VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

VYUŽITÍ WEBASSEMBLY V PRŮMYSLU 4.0

WEBASSEMBLY USAGE IN INDUSTRY 4.0

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Petr Chvátal**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Ladislav Dobrovský**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Petr Chvátal**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**Ing. Ladislav Dobrovský**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Využití WebAssembly v průmyslu 4.0

Stručná charakteristika problematiky úkolu:

Zvyšující se penetrace webových technologií se potkává s omezením na jazyk JavaScript na straně klienta (webový prohlížeč). Pro integraci systémů psaných v jiných jazycích může být funkčním řešením nastupující standard bytekódu WebAssembly.

Cíle bakalářské práce:

– Rešerše používaných řešení s kompilací klienta do WebAssembly s přehledem podporovaných programovacích jazyků a GUI knihoven.
– Zjištění možností podpory OpenGL přes WebGL.
– Implementace ukázkové aplikace s HMI v kontextu Industry 4.0 s použitím překladu do WebAssembly.
– Zhodnocení výhod a nevýhod využití WebAssembly pro jednodušší a složité aplikace oproti nativním implementacím.

Seznam doporučené literatury:

ZHENG, Gavin, Longxiang GAO, Liqun HUANG a Jian GUAN. WebAssembly(WASM). Ethereum Smart Contract Development in Solidity. Singapore: Springer Singapore, 2020, s. 317-334. ISBN 9811562172. Dostupné z: doi:10.1007/978-981-15-6218-1_11.

ROMANO, Alan a Weihang WANG. WASim: Understanding WebAssembly Applications through Classification. In: 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE) [online]. ACM, 2020, s. 1321-1325 [cit. 2021-10-18].

W3C WebAssembly Working Group [online]. 2021 [cit. 2021-10-21]. Dostupné z: https://www.w3.org/wasm/

Qt for WebAssembly [online]. 2021 [cit. 2021-10-21]. Dostupné z: https://doc-snapshots.qt.io/qt6-dev/wasm.html.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Bakalářská práce se zaměřuje na použití nového webového standartu WebAssembly. V textu práce jsou popsány technologie pro webové prohlížeče a jejich využití. Dále jsou popsány programovací jazyky, které podporují webový standart WebAssembly. Na závěr práce je představená implementace aplikace do tohoto prostředí a překlad aplikace napsané v jazyce C++ do WebAssembly pomocí nástroje Emscripten, zhodnocení výhod a nevýhod použití WebAssembly oproti starším webovým standardům.

ABSTRACT

The bachelor thesis focuses on the use of the new web standard WebAssembly. In the text of the thesis are described the technologies for web browsers and their use. Furthermore, are described the programming languages that support the WebAssembly web standard. Finally, the thesis presents the implementation of an application in this environment and the translation of an application written in C++ into WebAssembly using the Emscripten tool, evaluating the advantages and disadvantages of using WebAssembly over older web standards.

KLÍČOVÁ SLOVA

JavaScript, asm.js, WebAssembly, Emscripten, Qt, C, C++

KEYWORDS

JavaScript, asm.js, WebAssembly, Emscripten, Qt, C, C++

ÚSTAV AUTOMATIZACE A INFORMATIKY

2021

BIBLIOGRAFICKÁ CITACE

CHVÁTAL, Petr. *Využití WebAssembly v průmyslu 4.0* [online]. Brno, 2022 [cit. 2022-05-20]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/139981. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Ladislav Dobrovský.

PODĚKOVÁNÍ

Děkuji svému vedoucímu práce za trpělivost a ochotu opravovat text kdykoli to bylo možné. Dále bych chtěl poděkovat za cenné připomínky a rady, které mi byly poskytnuty během vypracování závěrečné práce.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 20. 5. 2022 ……

Chvátal Petr

OBSAH

1 ÚVOD

Webové prohlížeče jsou nejvíce požívané programy na mobilních telefonech, tabletech, stolních počítačích a noteboocích. Škála možností webových prohlížečů se stále rozrůstá, vznikají nové technologie a standardy, které uvítají jak programátoři, tak i web designeři. Zvětšující se možnosti těchto webových technologií mají za následek omezení používání programovacího jazyka JavaScript na straně klienta. Tyto problémy řešila podmnožina jazyku JavaScript zvaná asm.js. Podmnožina asm.js umožňovala spouštění nativních staticky typovaných jazyků. Asm.js má však své nevýhody, nenabízí stejnou rychlost jako u nativních aplikacích s použitím JIT kompilátoru, má horší zabezpečení na straně klienta a při spouštění složitých projektů se může uživatel potýkat s výraznými problémy spojené s výkonem nebo s kompatibilitou mezi prohlížeči. Kvůli velkým nedostatkům, přišli členové z W3C s novým webovým standardem bytekódu WebAssembly. Od svého představení v roce 2015 vzbudil ve světě front-endu velikou senzaci. WebAssembly je rychlý binární formát, který slibuje téměř stejný výkon jako u nativních aplikacích. Jde o novou cestu pro webové stránky, díky které jsme schopni přenést náročný kód o tisíce řádcích do webové aplikace.

V bakalářské práci jsou rozebrány technologie používané pro webové prohlížeče a implementace ukázkové aplikace převedené na webové stránky pomocí této nové technologie.

2 TECHNOLOGIE POUŽÍVANÉ PRO WEBOVÉ PROHLÍŽEČE

2.1 JavaScript

JavaScript je interpretovaný nebo-li *JIT – compiled* (Just In Time), dynamicky typovaný jazyk s podporou objektově orientovaného programování. Nejvíce je používaný ve webových prohlížečích. JavaScript si ale najde své uplatnění i v jiných prostředích, jako je například Node.js a Adobe Acrobat. Ve webových prohlížečích běží na straně klienta a je převážně používán pro dynamickou stránku webového prohlížeče. [1]

Takové jazyky jako je JavaScript je obvykle těžší dostat do požadované rychlosti spuštění. Nevýhodou dynamických typovaných jazyků je to, že gramátor ztrácí přehled o kódu. Další nevýhodou těchto druhů jazyků je například refaktoring. Jazyk se nekompiluje, takže není možné před zapnutím zjistit, zda byly všem funkcím přiřazeny správné parametry. Vezměme si například jednoduchý výraz A + B. U staticky typovaného jazyka víme, že jsou například hodnoty A, B celá čísla. Uživatel vydá jediný příkaz procesoru pro sečtení dvou čísel. U dynamicky typovaných jazyků nevíme, zda A, B bude číslo, objekt, funkce nebo pole. Takže po sečtení dvou prázdných polí získáme v JavaScriptu prázdný řetězec. Uživatel může přepsat proměnou například s chybou v názvu, pak je možné, že se tato chyba objeví až o několik kroků dál a JavaScript neoznámí neexistenci, pouze vytvoří novou proměnou. [1]

Z důvodu zmíněných bylo vytvořeno nad JavaScriptem více jazyků, které jsou do JavaScriptu překládány a tyto chyby eliminují. Jedním z nich je překladač Emscripten, který je popsán níže v práci. [1]

2.2 Asm.js

Asm.js je podmnožina jazyka JavaScript. Je to technologie, která umožňuje spouštění nativního kódu na webových prohlížečích s velmi vysokým výkonem. Asm.js lze použít ke spouštění herních enginů a složitých výpočtů nebo logických aplikacích. Asm.js není jiný programovací jazyk ani knihovna. Jde o způsob psaní programu v JavaScriptu, který je pro JavaScript engine a překladač jednoduší na zpracování. [1]

Celý tento překlad se děje pomocí nástroje Emscriptem, který má za úkol vzít existující zdrojový kód a přeloží ho do podmnožiny jazyka JavaScript. Asm.js dokáže spustit v prohlížeči nativní staticky typované jazyky. Asm.js není určen k psaní přímo. Obvykle uživatel vezme už existující zdrojový kód napsaný v jazyce C, C++, Rust nebo dalších nízko úrovňových jazycích a poté provede překlad do jazyka asm.js. [1]

Při kompilaci se přidávají nedůležité příkazy. asm.js přidává další výkonnostní optimalizaci a tou je absence garbage collection. Programy přeložené do asm.js nemají

produkovat žádné nepotřebné operace, což znamená že by si měly samy alokovat a delegovat paměť. [1]

2.3 asm.js a WebAssembly

WebAssembly je relativně nový typ kódu, který může běžet na webových prohlížečích a je popsán níže v práci. Asm.js však zůstává užitečným záložním zdrojem pro WebAssembly prostřednictvím programu napsaného organizací WebAssembly. Asm.js je po boku WebAssembly zastaralý. WebAssembly má formát bajtového kódu, který se rychleji analyzuje. WebAssembly je vedle HTML, JavaScriptu a CSS čtvrtým jazykem běžícím uvnitř prohlížeče. Je to binární jazyk, který má stejný účel jako asm.js, ale je implementován jiným způsobem. Programátor už neposílá ořezanou verzi JavaScriptu. Místo toho posíláte binární kód, který je spuštěn v sandboxu JavaScriptu. [1]

2.4 WebAssembly

WebAssembly je momentálně nejnovější webový standard. Od svého představení, které bylo v roce 2015 vzbudil ve světě front-endu obrovskou senzaci. Veliké společnosti zabývající se technologickými pokroky jako eBAY, Google a Apple tento webový standard používají v projektech cílících na uživatele, kterým by rychlost JavaScriptu nestačila. Zejména v dnešní době, kdy většina uživatelů používá internet na mobilních zařízeních, které nemají tak silné procesory a mají pomalé datové linky. [2]

2.4.1 Představení WebAssembly

WebAssembly je kompaktní, rychlý binární formát, který slibuje téměř stejný výkon jako u nativních aplikací. V současné době je nejběžnější využití WebAssembly ve webových prohlížečích. WebAssembly má být více než řešení pro webové prohlížeče, v budoucnu by měly přijít další podporované funkce a WebAssembly by se měl stát užitečný například v mobilních a desktopových aplikacích. [2]

Je to nová cesta pro web, jak vytvořit náročný kód, který jde spustit uvnitř Vaší webové aplikace. WebAssembly není náhrada za JavaScript, ale naopak spolupracují. WebAssembly Dokáže vzít jazyky jako je třeba C++, Rust a zkompiluje je do něčeho, čemu říkáme WebAssembly modul nebo weather module. Po kompilaci ho můžeme načíst do naší webové aplikace a vyvolat pomocí JavaSriptu. Když vytváříme takovou věc, hlavní prioritou je rychlost. Dokáže běžet extrémně rychle, a to až tak, že běží o nepoznání pomaleji jak nativní kód. [2]

Jak již bylo zmíněno je to binární formát, který je specificky navržen tak, aby byl bezpečnější než JavaScript. Instrukce jsou velmi omezené, tajné a důvěrné. Je zde navrhnutá celá ochranná vrstva a dokáže běžet uvnitř Sandboxu. To zaručuje velkou

skupinu vrstev ochrany pro uživatele. Jeho největší prioritou je, že dokáže být spuštěn na všech moderních, nejpoužívanějších webových prohlížečích. [2]

2.4.2 Hlavní cíle WebAssembly

Řekněme, že máme spoustu nativních kódů, jako například nativní aplikace běžící na ploše počítače nebo v AppStoru. Které mohou být napsané například v jazyce C++ a obsahovat firemní strategii nebo třeba herní engine. Aplikace, které mohou být napsané na tisících řádcích, ale jsou pouze uloženy ve vašem počítači. Nyní je však díky WebAssembly můžeme přenést na internet. [2]

V dnešní době se mnohem více aplikací přesouvá na internet. WebAssembly umožní získat nativní rychlost bez toho, aby byla aplikace překompilována na jinou platformu. [2]

Největší společnosti, kteří nyní využívají WebAssembly jsou Unity Technologies a Epic games. Mezi nejvýraznější projekty patří Unity Engine (Unity Technologies) a Unreal Engine (Epic games). [2]

2.4.3 Bezpečnost WebAssembly

Každý jednotlivý modul WebAssembly se spouští v prostředí Sandboxu odděleném od časového období, během kterého je program spuštěn (runtime). To má za následek to, že se aplikace spouštějí nezávisle a nemohou opustit prostředí Sandboxu, aniž by prošly příslušnými rozhraními API. Aplikace jsou obecně vykonávány deterministicky s omezenými výjimkami. Každý modul navíc podléhá bezpečnostním zásadám jeho vložení. V rámci webového prohlížeče to zahrnuje omezení toku informací prostřednictvím zásad stejného původu. [3]

Samotný návrh WebAssembly podporuje bezpečné programy tím, že se eliminují nebezpečné funkce ze sémantiky provedení, ale zároveň zachovává kompatibilitu s programy napsanými například pro jazyky C/C++. [3]

2.5 Anatomie WebAssembly

Stejně jako všechny ostatní programovací jazyky tak je i WebAssembly postaven na několika základních koncepcích. [4]

2.5.1 VALUES

V současné době jsou uvedeny čtyři typy hodnot včetně celých čísel a sofistikovanějších čísel s desetinou čárkou. Tato čísla většinou představují booleovské hodnoty a místa v paměti. [4]

2.5.2 MODULE

Je to část spustitelného strojového kódu zkompilovaného prohlížečem z binárního souboru WebAssembly. Má příponu *.wasm* a může být importován a exportován hostitelským prostředím, do kterého byl vložen. [4]

2.5.3 TABLE

Je to pole obsahující hodnoty jednotlivých položek. Obsahuje ukazatele na místa v paměti funkcí. [4]

2.5.4 FUNCTIONS

Organizované bloky kódu, které přijímají a vracejí sady hodnot jako parametry a výstupy. [4]

2.5.5 SHRNUTÍ

Všechny tyto pojmy tvoří strukturu kódu WebAssembly. Tento kód může být v textovém tak i v binárním formátu. Při kompilaci prochází kód třemi hlavními procesy: 1) dekódování modulu 2) Kompilace, optimalizace a ověření kódů podle předem definovaných kritérií 3) spuštění zkompilovaného kódu. [4]

2.6 Princip standardu WebAssembly

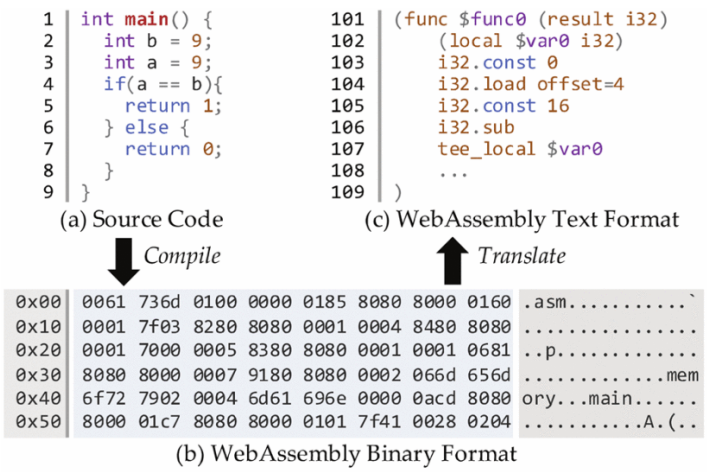
WebAssembly je jako kompilátor. Uživatelé nepíší WebAssembly přímo, ale v libovolném podporovaném jazyku dle svého uvážení, který je pak pomocí kompilátoru převeden do bajtového kódu na internetové stránky. Bajtový kód je poté spuštěn ve webovém prohlížeči, kde je tento kód přeložen do strojového kódu velmi vysokou rychlostí blížící se nativní. Kód WebAssembly by měl být načítán, analyzován a spouštěn rychleji než v JavaScriptu. Při používání WebAssembly ve webovém prohlížeči stále musíme stahovat a nastavovat modul WASM, to by však za ponechání ostatních podmínek nemělo mít vliv na rychlost a WebAssembly by měl běžet rychleji. [5]

Základním použitím pro WebAssembly je psaní softwaru ve webovém prohlížeči. Části zkompilované do WebAssembly mohou být napsány v některém z podporovaných jazyků (C++, C, Python a další) konečný výsledek WebAssembly je pak uživatelům doručen přes JavaScript. WebAssembly je navržen tak, aby vyhověl v řadě náročných úkonů jako jsou například hry, streamování hudby, střih videa anebo například šifrování. [5]

2.6.1 Ukázka kódu WebAssembly

Na obrázku můžeme vidět ukázkovou funkci, na které si můžeme popsat princip a základy fungování WebAssembly. Obrázek 1. ukazuje WebAssmebly modul ve formátu bytekódu a textu. Na obrázku 1a) je vidět část kódu napsaná v jazyce C++. Program vytvoří dvě proměnné (b, a) a přiřadí jim požadovanou hodnotu. Funkce má za úkol porovnat dvě proměnné a zhodnotit výsledek. Kód C++ je zkompilován do binárního kódu WebAssembly viz obrázek 1b). Binární formát je cílový formát, který je dodáván prohlížečům a zároveň je jimi kompilován. Pro snazší ladění můžeme binární soubor WebAssembly přeložit do textového formátu a tím se nám mohou objevit příklady příkazů WebAssembly viz obrázek 1c). [5]

Textový formát WebAssembly je lépe čitelný, ale je stále potřeba se tento formát naučit. Jazyk definuje pouze čtyři číselné typy: i32, i64, f32 a f64. Textový formát WebAsssembly je podobný assembleru a ve srovnání s vysokoúrovňovými jazyky je obtížnější mu porozumět a pochopit ho. Za tímto účelem byl vyvinut klasifikační nástroj WASim, který pomáhá uživatelům porozumět aplikaci WebAssembly. [5]

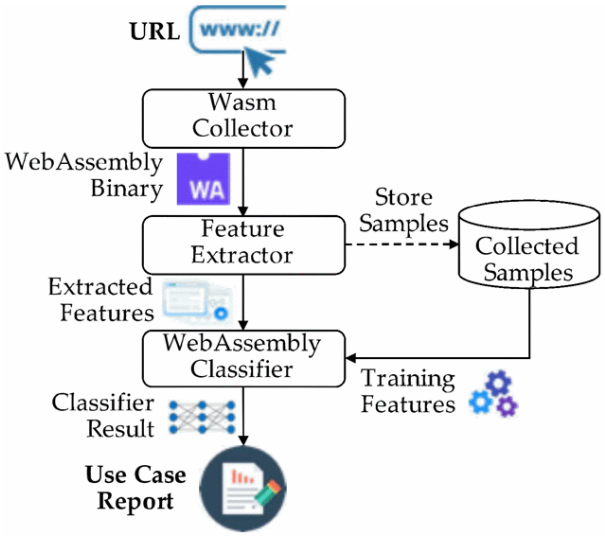


Obr. 1 Kód WebAssembly [5]

2.6.2 WASim

WASim má za úkol identifikovat záměry programů WebAssembly pomocí analýzy kódu. WASim se skládá ze tří hlavních částí: 1) WASim Collector – Po zadání adresy URL WASim collector zjišťuje, zda se na stránce vyskytuje WebAssembly a pokud ano, stáhne jeho binární soubory, které jsou spuštěné na stránce. 2) Feature Extractor – Má za úkol přenést binární soubory na textové soubory WebAssembly a poté textové soubory prohledá a extrahuje požadované funkce. 3) WebAssembly Classifier – Převezme extrahované soubory a použije je ke klasifikaci binárního souboru WebAssembly do

jedné z 11 předdefinovaných tříd. Konečný výstup programu WASim je zpráva, která obsahuje seznam identifikovaných funkcí programu a hlásí jejich účel. [5]



Obr. 2 WASim [5]

3 REŠERŠE PODPOROVANÝCH JAZYKŮ PRO WEBASSEMBLY

Mezi nejvíce používané programovací jazyky pro kompilaci do WebAssembly patří C, C++, C#, Rust. Avšak v dnešní době WebAssembly podporuje většina programovacích jazyků, které vidíme v následující tabulce. [7]

Následující tabulka zobrazuje podporu kompilací jazyků do WebAssembly. Najdeme mnohem více jazyků, než je zobrazeno v tabulce, ale zde je vypsaných hlavních devatenáct nejběžnějších používaných programovacích jazyků pro WebAssembly nebo jiné často používané programovací jazyky. [6], [7]

V tabulce je vyobrazena podpora kompilace pro spuštění v prohlížeči, kompilace pro spuštění mimo prohlížeč a v prostředí WASI. [7]

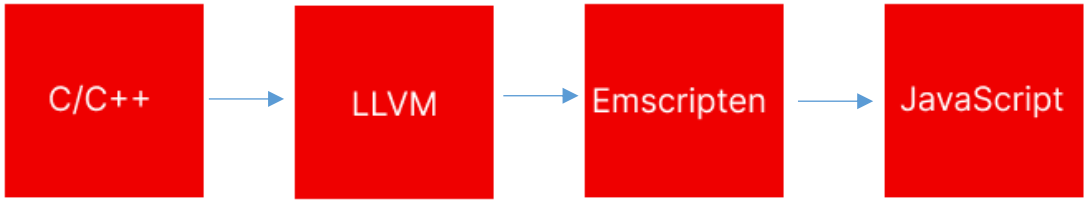
Programovací Jazyk	Prohlížeč	Ostatní	WASI
JavaScript	V přípravě	V přípravě	V přípravě
Python	V přípravě	Použitelný	Použitelný
Java	Použitelný	Použitelný	Není implementováno
PHP	Použitelný	Není implementováno	Není implementováno
C++	Použitelný	Použitelný	Použitelný
C# and .NET	Použitelný	Použitelný	Použitelný
TypeScript	Není implementováno	V přípravě	Není implementováno
Ruby	V přípravě	Použitelný	Použitelný
C	Použitelný	Použitelný	Použitelný
Swift	Použitelný	Použitelný	Použitelný
R	Použitelný	Není implementováno	Není implementováno
Objective-C	Není implementováno	Není implementováno	Není implementováno
Shell	Nelze použít	Nelze použít	Nelze použít
Scala (native)	Není implementováno	V přípravě	Není implementováno
Go	Použitelný	Použitelný	Použitelný
PowerShell	Není implementováno	Není implementováno	Není implementováno
Kotlin	Použitelný	V přípravě	V přípravě
Rust	Použitelný	Použitelný	Použitelný
Dart	V přípravě	Není implementováno	Není implementováno

Tab. 1 Přehled podporovaných programovacích jazyků do WebAssembly [7]

3.1 Emscripten

Emscripten je kompletní Open-Source LLVM kompilátor pro webové prohlížeče. Hlavním cílem Emscripten je to, že dokáže přeložit programy napsané v jazycích C, C++ a dalších, které podporují LLVM bytového kódu, který je následně přeložen do

JavaScriptu. I přes omezení, kterými Emscripten disponuje, se povedlo přeložit velké množství velkých projektů jako je například Unreal Engine 4 a Unity. Celý proces je znázorněn na obrázku 3. Zdrojový kód je přeložen do LLVM bytekódu pomocí Clnagu, který je po přeložení zpracován do finální podoby. [8]



Obr. 3 Proces kompilace Emscripten [vlastní zpracování]

Pomocí nástroje Emscripten lze do WebAssembly zkompilovat prakticky každý kód v jazyce C a C++ od výkonných her, které kladou velký požadavek na vykreslování grafiky až po multimediální přehrávače. Emscripten generuje malý a rychlý kód a jeho výstupním formátem je právě WebAssembly. [8]

3.2 Jazyky C, C++ a jejich kompilace

V následujících řádcích si představíme jazyky C, C++ a ukážeme kompilaci jejich kompilaci pomocí nástroje Emscripten, protože patří mezi nejběžnější používané programovací jazyky pro tento nástroj. [9]

3.2.1 Jazyk C

Jazyk C je imperativní strukturovaný programovací jazyk. Používá se při programování operačního systému a aplikačního softwaru jak u superpočítačů tak i u PLC. [9]

3.2.2 Jazyk C++

Jazyk C++ je multiparadigmatický programovací jazyk. C++ je rozšíření jazyka C a podporuje objektově orientované programování. [9]

3.3 Kompilace pomocí nástroje Emscripten

1) Pro ukázku si vezmeme jednoduchý příklad „Hello Word“. Tento kód si uložíme s názvem hello.c v adresáři na našem místním disku. [9]

2)

```c
#include <stdio.h>

int main() {
    printf("Hello World\n");
}
```

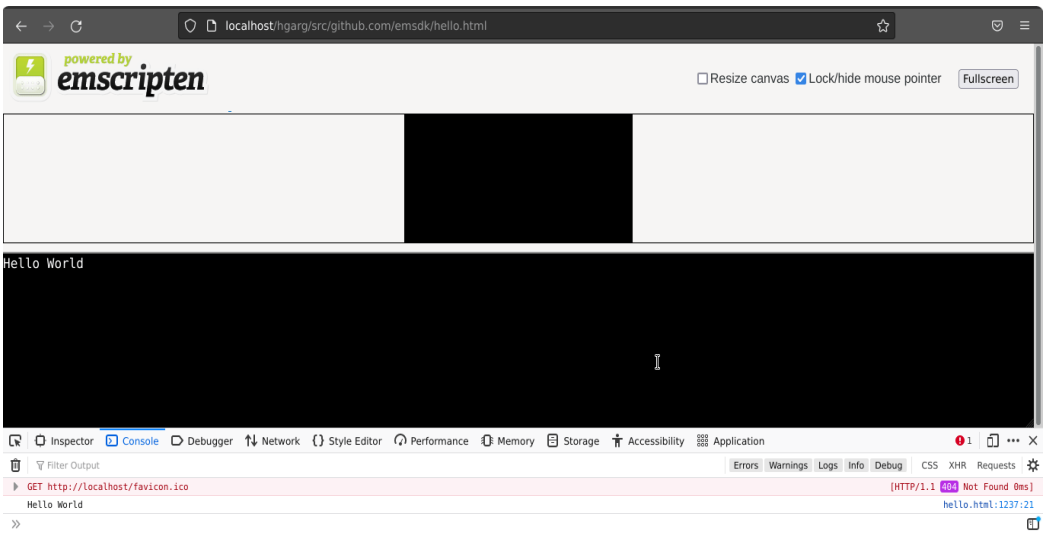
Obr. 4 Script příkladu Hello Word [9]

3) V terminálovém okně, které sloužilo jako vstup do kompilátoru Emscripten vyhledáme adresář, ve kterém jsme v kroku 1. uložili soubor hello.c. Poté spustíme následující příkaz. [9]

```
emcc hello.c -o hello.html
```

Obr. 5 Spuštění příkazu [9]

4) *-hello.html* – tento příkaz určuje, že chceme, aby Emscripten vytvořil stránku HTML, ve které bude náš kód spuštěn. Dále vygeneruje modul WASM a pomocný kód JavaScript, který WASM zkompiluje a upraví tak, aby mohl být spuštěn ve webovém prostředí. Ve zdrojovém adresáři se nám zobrazí binární kód WASM, soubor v JavaScriptu, který nám přeloží nativní funkce jazyka C a konečný soubor HTML, který zkompiluje kód WASM. Vše se nám zobrazí v prohlížeči. Načteme výsledný soubor v prohlížeči, který WebAssembly podporuje. Výstup *´´Hello Word´´* se objeví v konzoli Emscripten ve webovém prohlížeči. [9]



Obr. 6 Konzole Emscripten ve webovém prohlížeči [9]

4 OPENGL A WEBGL

4.1 OpenGL

OpenGL je softwarové rozhraní ke grafickému hardwaru. Rozhraní se skládá z přibližně 150 různých příkazů, které se používají k operacím potřebných k vytvoření interaktivního obrazce v trojrozměrném prostředí. OpenGL je zjednodušené rozhraní, které není závislé na hardwaru, díky tomu lze implementovat na mnoha systémech a různých hardwarových platformách. K dosažení těchto vlastností nejsou použity žádné příkazy pro zobrazení oken, nebo získání vstupů zadaných uživatelem. Uživatel pracuje s příkazy, které se v OpenGL systému ovládají konkrétním použitým hardwarem, který má uživatel k dispozici. OpenGl nepodporuje vysokoúrovňové příkazy pro popis trojrozměrných modelů. Takové příkazy by umožnili zadat složitější, komplexnější tvary, jako jsou například automobil, molekuly, části těla. Díky tomu může být rozhraní založeno na architektuře klient-server. Program díky tomu může fyzicky běžet na jakémkoliv počítači, než na kterém se příkazy vykonávají. Všechny příkazy se předávají pomocí počítačové sítě. [10]

4.2 WebGL

WebGL neboli Web Graphics Library je JavaScript API pro vykreslení vysoce výkonné interaktivní 3D a 2D grafiky kompatibilním prohlížeči, bez použití zásuvných modulů. WebGL a zavedení API odpovídá úzce OpenGL ES 2.0. Tato shoda umožňuje využívaní výhod hardwarové konfigurace ze zařízení uživatele. V současné době je WebGL k dispozici na všech předních prohlížečích, jakou jsou Firefox, Google Chrome, Opera, Safari, Microsoft Edge. WebGL je chápáno jako rozhraní API pro 3D vykreslování, které se mělo používat ve spolupráci s prvkem HTML 5. Jeho použití eliminuje některá slabá místa vykreslování rozhraní API 2D a poskytovalo téměř přímý přístup ke grafickému procesoru počítače. [11;12]

4.2.1 Použití

Použití WebGL je mnohem složitější díky složitosti trojrozměrného vykreslování. Z tohoto důvodu je nad WebGL postaveno několik knihoven. V současné době WebAssembly používá implementaci knihovny SDL (Simple Direct Media Layer), kterou používá většina vývojářů pro psaní her. Tato verze SDL pro WebAssembly je postavena nad WebGL a poskytuje vyšší výkon a je mnohem jednoduší na používání.
SDL knihovna nám umožňuje používat WebGL přímo z kódu programu C++, který již byl zkompilován do WebAssembly. [11;12]

4.3 Podpora OpenGL přes WebGL

Podpora OpenGL přes WebGL je aplikována pomocí nástroje Emscripten, který jsme zmínili výše v práci. Emscripten dokáže zkompilovat kód v jazyce C/C++, který používá OpenGL 2.0 nebo OpenGL ES 3.0 přes WebGL nebo WebGL 2. Z těchto důvodů Emscripten podporuje pouze malé množství příkazů OpenGL ES, které odpovídají příkazům dostupným uvnitř používané knihovny WebGL. Pokud chceme například používat verzi OpenGL ES 3.0 musíme při kompilaci zahrnout WebGL 2 předáním parametru `-s USE_WEBGL2=1` kompilátoru Emscripten. SDL knihovna pro WebAssembly v „zákulisí“ používá implementaci OpenGL od Emscripten, která je vázána a spoléhá na WebGL. Lepší pochopení OpenGL ve WebAssembly pomůže uživateli pochopit a zlepšit své dovednosti například při vývoji her. [11]

5 PRŮMYSL 4.0 A HMI APLIKACE

5.1 Historie Průmyslu 4.0

Pojem průmysl 4.0 se zrodil především v Německu v roce 2011. Průmysl 4.0 znamená čtvrtou průmyslovou revoluci. Za první průmyslovou revoluci je považován parní stroj, tento vynález umožnil využití parní síly a otevřel tak éru průmyslu. Za druhou průmyslovou revoluci je považováno zavedení elektřiny, zavedení elektřiny mělo za následek vytvoření masové výroby zejména v novém automobilovém průmyslu. Třetí průmyslová revoluce je převážně spojovaná s automatizací výroby a využitím elektroniky a informačních technologií. [13; 14]

Na rozdíl od čtvrté průmyslové revoluce jsou předešlé tři spojené s vynálezy založenými na průlomových vědeckých objevech a našli své uplatnění v novém průmyslovém prostředí. Průmysl 4.0 není považován za technickou revoluci spojenou s vědeckým objevem. [13; 14]

5.2 Současnost Průmyslu 4.0

V současné době se výroba stává dynamická a rychlá. Kvalita výrobků a rychlost hraje důležitou roli v udržitelnosti firmy na trhu. Průmysl 4.0 představuje nové technologie, transformuje průmyslovou strukturu a umožňuje firmám zlepšovat výrobní systém prostřednictvím přístupu k informacím z výrobních sítí v reálném čase. Koncepce Průmyslu 4.0 nabízí zjednodušený provozní systém a výrazně minimalizuje náklady a dobu realizace. [13; 14]

Průmysl 4.0 je ale zároveň velkou výzvou pro firmy. Jedná se o relativně nově vyvinutý koncept. Firmy se potýkají s neznalostí problematiky, nedostatkem financí a provozními komplikacemi. Přijetí této technologie může být pomalé, pokud ji firmy budou považovat za složitou a obtížně implementovanou. Zavádění této technologie vyžaduje značné úsilí a znalosti. [13; 14]

5.3 HMI – Human Machine Interface

HMI z anglického *Human Machine Interface* je rozhraní člověk-stroj. Je to systém pro komunikaci mezi obsluhou a průmyslovým systémem jako je například výrobní linka. HMI zobrazuje stavy, hodnoty, trendy a umožňuje přímou interakci mezi člověkem a strojem. Vytvoření aplikace HMI není tak jednoduché, jak se na první pohled zdá. Při vytváření aplikace HMI se vývojář musí zamyslet jak a kdo bude schopen naši aplikaci používat. Proto je důležité zobrazovat pouze relevantní informace. V oblasti HMI je nejdůležitější vytvoření a celkové nastavení aplikace od vizuální po praktickou stránku aplikace, aby byla pro člověka srozumitelná. Vytváření dobré HMI aplikace hraje důležitou roli i psychologická stránka. Člověk by si měl zapamatovat méně informací než

počítač, protože množství a rychlost lidských úkonů se s rostoucím počtem informací výrazně snižuje. HMI pouze vizualizuje a umožňuje člověku ovládat technologickou vrstvu. Primárním úkolem HMI je zobrazení reálného světa prostřednictvím digitální formy uživateli. Pro aplikaci HMI jsou vhodné webové technologie. [15]

5.3.1 Webové technologie s HMI

Webové technologie jsou neoddělitelnou součástí životů. Webové technologie si našli svoji cestu do průmyslového odvětví především díky zvýšení rychlosti a spolehlivosti. Webové technologie se především používají k vytváření HMI systémů. [15]

5.3.2 Výhody používání webových technologií

Mezi hlavní výhodu patří přizpůsobivost. Vývojáři mohou vytvořit systém HMI jako webovou aplikaci nebo stránku. Webové stránky lze prohlížet na zařízeních s různými operačními systémy a s rozlišením obrazovky. [15]

Operátoři mohou být spojeni s výrobní linkou po celou dobu své služby i mimo ní a to třeba díky chytrému zařízení. Operátoři dokážou řídit výrobní proces přímo z výrobní linky nebo naopak ze vzdáleného místa. [15]

5.3.3 Hlavní požadavky

Webové technologie a způsob jejich implementace do průmyslové aplikace HMI patří ergonomie a kontextová uvědomělost. [15]

5.3.4 Ergonomie a psychologie v HMI

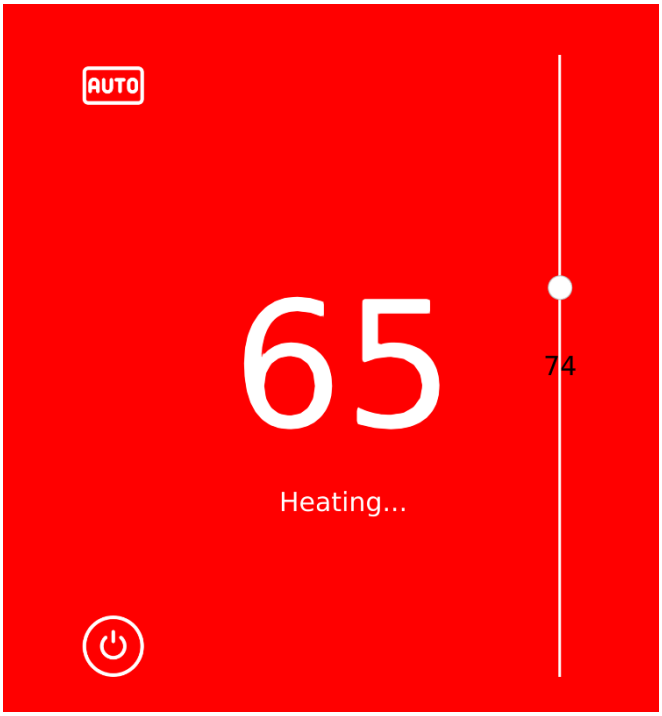
Systémy HMI s dobrou ergonomií minimalizují ztráty spojené se selháním obsluhy. Uživatel potřebuje od HMI aplikace vidět pouze důležité informace. Příliš mnoho informací nebo nedůležité informace odvádí pozornost a prodlužuje tím reakční dobu obsluhy. Mezi nejdůležitější požadavky patří: 1) obsluha musí vidět pouze pro něj důležité informace. 2) informace musí být logicky seřazeny. 3) Barevné označení a symboly musí mít vždy stejný význam. [15]

6 IMPLEMENTACE UKÁZKOVÉ APLIKACE HMI V KONTEXTU PRŮMYSLU 4.0

6.1 Úvod do aplikace HMI

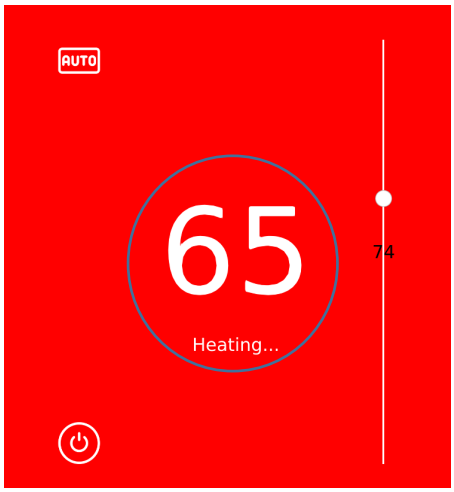
Pro vytvoření ukázkové aplikace byl vybrán jednoduchý chytrý teploměr. Byl zrealizován pomocí QML a backhand C++ v prostředí Qt creator 7.0.1. Grafický návrh byl vytvořen pomocí internetové designové aplikace FIGMA. Finální aplikace je převedena pomocí kompilátoru Emscripten do WebAssembly a poté spuštěna na webových stránkách.

6.2 Funkčnost



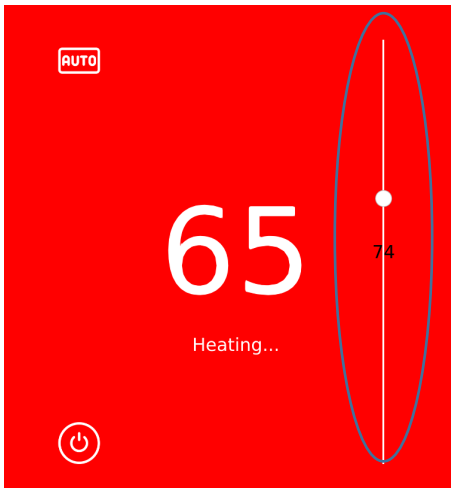
Obr. 7 Aplikace HMI [vlastní zpracování]

Desing aplikace HMI byl vytvořen v prostředí FIGMA. V hlavním panelu jsme vytvořili číselný ukazatel, který nám zobrazuje aktuální stav teploty.



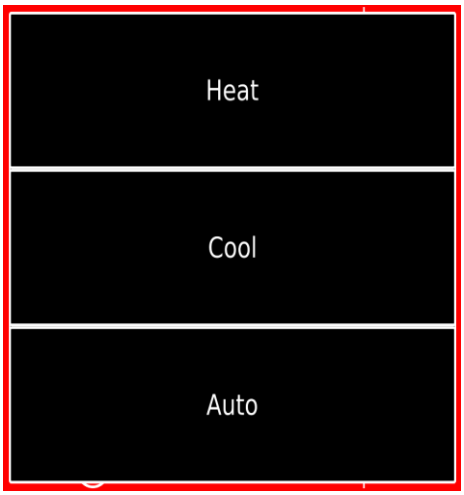
Obr. 8 Hlavní panel aplikace [vlastní zpracování]

Na pravé straně je posuvný slider, který nám umožňuje měnit hodnotu teploty.



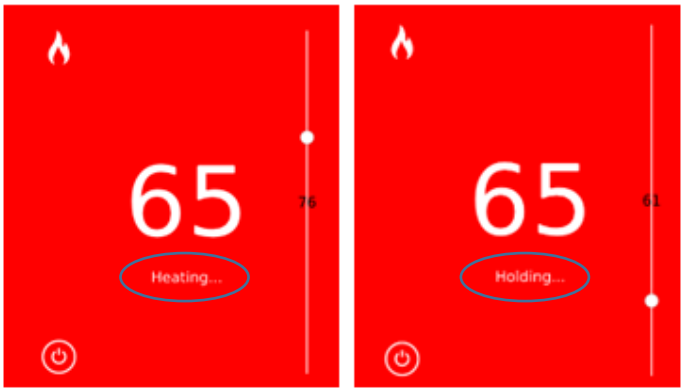
Obr. 9 Posuvný slider [vlastní zpracování]

V pravém horním rohu máme aktuální symbol. Po rozkliknutí tohoto symbolu můžeme vidět hlavní nabídku, která se skládá ze třech možností Heat, Cool a Auto.



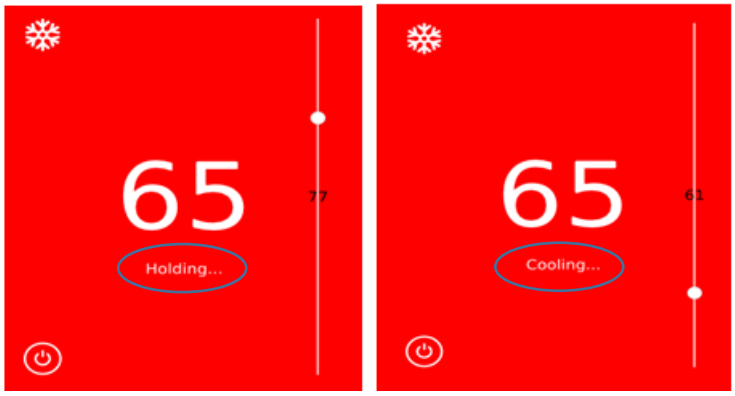
Obr. 10 Hlavní nabídka [vlastní zpracování]

V režimu Heat se nám drží ukazatel pod hodnotou 65, po překročení této hodnoty teploměr ukazuje signalizuje topení (Heating).



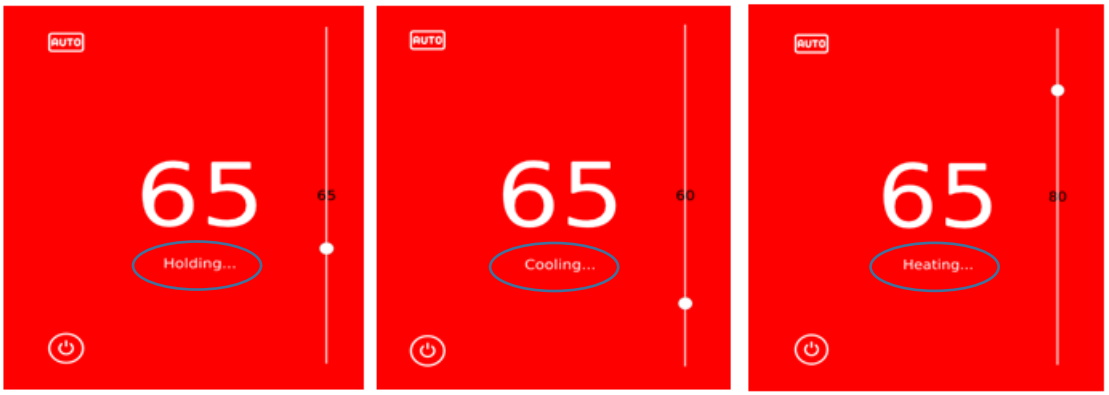
Obr. 11 Režim Heat [vlastní zpracování]

V režimu Cool se hodnoty nad hranicí 65 drží provozních hodnotách (holding), po překročení hranice 65 a nižší teploměr signalizuje chlazení (Cooling).



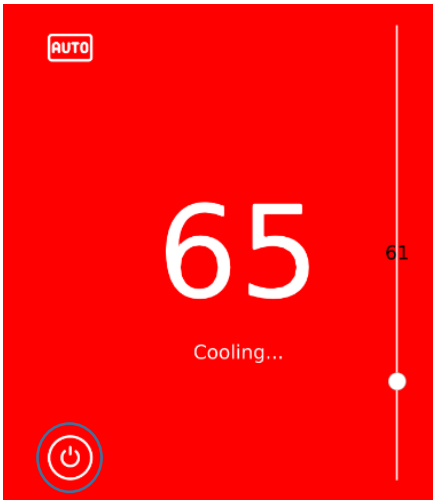
Obr. 12 Režim Cool [vlastní zpracování]

V režimu AUTO drží teploměr teplotu (holding) pouze na zvolené žádané hodnotě 65. Po překročení této hodnoty směrem dolů, aplikace signalizuje chlazení (cooling) naopak po překročení hodnoty 65 směrem nahorů, aplikace ukazuje topení (heating).



Obr. 13 Režim Auto [vlastní zpracování]

V levém spodním rohu můžeme vidět vypínací tlačítko (on/off), které nám aplikaci teploměru ukončí.



Obr. 14 Tlačítko on/off [vlastní zpracování]

6.3 Využití Qt pro WebAssembly

Qt pro WebAssembly znázorňuje zásuvný model platformy, který umožňuje vytvářet aplikace v prostředí Qt, jež lze implementovat na webové stránky. [16]

6.4 Instalace Emscripten

Jako první musí uživatel nainstalovat Python 3.6 nebo novější. Starší verze by nemuseli fungovat kvůli změnám na GitHubu s protokolem SSL. Po nainstalováni Pythonu musíme získat základní ovladač Emcsripten SDK (emsdk) což je script v programovacím jazyce Python. [17]

```
# Get the emsdk repo
git clone https://github.com/emscripten-core/emsdk.git

# Enter that directory
cd emsdk
```

Obr. 15 Základní ovladač Emscripten pro Python [17]

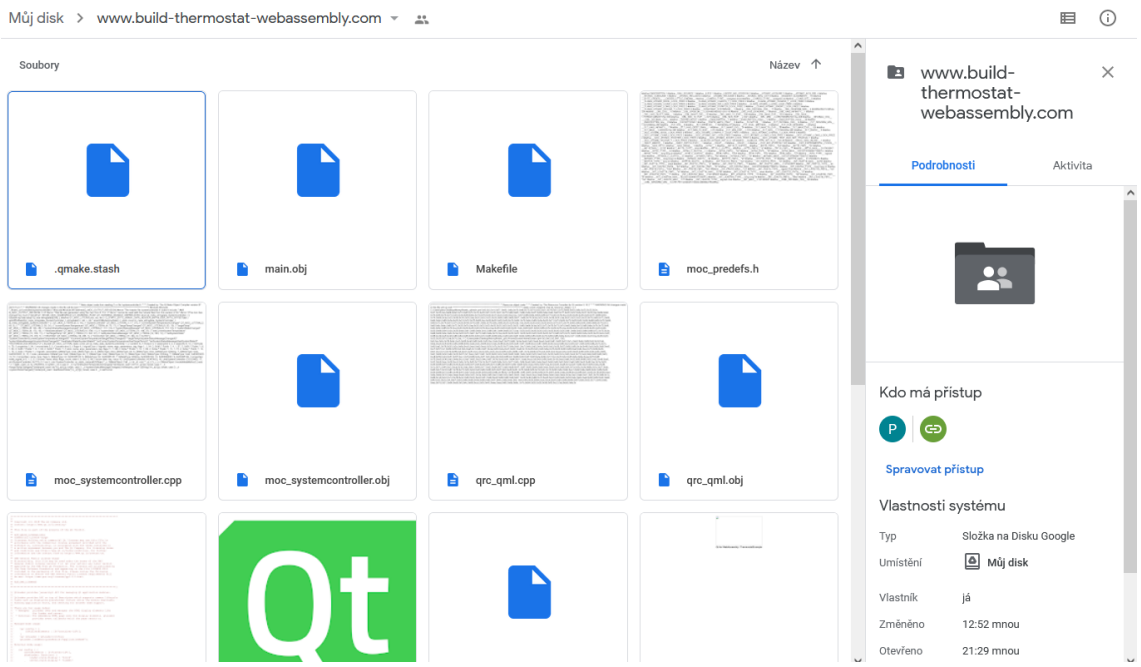
Poté spustíme následující příkazy emsdk, abychom nainstalovali potřebné nástroje ze služby GitHub a nastavili je jako aktivní. [17]

```
# Fetch the latest version of the emsdk (not needed the first time you clone)
git pull

# Download and install the latest SDK tools.
./emsdk install latest

# Make the "latest" SDK "active" for the current user. (writes .emscripten file)
./emsdk activate latest

# Activate PATH and other environment variables in the current terminal
source ./emsdk_env.sh
```

Obr. 16 Instalace nástrojů ze služby GitHub [17]

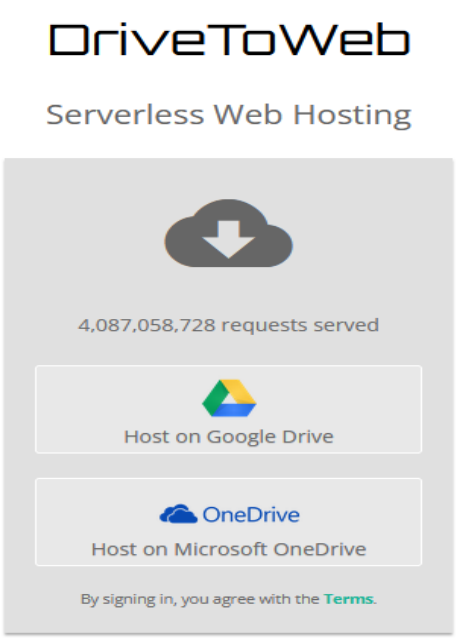
6.5 Pokyny pro zobrazení na webových stránkách

Po nainstalování a aktivaci potřebných nástrojů a funkcí můžeme náš projekt nahrát na hosting pomocí Google drive.
Na Google disku vytvoříme potřebnou složku, kam nahrajeme svůj projekt.



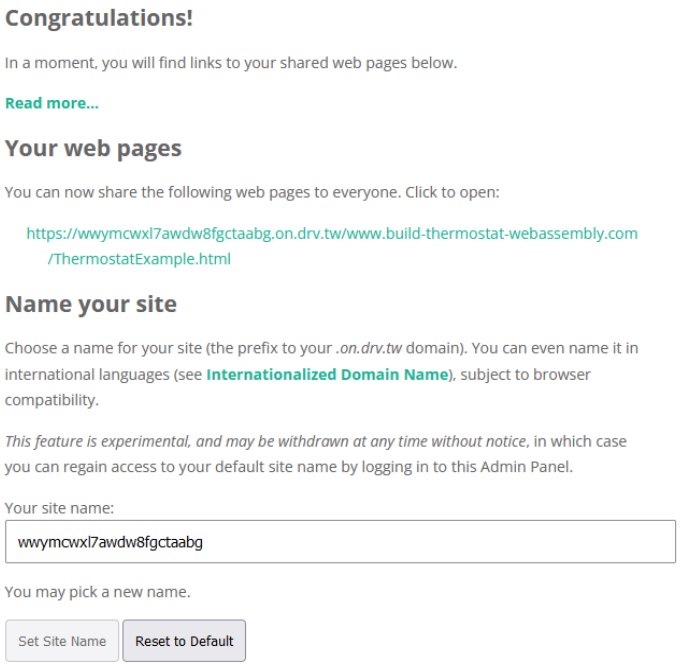
Obr. 17 Google drive [vlastní zpracování]

Po nahrání požadovaných souborů do složky a povolení přístupu veřejnosti otevřeme webovou stránku: https://drv.tw



Obr. 18 DriveToWeb [vlastní zpracování]

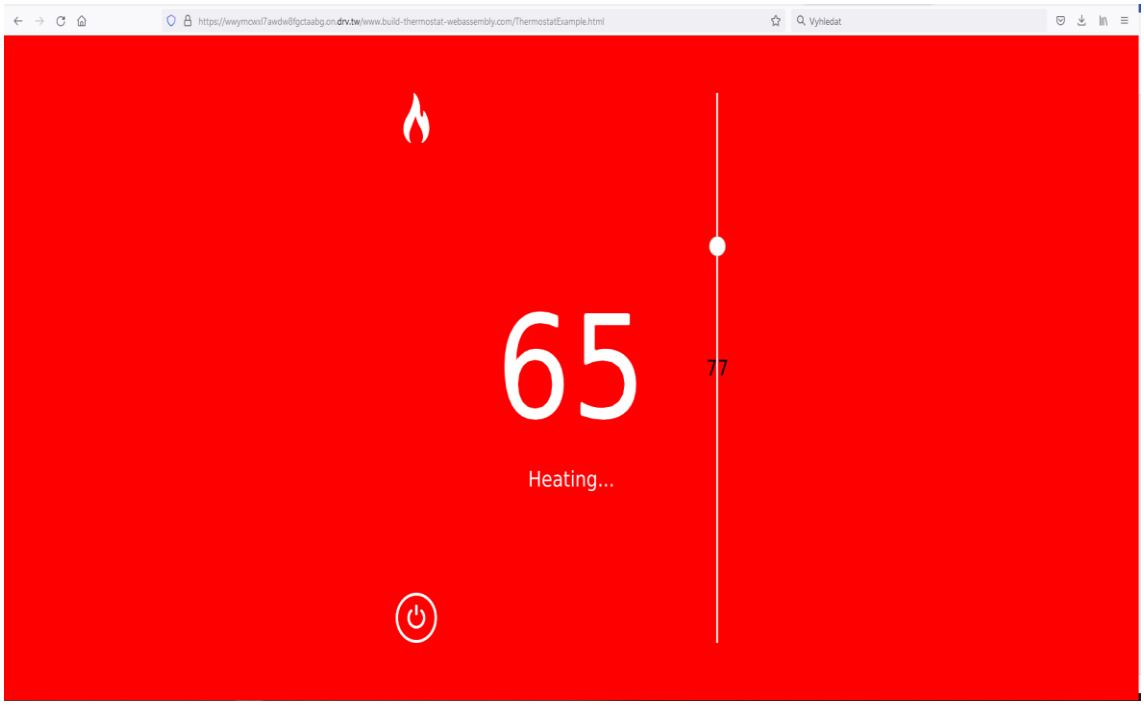
Rozklikneme Host on Google Drive, a po rozkliknutí vidíme naše zpřístupněné webové stránky.



Obr. 19 Naše webové stránky [vlastní zpracování]

Po kliknutí na požadovaný odkaz můžeme vidět naší aplikaci, která je otevřená ve webovém prohlížeči.

https://wwymcwxl7awdw8fgctaabg.on.drv.tw/www.build-thermostat-webassembly.com/ThermostatExample.html



Obr. 20 Spuštěná aplikace ve webovém prohlížeči [vlastní zpracování]

7 ZHODNOCENÍ VÝHOD A NEVÝHOD WEBASSEMBLY OPROTI JAVASCRIPTU

7.1 JavaScript

Mnoho webových stránek a e-shopů v dnešní době používá jako svůj jazyk právě JavaScript.

7.1.1 VÝHODY

7.1.2 FLEXIBILITA

Univerzálnost patří mezi hlavní důvody popularity jazyka JavaScript. Vývojáři mohou JavaScript používat pro velkou škálu možností. Většina významných elektronických obchodů jako je například eBAY a Walmart mohou používat knihovny JavaScriptu jako je třeba React k vytváření moderních úvodních stránek a zároveň využívat Node.js k napájení funkcí ze strany serveru.

7.1.3 JEDNODUCHOST

JavaScript má dobře čitelnou syntaxi pro uživatele díky které se ho lze naučit poměrně rychle. Proto se JavaScript stal preferovaným programovacím jazykem pro začátečníky.

7.1.4 RYCHLOST

JavaScript je interpretovaný jazyk a může být poměrně rychlý při použití pro dané webové úlohy dosahuje optimální rychlosti.

7.1.5 NEVÝHODY

7.1.6 ZABEZPEČENÍ NA STRANĚ KLIENTA

Použití scriptů přidává další procesy, které mohou být zneužity.

7.1.7 NEKONZISTENTNÍ PODPORA WEBOVÝCH PROHLÍŽEČŮ

Problémy začínají s kompatibilitou mezi prohlížeči, a to i přes velkou snahu o plošnou centralizaci funkcí JavaScriptu.

7.1.8 NEEFEKTIVNÍ PODPORA LADĚNÍ

Jak bylo zmíněno ve výhodách, že je JavaScript interpretovaný jazyk, který nevyžaduje kompilátor tak je snadné spustit aplikace, aniž by si uživatel všiml kritických zranitelností.

7.1.9 SHRNUTÍ

JavaScript se primárně používá pro přidání interaktivity do webových stránek a aplikací. Přes to mohou uživatelé používat JavaScript k více účelům, a to od vestavěných systémů až po desktopové aplikace. Při použití složitých projektů mohou mít uživatelé výrazné problémy s výkonem. Mezi nejběžnější použití JavaSriptu patří: Mobilní hry, vývoj mobilních aplikací, vytváření uživatelského prostředí pro webové aplikace.

7.2 WebAssembly

7.2.1 VÝHODY

7.2.2 VYSOKÝ VÝKON

WebAssembly byl navržen pro provádění vysoce výkonných příkazů velmi vysokou rychlostí blížící se nativnímu výkonu. Na rozdíl od JavaScriptu je WASM staticky typovaný to má za následek to, že optimalizace kódu probíhá mnohem dříve v procesu kompilace, než se kód dostane do prohlížeče. Binární soubory WASM jsou podstatně menší než soubory JavaScriptu, což vede k výrazně rychlejšímu načítání.

7.2.3 PODPORA

Jak bylo řečeno v úvodu, jedním z nevětších lákadel a faktorů pro zavedení WASM je to, že vývojáři mohou psát kód pro web v jiných podporovaných jazycích než v JavaScriptu a přenášet je na webové stránky. Přenositelnost patří mezi významnou vlastnost WebAssembly a proto se vyplatí pohánět efektivní, složité a výkonné aplikace mimo prohlížeč v různých operačních systémech.

7.2.4 ZABEZPEČENÍ

Od svého vzniku byl WebAssembly vytvořen s ohledem na bezpečnost. Hlavním cílem je chránit uživatele před potenciálními webovými nebezpečími a zároveň umožnit vývojářům vytvářet bezpečné aplikace. WebAssembly poskytuje zabezpečení tím, že

izoluje spouštění modulů v prostředí Sandboxu a zároveň prosazuje známé zásady zabezpečení prohlížeče.

7.2.5 NEVÝHODY

7.2.6 VÝVOJ

WebAssembly je stále v rané fázi vývoje a bude ještě trvat, než vytvoří bohaté prostředí, na které měl JavaScript 20let. WebAssembly postrádá funkce objektového modelu dokumentu (DOM) a pro plnohodnotný přístup stále spoléhá na JavaScript.

7.2.7 SLOŽITOST

WebAssembly je nízko úrovňový jazyk, což ho činí obtížným na naučení a není vhodné v něm programovat přímo, ale vyžaduje různé nástroje ke kompilaci.

7.2.8 SHRNUTÍ

Vzhledem k výhodám jazyka WebAssembly by jej vývojáři měli převážně používat při vytváření složitých aplikací náročných na výpočetní výkon. Mohou používat i zavedené nízko úrovňové jazyky uznávané především pro své výkonnostní schopnosti a celý kód zkompilovat na webové stránky pomocí WebAssembly. Nejběžnější využití WebAssembly: Aplikace pro hybridní platformy, vývoj her náročných na procesor, výkonné simulace, aplikace na streamování hudby, ukládání obrázků do mezi paměti, vývoj aplikací šifrujících citlivá data.

8 ZÁVĚR

Bakalářská práce se zabývá využitím WebAssembly v Průmyslu 4.0. Závěrečná práce je rozdělená do sedmi kapitol. V první části byla vypracována rešerše používaných řešení s kompilací klienta do WebAssembly a seznámení s aplikací. Na závěr první části jsou vypsány základní informace o aplikaci, představení anatomie WebAssembly a ukázka kódu.

Druhá část je zaměřena na rešerši podporovaných jazyků, představení kompilace kódu jazyka C++. A na závěr kapitoly je zmíněn nástroj Emscripten a ukázka jeho kompilace.

Ve třetí části je představení softwarového rozhraní OpenGL a WebGL. Na závěr je představena podpora OpenGL přes WebGL

Čtvrtá část se zabývá Průmyslem 4.0 a s jeho seznámením. Ke konci této kapitoly je zmíněno rozhraní mezi člověkem a strojem – HMI (Human Machine Interface) v kontextu s webovými technologiemi.

V páté kapitole je představena ukázková aplikace HMI v kontextu 4.0 Aplikace je zrealizována pomocí QML a backhand v C++ v prostředí Qt. Na závěr je představená aplikace pomocí nástroje Emscripten přenesena na webové stránky.

Šestá kapitola se zaměřuje na celkové shrnutí výhod a nevýhod JavaScriptu a WebAssembly v jednodušších a složitějších aplikacích.

9 SEZNAM POUŽITÉ LITERATURY

[1] Asm.js and WebAssembly. *Nurkiewicz* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://nurkiewicz.com/5

[2] Coding Tech. WebAssembly: Real World Applications. *Youtube* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://www.youtube.com/watch?v=ysFJHpS-O08

[3] WebAssembly. *Webassembly* [online]. [cit. 20-05-2022]. Dostupné z: https://webassembly.org/docs/security/

[4] Webassembly vs. JavaScript. S*nipcart* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://snipcart.com/blog/webassembly-vs-javascript

[5] WASim. *Ieeexplore.ieee* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://ieeexplore.ieee.org/document/9286089

[6] The RedMonk Programming Language Rankings. *Redmonk* [online]. [cit. 20-05-2022]. Dostupné z: https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/

[7] WebAssembly Support in Top 20 Languages. *Fermyon* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://www.fermyon.com/wasm-languages/webassembly-language-support

[8] About Emscripten. *Emscripten* [online]. ©2015 [cit. 20-05-2022]. Dostupné z: https://emscripten.org/docs/introducing_emscripten/about_emscripten.html

[9] Compiling a New C/C++. *Developer.mozilla* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/WebAssembly/C_to_wasm

[10] Introduction to OpenGL. *Users.polytech.unice* [online]. [cit. 20-05-2022]. Dostupné z: http://users.polytech.unice.fr/~buffa/cours/synthese_image/DOCS/redbookSliced1.2ps+pdf/chapterI.pdf

[11] Introduction to WebGL. *Subscription.packtpub.* [online]. [cit. 20-05-2022]. ,Dostupné z: https://subscription.packtpub.com/book/game_development/9781838644659/3

[12] WebGL: 2D and 3D graphics for the web. *Developer.mozilla.* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

[13] Industry 4.0. *Ieeexplore.ieee* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://ieeexplore.ieee.org/document/8065927

[14] Industry 4.0–Technological Revolution. *Ieeexplore.ieee* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://ieeexplore.ieee.org/document/9619363

[15] Web technologies in industry HMI. *Ieeexplore.ieee* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://ieeexplore.ieee.org/document/7329647

[16] Qt for WebAssembly. *Doc.qt.io* [online]. ©2022 [cit. 20-05-2022]. Dostupné z: https://doc.qt.io/qt-5/wasm.html

[17] Download and install Emscripten. *Emscripten* [online]. ©2015 [cit. 20-05-2022]. Dostupné z: https://emscripten.org/docs/getting_started/downloads.html

10 SEZNAM OBRÁZKŮ

11 SEZNAM TABULEK

12 SEZNAM PŘÍLOH

12.1 ELEKTRONICKÉ PŘÍLOHY

Zip obsahující:

- Projekt teploměru