženy všechny změny. Po spuštění simulátoru se zobrazí dialogové okno s vytvořenou aplikací.

Jelikož v menu „option“ se nastavují jednotlivé globální hodnoty i zde byl použit lua script. Byly opět vytvořeny nové funkce a nabindování funkcí bylo provedeno analogicky jako výše.



Obr.č. 40 – Spuštěný simulátor v akci

Po dokončení testování hotové aplikace jsem přešel k vygenerování výstupu. Jelikož jsem neměl podpůrné desky, které výrobce také nabízí, rozhodl jsem se vygenerovat výstup jako nativní aplikaci pro android a prezentovat ji na smartphonu. Toho jsem docílil pomocí kontextového menu File/Export, kde jsem vybral možnost nativní android aplikace a potvrdil.



Obr.č. 41 – Nastavení exportu GUI do apk

V následujícím dialogovém okně jsem zvolil vizuální projekt aplikace. Zadal jsem patřičné jméno aplikace, které se pak objeví jako název aplikace na smartphonu a nastavil parametry aplikace. Orientaci jsem nastavil na „landscape“ z důvodu poměrů stran displeje a zároveň jsem vybral možnost roztáhnutí na velikost displeje, protože kdybych ji nezatrl, tak by aplikace byla na cílovém zařízení menší než ve skutečnosti.

## 11 SROVNÁNÍ GUI SOFTWARE

Na závěr jsem provedl srovnání nalezených GUI softwarů a individuálního řešení GUI aplikace. Bral jsem ohledy na několik hledisek, které jsem určil ze sestavených požadavků na tvorbu GUI. Nejprve jsem srovnával z pohledu sekce vizuální tvorby GUI aplikace.

Tab.č. 1 – Srovnání GUI softwarů a individuálního řešení z vizuální pohledu

Typ	Vizuální tvorba	Ovládací prvky	Simulátor	Uživatelské proměnné
Individuální návrh	-	+	-	+
Fore UI	+	+	+	+
Crank Storyboard	+	+	+	+
Proto.io	+	+	+	+
GEM studio Pro	+	-	+	-

Co se týče vizuální stránky a ovládacích prvků jsou na tom všechny softwary dobře. Každý obsahuje požadované ovládací prvky a správu obrazovek. Oproti individuálnímu řešení je při tvorbě vizuální stránky GUI uspořena velká část času, protože u svého řešení jsem musel každý ovládací prvek nastylovat (napozicovat) tak, aby seděl s návrhem. Zarovnání ovládacích prvků bylo provedeno ručně pomocí hodnot. U GUI softwarů jsem využil vestavěné nástroje. Jako nejlepší varianta v oblasti GUI designerů bez generování kódu se jeví program Fore UI. Je to způsobeno jednoduchostí a přehledností uživatelského prostředí. Přístup k jednotlivým parametrům ovládacích prvků je rychlý a intuitivní. U Fore UI je dobře vyřešena i možnost přidávání uživatelských proměnných. S tvorbou obrazovek se blíží k webové aplikaci Proto.io, avšak ta je primárně zaměřena na mobilní systémy. Proto.io je webová aplikace a pro uživatele je tedy nejvíce přívětivá, protože není nutná instalace.

V oblasti tvorby embedded GUI je nejlepší volba Crank Storyboard. Z pohledu vizuální tvorby nabízí vrstvený model jako grafické programy a jednoduchou správu jednotlivých obrazovek. Oproti GEM studiu vypadá moderněji a má přehlednější uživatelské prostředí. Uživatelské proměnné je možno vytvořit pro každou obrazovku zvlášť a umožňuje měnit i

parametry jednotlivých ovládacích prvků, jako schovat daný element a zobrazit nový, nebo měnit dynamicky stavové parametry.

Naopak nejhorší software pro vizuální tvorbu jsem vybral GEM studio. Uživatelské prostředí na mě působí zastarale a nenabízí mnoho možností. Navíc má společný panel pro vytváření obrazovek i ovládacích elementů, což je mnohdy nepřehledné. Ale naopak u obrázkového tlačítka obsahuje 2 zdroje obrázku, jeden pro stlačení elementu a druhý pro normální stav. Tedy GUI návrh vypadá dynamicky.

Oproti individuálnímu návrhu je úspora času po tvorbě vizuální stránky velká, protože u mnou vytvořeného návrhu jsem musel brát ohledy na velikosti elementů a ručně zadávat pozice, kde se mají nacházet. V kategorii ovládacích prvků jsou GUI softwary srovnatelné s individuálním řešením.

Tab.č. 2 - Srovnání GUI softwarů a individuálního řešení z pohledu chování GUI

Typ	Definice chování	Akce a funkce	Časová náročnost	Provedení změn
Individuální návrh	-	-	-	-
Fore UI	+	+	+	+
Crank Storyboard	+	+	+	+
Proto.io	+	+	+	+
GEM studio Pro	-	-	-	+

Po stránce úprav jsou všechny softwary dobré a ušetří spoustu času, avšak ne všechny jsou tak dobré a uživatelsky přívětivé.

Z pohledu tvorby chování GUI aplikace jsou nejlepší varianty Fore UI a Crank Storyboard. Fore UI má výborně zpracovaný panel chování a definování jednotlivých událostí a akcí, které se mají provést po spuštění události. Má přehledně odděleny události a akce a také barevně odlišeny i s odsazením. Umožňuje uživateli větvit na základě podmínek a také několikanásobné větvení (switch). Počet provedených akcí na jednu událost nebo podmínku není nijak omezen.

Crank Storyboard má chování aplikace vyřešené trošku jinak, ale také přehledně. Události a akce se vytvářejí v jednom dialogovém okně. Nabízí poměrně hodně efektů přechodů obrazovek i událostí. Pro obrazovky jsou zde k dispozici události po načtení obrazovky, po změně obrazovky atd. Není zde nijak omezen počet akcí na element nebo obrazovky a tedy uživatel může vytvořit, na co si vzpomene. Nevýhodou je, že neobsahuje větvení chování aplikace. Pokud by uživatel chtěl podrobněji větvit GUI aplikaci, musí si prostudovat LUA skripty. Poté při zvolení události vybere akci Lua script a musel by sám definovat nově vzniklou funkci. Lua skripty se poté připojí do složky scripts.

Nejhorší nástroje pro definici chování GUI aplikace má program GEM studio Pro. Jeho jazyk GEM script je chaotický a uživatelsky naprosto nepoužitelný. Jediné co se dalo snadno dělat, byly změny obrazovek. Akce se prováděly pomocí zápisu funkcí GEM scriptu, které se vyvolaly po stlačení klávesy tab. Pro návrh dynamického GUI se nehodí.

Tab.č. 3 - Srovnání GUI softwarů a individuálního řešení z pohledu využití

Typ	Generování kódu / Export	Cena	Instalace	Specifické funkce
Individuální návrh	Ne	-	Ano	+
Fore UI	HTML	+	Ano	-
Crank Storyboard	Ano (.apk, SEE)	-	Ano	+
Proto.io	HTML	+	Ne	-
GEM studio Pro	Ano	-	Ano	-

Z pohledu generování je na tom nejlépe Crank Storyboard. Umožňuje generovat kód pro embedded systém i pro mobilní systémy android a iOS ve formě instalačního balíčku. GEM studio generuje také kód pro embedded systém, ale je podmínkou vlastnit vývojovou desku, kde běží Amulet OS, který vygenerovaný kód dokáže zpracovat a interpretovat. Ostatní GUI designery nabízejí export hotové GUI aplikace do HTML včetně všech souborů bez nutnosti úprav. Ceny jednotlivých softwarů jsem popsal výše. Co se týká instalace, všechny GUI softwary jsou klasické programy, kromě webové aplikace Proto.io, která instalaci má ve formě registrace. Když bych chtěl použít specifické funkce jako výběr

barvy z obrázku, tak žádný GUI designer tuto možnost nenabízel, pouze u Crank Storyboard jsem dal + za možnost animování jednotlivých elementů a zpracování 3D objektů. U individuálního návrhu tato možnost je, ale je velmi časově náročná na inicializaci a implementaci a výsledný generovaný kód pro cílovou platformu není možný, protože tyto funkce jsou tvořeny pomocí JQuery knihoven.

Když zhodnotím softwary po všech stránkách tak nejlepší GUI softwary jsou Crank Storyboard a Fore UI.

## ZÁVĚR

V dnešní době všichni lidé používají embedded systémy denně aniž by tušili, že je používají. V teoretické části práce jsem vysvětlil pojmy GUI a seznámil čtenáře práce, co je embedded systém. Dále jsem lehce nastínil jak embedded systém vznikl a také jak se vyvíjelo grafické uživatelské prostředí od samého počátku do podoby, jakou známe dnes.

Vytvořením individuálního návrhu GUI pomocí technologie DHTML jsem získal mnoho informací a povědomí o tvorbě GUI. Návrh slouží jako předloha logické funkce fyzického ovladače firmy Santech. Z vytvořeného návrhu jsem analyzoval požadavky na GUI generátor z několika hledisek tvorby. Definoval jsem požadavky na jednotlivé elementy, které se mohou vyskytovat v GUI návrhu a také co by měl daný software obsahovat za funkce a nástroje. Definici jsem shrnul do funkčních a nefunkčních požadavků. Požadavky mi pomohly při prozkoumávání a hledání vhodných GUI designerů a GUI generátorů. Testování těchto softwarů jsem prováděl ve virtualizovaném stroji pomocí programu Vmware Player. Pro každý GUI software jsem co nejvěrohodněji předělal vytvořený GUI návrh a následně vyexportoval nebo vygeneroval výstup. Ně vždy se podařilo vytvořit kompletně identický návrh, protože některé specifické funkce nešly realizovat. V praktické části jsem popisoval tvorbu návrhů GUI aplikace v jednotlivých softwarech a demonstroval tak možnosti, které software nabízí. Na konci práce jsem srovnal a zhodnotil všechny GUI softwary z několika základních pohledů na tvorbu GUI návrhů. Tyto srovnání jsou provedena tabulkou a je vždy k nim vybrán nejvhodnější software.

Výstupem diplomové práce jsou 2 GUI mockupy ve formě HTML, které jsou připraveny pro prezentaci skrze webový prohlížeč. Součástí práce je také vygenerovaný instalační balíček pro mobilní systém android, který byl odzkoušen na fyzickém zařízení. Závěrem nejlepší softwary, které jsem zkoumal, byly Crank Storyboard a ForeUI. Pro mnoho uživatelů, kteří nepožadují výstup pro embedded systém, postačí webová aplikace Proto.io.

**SEZNAM POUŽITÉ LITERATURY**

- [1] BLIŽŇÁK, Michal, Tomáš DULÍK a Vladimír VAŠEK. WxShapeFramework: An Easy Way for Diagrams Manipulation in C++ Applications [online]. WSEAS TRANSACTIONS on COMPUTERS: WSEAS Press, 2010, roč. 9, č. 1 [cit. 2014-01-23]. ISSN 1109-2750. Dostupné z: <http://www.worldses.org/journals/computers/computers-2010.htm>
- [2] BLIŽŇÁK, Michal, Tomáš DULÍK a Roman JAŠEK. Production-Ready Source Code Round-Trip Engineering [online]. INTERNATIONAL JOURNAL OF COMPUTERS: NAUN Press, 2012 [cit. 2014-01-23]. ISSN 1998-4308. Dostupné z: <http://www.naun.org/wseas/cms.action?id=3036>
- [3] BLIŽŇÁK, Michal, Tomáš DULÍK, Roman JAŠEK a Pavel VAŘACHA. Optimized Production-Ready Source Code Generation Based on UML [online]. INTERNATIONAL JOURNAL OF SYSTEMS APPLICATIONS, ENGINEERING & DEVELOPMENT: NAUN Press, 2013, roč. 7, č. 1 [cit. 2014-01-23]. ISSN 2074-1308. Dostupné z: <http://www.naun.org/main/UPress/saed/16-498.pdf>
- [4] PRATA, Stephen. Mistrovství v C++. 3. aktualiz. vyd. Překlad Boris Sokol. Brno: Computer Press, 2007, 1119 s. ISBN 978-80-251-1749-1.
- [5] CORMEN, Thomas H. Introduction to algorithms. 3rd ed. Cambridge: MIT Press, c2009, xix, 1292 s. ISBN 978-0-262-03384-8.
- [6] KANISOVÁ, Hana a Miroslav MÜLLER. UML srozumitelně. 2. aktualiz. vyd. Brno: Computer Press, 2006, 176 s. ISBN 80-251-1083-4.
- [7] SMART, Julian. Cross-platform GUI programming with wxWidgets. Upper Saddle River: Prentice-Hall, c2006, xxxv, 700 s. ISBN 01-314-7381-6.
- [8] Seriál historie vývoje GUI. Root.cz [online]. 2010 [cit. 2014-03-20]. Dostupné z: <http://www.root.cz/serialy/historie-vyvoje-grafickeho-uzivatelskeho-rozhrani/#ic=serial-box&icc=title>
- [9] A History of the GUI. Arstechnica.com. [online]. 2005 [cit. 2014-03-20]. Dostupné z: <http://arstechnica.com/features/2005/05/gui/>

- [10] Grafické uživatelské rozhraní. cs.wikipedia.org [online]. [cit. 2014-05-06].  
Dostupný na WWW:  
[http://cs.wikipedia.org/wiki/Grafick%C3%A9\\_u%C5%BEivatelsk%C3%A9\\_rozhran%C3%AD](http://cs.wikipedia.org/wiki/Grafick%C3%A9_u%C5%BEivatelsk%C3%A9_rozhran%C3%AD)
- [11] Embedded System. PREETI, Jain. [online]. 2012 [cit. 2014-05-06]. Dostupné z:  
<http://www.engineersgarage.com/articles/embedded-systems>
- [12] Vestavěný systém. cs.wikipedia.org [online]. [cit. 2014-05-06]. Dostupné z:  
[http://cs.wikipedia.org/wiki/Vestav%C4%9Bn%C3%BD\\_syst%C3%A9m](http://cs.wikipedia.org/wiki/Vestav%C4%9Bn%C3%BD_syst%C3%A9m)
- [13] Embedded GUI design software. Amulettechnologies.com: Embedded GUI solutions made easy [online]. [cit. 2014-04-05]. Dostupné z:  
<http://www.amulettechnologies.com/products/gui-design-software>
- [14] Storyboard Suite. Crank software inc.: Embedded GUI Prototype and Development [online]. [cit. 2014-05-06]. Dostupné z:  
<http://cranksoftware.com/storyboard-suite>
- [15] Fore UI. Foreui.com: Easy-To-Use UI Prototyping Tool [online]. [cit. 2014-03-05]. Dostupné z: <http://www.foreui.com/>
- [16] The History of Embedded Systems. EE times: Connecting the global electronics community [online]. 2008 [cit. 2014-05-06]. Dostupné z:  
[http://www.eetimes.com/video.asp?section\\_id=124&doc\\_id=1317775](http://www.eetimes.com/video.asp?section_id=124&doc_id=1317775)
- [17] Features. Proto.io: Silly-fast mobile prototyping [online]. [cit. 2014-05-06].  
Dostupné z: <http://proto.io/en/features/>
- [18] Pidoco. Pidoco: The Rapid Prototyping Tool [online]. [cit. 2014-04-22]. Dostupné z: <https://pidoco.com/en>
- [19] GUI (graphical user interface). ROUSE, Margaret. SearchWinDevelopment: Silly-fast mobile prototyping [online]. 2006 [cit. 2014-05-01]. Dostupné z:  
<http://searchwindevelopment.techtarget.com/definition/GUI>
- [20] Operating System Interface Design Between 1981-2009. Webdesignerdepot.com [online]. 2009 [cit. 2014-04-11]. Dostupné z:  
<http://www.webdesignerdepot.com/2009/03/operating-system-interface-design-between-1981-2009/>

- [21] Milestones in embedded systems design. Embedded.com [online]. 2008 [cit. 2014-04-18]. Dostupné z: <http://www.embedded.com/design/prototyping-and-development/4007514/Milestones-in-embedded-systems-design>
- [22] Embedded System. Technopedia.com [online]. [cit. 2014-04-06]. Dostupné z: <http://www.techopedia.com/definition/3636/embedded-system>
- [23] The short history of GUI. TheOligarch.Com [online]. 2007 [cit. 2014-05-01]. Dostupné z: [http://www.theoligarch.com/microsoft\\_vs\\_apple\\_history.htm](http://www.theoligarch.com/microsoft_vs_apple_history.htm)
- [24] The real history of GUI. Sitepoint.com [online]. 2001 [cit. 2014-05-01]. Dostupné z: <http://www.sitepoint.com/real-history-gui/>
- [25] Embedded in our society. Visual.ly [online]. [cit. 2014-05-07]. Dostupné z: <http://visual.ly/embedded-our-society-history-embedded-operating-systems>
- [26] Altera Embedded Systems Development Kit. Altera.com [online]. [cit. 2014-05-12]. Dostupné z: <http://www.altera.com/products/devkits/altera/kit-emb-dev-cyc3.html>
- [27] Qt for Embedded Development. Qt.digia.com [online]. [cit. 2014-05-6]. Dostupné z: <http://qt.digia.com/Product/Qt-for-Embedded-Development/#>
- [28] One Giant Leap: The Apollo Guidance Computer. Drdobbs.com [online]. 2000 [cit. 2014-05-12]. Dostupné z: <http://www.drdobbs.com/architecture-and-design/one-giant-leap-the-apollo-guidance-compu/184404139>
- [29] ImageColorPicker. Github.com [online]. [cit. 2014-03-21]. Dostupné z: <https://github.com/Skarabaeus/ImageColorPicker>
- [30] Santech Comfort - grafický návrh GUI rozhraní ovladače. [online]. [cit. 2014-05-02]. Dostupné z: <http://santech-comfort.solansky.cz/>
- [31] Download Fore UI. Foreui.com: Easy-To-Use UI Prototyping Tool [online]. [cit. 2014-04-25]. Dostupné z: <http://www.foreui.com/download.htm>
- [32] GEM studio Pro. Amulettechnologies.com: Embedded GUI solutions made easy [online]. [cit. 2014-04-25]. Dostupné z: <http://www.amulettechnologies.com/index.php/products/gui-design-software>

- [33] Storyboard Suite Evaluation. Cranksoftware.com: Embedded GUI Prototype and Development [online]. [cit. 2014-05-02]. Dostupné z: [http://www.cranksoftware.com/storyboard\\_suite\\_eval](http://www.cranksoftware.com/storyboard_suite_eval)
- [34] Download VMware Player. VMware.com [online]. [cit. 2014-04-22]. Dostupné z: [https://my.vmware.com/web/vmware/free#desktop\\_end\\_user\\_computing/vmware\\_player/6\\_0](https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/6_0)
- [35] Brackets. Brackets [online]. [cit. 2014-04-25]. Dostupné z: <http://brackets.io/>
- [36] Download GIMP. Gimp.com: The GNU Image Manipulation Program [online]. [cit. 2014-04-15]. Dostupné z: <http://www.gimp.org/downloads/>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

GUI	Grafické uživatelské prostředí
PARC	Palo Alto Research Center
Morphing	Plynulá záměna jednoho digitálního obrázku na jiný
HTML	Hypert Text Markup Language
CSS	Cascading Style Sheet
DHTML	Dynamic Hyper Text Markup Language
JS	Javascript
Mockup	Model
Jquery	Javascript Framework
PNG	Portable Network Graphic – bitmapový formát
JPG	Joint Photographics Experts Group - bimapový formát
GIF	Graphics Interchange Format – bitmapový formát
BMP	Windows bitmap – bitmapový formát
UI	Uživatelské prostředí
PDA	Personal Digital Assistant – kapesní počítač
ROM	Read Only Memory – paměť pouze pro čtení
WYSIWYG	What You See Is What You Get – program, který zobrazuje výstup
PDF	Portable Document Format
CAD	Computer Aided Design - Počítačem podpořený návrh
NASA	Národní úřad pro letectví a kosmonautiku
FLASH	Flash Erasable Programmable Read Only Memory
MIT	Massachusettský technologický institut
Widget	Ovládací prvek

**SEZNAM OBRÁZKŮ**

Obr.č. 1 – Ukázka embedded systému [26] .....	12
Obr.č. 2 – Ukázka GUI embedded systému [27] .....	13
Obr.č. 3 – Apollo guidance computer [28] .....	14
Obr.č. 4 – oN-Line systém, klávesnice, myš [23] .....	17
Obr.č. 5 – GUI počítače Xerox Alto [20] .....	18
Obr.č. 6 – GUI SmallTalku [9] .....	19
Obr.č. 7 – GUI počítače Apple Lisa [20] .....	19
Obr.č. 8 – GUI operačního systému Windows 1.0 [9].....	20
Obr.č. 9 – GUI systému Next [23] .....	21
Obr.č. 10 – Vmware Player.....	24
Obr.č. 11 – Uživatelské rozhraní programu GIMP .....	25
Obr.č. 12 – Uživatelské prostředí Brackets Sprint.....	26
Obr.č. 13 – Schéma požadavků na GUI software .....	28
Obr.č. 14 – Hierarchické schéma obrazovek GUI návrhu .....	34
Obr.č. 15 – Diagram aktivity funkce hydro .....	35
Obr.č. 16 – Diagram aktivit změny teploty.....	36
Obr.č. 17 – Diagram aktivit funkce výběru barvy.....	37
Obr.č. 18 – Rozdělení logiky GUI ovladače na 3 zóny.....	38
Obr.č. 19 – ImageColorPicker v akci.....	40
Obr.č. 20 – Nastavení intenzity hydro .....	41
Obr.č. 21 – Komponenty aplikace pro tvorbu embedded GUI rozhraní.....	42
Obr.č. 22 – Uživatelské prostředí Fore UI.....	46
Obr.č. 23 – Aplikace Proto.io .....	47
Obr.č. 24 – GEM studio Pro .....	48
Obr.č. 25 – Uživatelské prostředí Crank Storyboard.....	49
Obr.č. 26 – Import obrázku do projektu .....	51
Obr.č. 27 – Tvorba obrazovek .....	52
Obr.č. 28 – Definice chování elementu při kliku.....	53
Obr.č. 29 – Přejít mezi obrazovkami.....	54
Obr.č. 30 – Vývojový diagram pro funkci hydro .....	55
Obr.č. 31 – Vlastnosti elementu aplikace proto.io.....	57

---

Obr.č. 32 – Úprava hodnoty proměnné.....	57
Obr.č. 33 – Simulátor aplikace Proto.io.....	58
Obr.č. 34 – Tvorba obrazovek GEM studio Pro .....	59
Obr.č. 35 – Panel image.....	60
Obr.č. 36 – Tvorba obrazovek v Crank Storyboard.....	61
Obr.č. 37 – Přejít mezi obrazovkami.....	62
Obr.č. 38 – Příklad akce data change.....	63
Obr.č. 39 – Příklad lua scriptu animace ukazatele intenzity.....	64
Obr.č. 40 – Spuštěný simulátor v akci .....	64
Obr.č. 41 – Nastavení exportu GUI do apk.....	65

**SEZNAM TABULEK**

Tab.č. 1 – Srovnání GUI softwarů a individuálního řešení z vizuální pohledu.....	66
Tab.č. 2 - Srovnání GUI softwarů a individuálního řešení z pohledu chování GUI.....	67
Tab.č. 3 - Srovnání GUI softwarů a individuálního řešení z pohledu využití .....	68

## SEZNAM PŘÍLOH

P I    OBSAH PŘILOŽENÉHO CD

## **PŘÍLOHA P I: OBSAH PŘILOŽENÉHO CD**

**Text** – složka s textem a zadáním diplomové práce

fulltext.doc – Diplomová práce ve formátu Word

fulltext.pdf - Diplomová práce ve formátu PDF

ZadaniDP.pdf – Naskenované zadání diplomové práce

**Vystupy** – složka s vygenerovanými výstupy GUI aplikací

CrankStoryboard – složka s výstupem .apk programu CrankStoryboard

Fore UI – složka s HTML výstupem programu Fore UI

Proto.io – složka s HTML výstupem aplikace Proto.io

Individualni\_navrh – složka s vzorovou GUI aplikací v DHTML

**Zdrojove\_soubory** – složka se zdrojovými soubory GUI aplikací

CrankStoryboard – složka se zdrojovými soubory programu CrankStoryboard

displej.4ui – soubor projektu programu ForeUI

Individualni\_navrh – složka s vzorovou GUI aplikací v DHTML

GEM studio Pro – složka se zdrojovými soubory programu GEM studio Pro