ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA POČÍTAČŮ

IoT časovač

IoT timer

Bakalářská práce

Studijní program: **Softwarové inženýrství a technologie**

Vedoucí práce: **doc. Ing. Stanislav Vítek, Ph.D.**

Jakhongir Tashpulatov

Praha 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tashpulatov** Jméno: **Jakhongir** Osobní číslo: **483817**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

IoT časovač

Název bakalářské práce anglicky:

IoT timer

Pokyny pro vypracování:

Navrhněte a implementujte IoT časovač (budík). Zařízení obsahuje vlastní databázi událostí, kterou je možné plnit pomocí kombinace různých služeb (např. Google kalendář) a vlastního API. Zařízení může mít funkci klasického budíku (např. přehrání zvuku v nastavený čas), komunikovat s externími službami nebo ovládání dalších zařízení pomoci HW rozhraní. V rámci práce diskutujte možnosti zabezpečení aplikace a její komunikace s externími službami.

Seznam doporučené literatury:

[1] GAY, Warren. Raspberry Pi System Software Reference. Apress, 2014
[2] HU, Fei. Security and privacy in Internet of things (IoTs): Models, Algorithms, and Implementations. CRC Press, 2016.
[3] SERPANOS, Dimitrios; WOLF, Marilyn. Internet-of-things (IoT) systems: architectures, algorithms, methodologies. Springer, 2017.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Stanislav Vítek, Ph.D., katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **11.02.2021** Termín odevzdání bakalářské práce: **13.08.2021**

Platnost zadání bakalářské práce: **30.09.2022**

doc. Ing. Stanislav Vítek, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání Podpis studenta

Čestné prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

_____

Poděkování

Děkuji Stanislavu Vítkovi za odborné vedení práce, věcné připomínky, dobré rady, vstřícnost, a hlavně trpělivost při konzultacích a vypracovávání bakalářské práce.

Abstrakt

Cílem této práce je vyrobit IoT zařízení, které dokáže fungovat samostatně jako budík. Toto zařízení musí podporovat vzdálené a kontaktní ovládání, mít dotykový displej, reproduktor na vzbuzení, hardwarové tlačítko na ovládání zvuku, přístup na internet.

Klíčova slova: Internet of Things, Raspberry Pi, JavaScript, časovač

Abstract

The aim of this work is to create an IoT device that can function independently as an alarm clock and timer. This device must support remote and contact control, have a touch screen, speaker, hardware button for sound control and internet access for synchoronization.

Keywords: Internet of Things, Raspberry Pi, JavaScript, timer

Obsah

Kapitola 1

Úvod

Internetová technologie se rychle rozvíjí. Počet připojených zařízení ke globální síti se každým rokem zvyšuje. Poradenská divize Cisco IBSG odhaduje, že v letech 2008 až 2009 počet objektů připojených k internetu přesáhl počet lidí, do roku 2015 dosáhl počet připojených zařízení 25 miliard a do roku 2020 - 50 miliard. Došlo tak k přechodu od „internetu lidí“ k „internetu věcí“ (IoT).

Obecně se internet věcí týká nejrůznějších zařízení, senzorů, zařízení, která jsou vzájemně propojena prostřednictvím všech dostupných komunikačních kanálů, používají různé komunikační protokoly a jediný protokol pro přístup do globální sítě.

Za posledních pět let komunikační zařízení vedou chytré telefony a pomáhají při každodenních úkolech. Další jsou tablety s mobilním operačním systémem a dotykovou obrazovkou. Jakmile se internet věcí stane běžným, je snadné si představit celé kategorie spotřebitelských a podnikových aplikací zapojených do tohoto nového systému, které lze popsat výrazem „propojený život“. Patří sem automatizace budov a domů, systémy vytápění a klimatizace, řízení dopravy (například inteligentní semafory), opatření péče o starší osoby, bezpečnostní systémy, stejně jako automobily připojené k internetu a venkovní reklama.

Vývoj technologií IoT zahrnuje nejen výrobce zařízení, ale také mobilní operátory, kteří se primárně zaměřují na vývoj a implementaci nejžádanějších služeb IoT.

Ve své práci jsem dokázal vyrobit IoT zařízení sám z vlastních prostředku.

Cíl práce

Cílem práci je především návrh a výroba IoT zařízení s využitím Raspberry Pi, které má vlastnosti budíku: zvonit v určitý stanovený čas, záznam vice casu na zvonění, zobrazení aktuálního datumu a času.

Motivace

Motivace pro tuto práce byly tři:

1. Vyzkoušet různé moderní technologie a metodiky pro vývoj JavaScript aplikace.
2. Výroba IoT zařízení od plánování a pájení součástek do implementace a sestavení.
3. Vyzkoušet Raspberry Pi 4 jako základní deska pro IoT zařízení.

Sekundárním účelem práce je dokázat, že je možné použít JavaScript na vývoj nejen webových stránek, ale i na implementace IoT zařízeni. V této práci bude hodně textu věnováno Raspberry Pi, nikoliv její porovnaní s ostatními mikrokontrolery a komputery.

Kapitola 2

IoT

IoT – Internet of Things, česky – Internet Věcí, je v informatice označení pro síť fyzických zařízení, vozidel, domácích spotřebičů a dalších zařízení, která jsou vybavena elektronikou, softwarem, senzory, pohyblivými částmi a síťovou konektivitou, která umožňuje těmto zařízením se propojit a vyměňovat si data. Každé z těchto zařízení je jasně identifikovatelné díky implementovanému výpočetnímu systému, ale přesto je schopno pracovat samostatně v existující infrastruktuře internetu.

2.1 Specifikace

Internet věcí (IoT) získal širokou popularitu jak v akademickém, tak v průmyslovém kontextu. Na rozdíl od tradičních vestavěných zařízení(embedded), moderní zařízení IoT přizpůsobují operačním systémům pro všeobecné účely a umožňují vývojářům spouštět sofistikovanější aplikace napsané v jazycích vysoké úrovni, jako třeba je JavaScript. Protože zařízení IoT podléhají omezením zdrojů, jako je dostupné napájení z baterie, musíme dynamicky migrovat proces běhu mezi různými zařízeními, abychom zabránili ztrátě stavu.

Do zařízení IoT pak je možné zahrnout například kávovar, pračku, žárovku nebo lampu. Notebooky, smartphony a podobná zařízení jsou v odvětví IoT považována za koncová zařízení, a proto je nelze přímo připsat hlavním zařízením internetu věcí. Úkolem koncového zařízení je komunikovat na dálku (shromažďovat data, sledovat a sledovat aktivity) s hlavním zařízením IoT. Zařízení IoT jsou schopná provozu díky vestavěné technologii a připojení k internetu umožňuje zařízení IoT komunikovat přes internet s koncovými zařízeními, která umožňují vzdálené IoT ovládat a spravovat.

2.2 Historie

Pojem “internetu věcí” poprvé použil Kevin Ashton v roce 1999, kdy pracoval pro společnost Procter & Gamble. Spojil novou technologie RFID (Radio Frequency Identification) s internetem v dodavatelském řetězci Procter & Gamble. Definoval IoT jako: „Síť objektů ve fyzickém světě, které jsou připojeny k internetu.“[1].

K samotnému vzniku internetu věcí však došlo v letech 2008-2009, kdy podle odhadů společnosti Cisco počet připojených zařízení k internetu přesáhl světovou populaci.

Počítače byly mezi prvními zařízeními, která byla připojena k internetu. Poté následovaly smartphony a tablety.

2.3 Bezpečnost

Bezpečnost je důležitá část života každého člověka. Když se miliardy chytrých zařízení připojují k internetu, musí existovat robustní bezpečnostní mechanismy, které dostanou správné informace na správné místo ve správný čas správným kanálem. Hlavním faktorem, proč je IoT tak významnou bezpečnostní výzvou, je jeho vlastní explozivní růst. Důvodem je, že výrobci a vývojáři zařízení IoT jsou pod velkým tlakem vyrábět zařízení v co nejkratším čase s nejnižší možnou pořizovací cenou. Bezpečnostní opatření často jdou stranou. Tento problém není v žádném případě jedinečný pro IoT, ale je spojen s řadou faktorů, například s počtem zapojených zařízení.

Odborníci trvají na tom, že poskytovatelé služeb a zařízení na trhu IoT porušují zásadu end-to-end information security (IS), která se doporučuje pro všechny produkty a služby. Podle této zásady by měla být informační bezpečnost stanovena v počáteční fázi navrhování produktu nebo služby a udržována až do konce jejich životního cyklu.

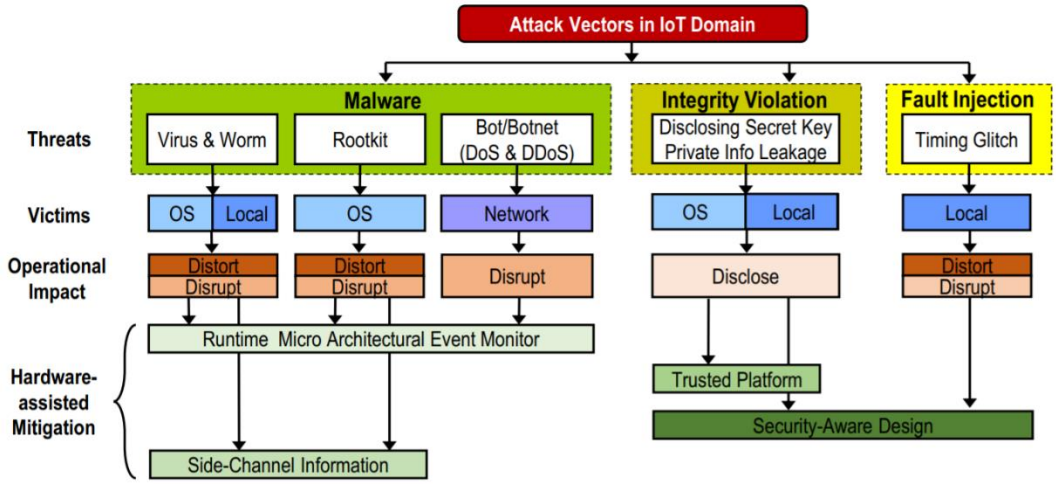
Prvky sítí IoT si mohou vyměňovat data bez přímé lidské účasti. Transformace zařízení na nezávislé internetové uzly vede k výraznému snížení bezpečnosti systému. Všechna „chytrá“ zařízení připojená k síti přes něj přenášejí data odpovídající jejich funkčnosti, což je cíl pro kyberzločince.

Zabezpečení zařízení IoT je zajištěno především udržováním integrity kódu, ověřováním uživatelů a zařízení, přiřazováním vlastnických práv uživatelům (včetně generovaných dat) a také schopností odrazit virtuální a fyzické útoky.

Před vývojem a implementací řešení zabezpečení proti různým útokům na zařízení IoT je velmi důležité pochopit, jaké schopnosti útočník má a jaké cíle sleduje. Útočník může získat fyzický přístup k jednoduchým a levným zařízením, jejichž pravidelné sledování a neustálá ochrana nemusí být vždy praktické a finančně proveditelné. Fyzická kontrola nad takovými zařízeními otevírá příležitosti pro útoky na vedlejší kanály, umožňuje hardwarové chyby a trojské koně a nahrazuje je falešnými zařízeními [2].

Hlavním cílem kybernetických útoků je zkreslení výstupu (a následných akcí v důsledku nesprávných údajů) zařízení nebo narušení současných procesů (odmítnutí služby) nebo zveřejnění citlivých informací uložených v zařízení, jako jsou tajné klíče a hesla. Moderní zařízení shromažďují obrovské množství dat o svých uživatelích. Některé z nich vyžadují k práci nejen heslo, ale také uživatelské jméno, jeho kontakty, informace o jeho biografii. Toto množství informací vyžaduje spolehlivou a kvalitní ochranu, což v tuto chvíli se IoT nemůže slíbit.

Jak ukazuje obrázek č. 1, zařízení IoT může být napadeno následujícími typy:



Obrázek 1. Běžné kybernetické útoky pro zařízení IoT a techniky zmírňování pomocí hardwaru

2.3.1 Malware

Moderní zařízení IoT mohou být v různých fázích provozu infikována různým malwarem. Ve většině případů se malware (viry, trojské koně a červi) obvykle zaměřuje na místní exploataci a exploataci na úrovni operačního systému, v závislosti na složitosti útoku a jeho provedení. Hlavním úkolem malwaru je narušit současné operace a převzít kontrolu nad zařízením. Nejčastějšími zbraněmi útočníků jsou rootkity (sada nástrojů, které si hacker nainstaluje na zařízení, které kompromitoval po získání počátečního přístupu, což hackerovi umožní získat oporu v kompromitovaném systému a skrývat stopy jeho aktivit). Poskytují hackerům trvalý privilegovaný přístup do systému a současně aktivně skrývají svou přítomnost. Útočník tím převezme veškerý výpočetní výkon a data zařízení a může také umístit ovladače a služby, které jsou pro uživatele neviditelné. Zařízení IoT se navíc mohou stát kořistí útoků typu odmítnutí služby (DoS) nebo distribuovaného odmítnutí služby (DDoS), které se každým dnem stávají vážným problémem. Takové zranitelné zařízení může fungovat jako robot (nebo „zombie“) k infikování jiných legitimních zařízení v síti nebo může spotřebovávat šířku pásma sítě a výpočetní výkon zařízení, což útočníkovi poskytne další zdroje.

2.3.2 Zveřejnění soukromého klíče

Získání přístupu k soukromému klíči (používanému pro šifrování) a / nebo k osobním informacím uloženým v zařízení IoT je pro útočníka lahůdka ke kompromitaci „kořenů důvěry“ systémů. To umožňuje útočníkovi převzít kontrolu nad komunikačními procesy, zmocnit se výpočetního výkonu zařízení, a hlavně důvěrných informací.

2.3.3 Softwarově řízené zadávání poruch

Další třídou útoků proti zařízením IoT je softwarové vkládání (anglický - Fault Injection) selhání do hardwaru, když je zařízení spuštěno. I když je tento typ útoku poměrně složitý, protože vyžaduje důkladnou znalost hardwaru specifického pro platformu a základního softwaru, je náprava takového útoku velmi obtížná. Protože tato třída útoků hledá a využívá drobné zranitelnosti hardwaru, je zbytečné používat čistě softwarové obranné mechanismy.

Samozřejmě existují účinná softwarová řešení ke zmírnění stávajících bezpečnostních problémů zařízení IoT. Sofistikovaným útokům a kybernetickým hrozbám však nelze vždy zabránit pomocí takových softwarových metod, protože:

1. Mechanismus softwarové ochrany použitý v samotném zařízení IoT může být zranitelný vůči vzdáleným útokům. Navíc nemusí mít nutně širokou ochranu, často jej lze obejít a hacknout bez vědomí uživatele. Útok Double Agent například ohrozil mnoho známých antivirových programů, což vedlo k narušení integrity informací o zařízeních [3].
2. Softwarová řešení vyžadují pravidelné aktualizace a opravy, zatímco malware se neustále vyvíjí a vyvíjí. Navíc aktualizace softwaru mnoha vzdálených zařízení IoT není vždy fyzicky proveditelná. Proto je nutné dodržovat koncept „nastavit a zapomenout“. V tomto se hardwarová řešení mohou dobře osvědčit.

2.3.4 Metody zabezpečení hardwaru

Zabezpečení hardwaru se ukázalo jako alternativa k čistě softwarovému obrannému mechanismu, protože tento samotný neposkytuje úroveň zabezpečení požadovanou pro moderní zařízení IoT. Hardwarové zabezpečovací systémy používají hardwarové moduly a mohou shromažďovat mikroarchitekturní informace k analýze převládajících hrozeb a zranitelností na softwarové úrovni. Jak ukazuje obrázek 1, hardwarové metody poskytují širokou škálu řešení pro bezpečné a spolehlivé aplikace IoT.

2.3.4.1 TPM

Jedním z hlavních požadavků pro provádění zabezpečené transakce informací mezi zařízeními IoT přes nedůvěryhodnou síť je použití spolehlivé a bezpečné schématu správy klíčů a zpracování dat na zařízení. V tomto ohledu modul Trusted Platform Module (TPM je název specifikace popisující kryptoprocesor, který ukládá kryptografické klíče k ochraně informací, a také obecný název implementací zadané specifikace). Tyto TPM (obrázek č.2) umožňují použití kryptografických klíčů, které lze svázat s konkrétními parametry platformy a chránit před

odhalením jakoukoli jinou nedůvěryhodnou hardwarovou komponentou, procesem nebo softwarem. Diskrétní čipy TPM (dTPM) a plastové moduly plošných spojů mohou nabídnout větší pokrytí službou tím, že umožní sdílení zdrojů mezi více aplikacemi na jednom fyzickém počítači. Architektury podporující ARM TrustZone a Intel Software Guard Extension (SGX) navíc přidávají nové funkce do moderních SoC (systém na čipu) a poskytují spolehlivé a zabezpečené prostředí pro spouštění procesů kritických pro zabezpečení, přestože privilegované jádro a software jsou potenciálně zlomyslný [4].

2.3.4.2 Mikroarchitekturní monitorování událostí

Zatímco TPM a další kryptografické systémy odolnosti nabízejí robustní prostředí pro aplikace citlivé na zabezpečení, takový hardware bývá drahý, energeticky náročný a nevhodný pro lehká a levná zařízení IoT. Navíc malware, jako jsou viry, trojské koně a roboti, může neviditelně infikovat zařízení obcházející takové systémy, pokud síť není dostatečně zabezpečena. Je velmi obtížné detekovat malware bezprostředně po infekci, protože může obejít antivirové programy v zařízení. V takových případech může pomoci hardwarové monitorování událostí mikroarchitektury a systémů SIEM (Bezpečnostní informace a správa událostí). Nabízí jemné filtrování pro jednotlivá spuštění, může shromažďovat vícerozměrné informace a poskytuje rychlejší časy odezvy než protějšky softwaru proti malwaru.

Srdcem těchto hardwarových monitorů jsou jednotky monitorování výkonu (PMU)

Obrázek 2. TPM2 modul

dostupné v moderních procesorech a SoC. Hlavním účelem PMU je poskytnout vhled do výkonu procesoru protokolováním sady mikroarchitekturních událostí a souvisejících počtů pomocí integrovaných čítačů výkonu hardwaru (HPC). Například jeden nebo více HPC v PMU může určit, kolikrát se během provádění programu vyskytne předem stanovená událost (povolená příslušnou architekturou), například chyby mezi paměti, aby se vyhodnotil výkon testovaného

systému. PMU na architektuře ARM a Intel x86 lze ovládat pomocí softwarových modulů, jako je nástroj Linux Perf [5]. Poskytuje zpětnou vazbu v reálném čase pro diagnostiku chyb nebo identifikaci úzkých míst softwaru. PMU byl původně navržen pro monitorování výkonu, ale je také inteligentně používán v systémech SIEM, což výrazně zrychluje zpracování incidentů zabezpečení informací a také pomáhá detekovat útoky a další hrozby pro prvky infrastruktury. Další výhodou je, že PMU jako integrovaný hardware funguje transparentně s jakýmkoli softwarem běžícím na procesoru a nemůže být podveden externím malwarem. To znamená, že samotný monitor hardwaru nevěnuje pozornost procesům. Protože jakýkoli malware, nebo dokonce upravený firmware nebo rootkit, musí provádět určité akce, monitorování událostí PMU má potenciál takové škodlivé akce detekovat.

Vývojáři z NYU Polytechnic School of Engineering, Brooklyn, New York, USA navrhli hostitelský rámec detekce DDoS nazvaný BRAIN (BehavioR based Adaptive Intrusion detection in Networks). Používá hardwarové funkce k simulaci bezpečného chování a útoků DDoS. K detekci útoků DDoS využívá techniky strojového učení k modelování chování aplikací a statistik sítě. Protože korelace mezi statistikami sítě a aplikací s daty HPC není triviální, je nutné vybrat hardwarové události s vysokou přesností. Autoři navrhli implementovat integrovaný mechanismus detekce útoků DDoS (DDoSDE), který monitoruje chování hardwaru i sítě. Rozhraní DDoS Prevention Interface (DDoSPI) reaguje na jakýkoli zjištěný útok zařazením IP adres do Black listů (a případně odebráním) na základě prahových hodnot založených na dynamické síti a HPC, čímž brání útočníkovi naučit se bezpečnostní kritéria a zásady zařízení [6].

2.4 Oblast využití

Vzhledem k současnému vývoji, technologické společnosti nepředpokládají, že by se růst IoT technologií měl v přechodném období zpomalit. Do roku 2025 se očekává, že množství IoT připojených zařízení přesáhne 75 miliard. V důsledku to znamená, že IoT se může vyskytovat ve všech částech života, což přináší zcela novou úroveň bezpečnostních rizik. Stále větší část IoT zařízení je vytvářena pro běžného spotřebitele [7] .

2.4.1 Průmyslový obor

2.4.1.1 Zdravotnictví

Ve zdravotnictví má IoT celou řadu aplikací: od vzdálených monitorovacích zařízení až po pokročilé a inteligentní senzory pro integraci zařízení. To má potenciál zlepšit způsob, jakým lékaři poskytují péči. Zdravotnická zařízení IoT mohou pacientům umožnit sdílení zdravotních informací lékařem v reálném čase. Od osobních senzorů fitness až po chirurgické roboty přináší IoT ve zdravotnictví nové nástroje, které představují nejnovější technologie.

IoT pomáhá revoluci ve zdravotnictví a poskytuje pacientům a zdravotnickým pracovníkům cenově dostupnější řešení. Moderní nemocnice vyžadují funkčnost softwaru a hardwaru, protože některá zařízení se používají k záchraně nebo udržení lidského života. Stejně jako všechna elektronická zařízení je i toto zařízení náchylné k mnoha rizikům, od výpadků proudu až po poruchy systému, které mohou být otázkou života nebo smrti.

2.4.1.2 Chytrá města

Smart Cities pokrývají širokou škálu použití: od distribuce vody, řízení dopravy až po odpadové hospodářství, monitorování životního prostředí a městskou bezpečnost. Důvodem, proč se koncepty inteligentních měst stávají tak populární, je to, že se snaží eliminovat nepohodlí a problémy lidí žijících ve městech. Řešení IoT nabízená v oblasti inteligentních měst řeší různé dopravní problémy, snižují znečištění ovzduší a hluk a pomáhají zvyšovat bezpečnost měst.

Sběr odpadu je jednou z oblastí komunálních služeb, na které města a společnosti IoT obrátili oči. Celý proces spočívá ve skutečnosti, že v každé popelnici jsou prediktivní senzory, které zaznamenávají, kdy bude nutné sbírat odpad. Tyto senzory také předpovídají, kdy je nejpravděpodobnější, že dané množství odpadu bude třeba shromáždit. Data z těchto senzorů jsou přístupná prostřednictvím připojení k internetu a jsou zadávána do geografického informačního systému, který na základě získaných dat optimalizuje trasy sběru odpadu s důrazem na časový harmonogram.

2.4.2 Spotřebitelský obor

2.4.2.1 Chytrá domácnost

Smart Home je možná nejdůležitější oblastí aplikace pro internet věcí. Počet lidí, kteří používají prvky inteligentních domů se každý měsíc zvyšuje o 60 000. Společnosti vyvíjejí aplikace. Chytré domy jsou nyní mnohem více zapojeny než jakákoliv jiná zařízení IoT [8].

Smart Home může zahrnovat elektroměr se zařízením IoT, které umožní sledovat denní spotřebu vody, set-top box, který umožní nahrávat programy dálkové, automatické osvětlovací systémy, pokročilé uzamykací systémy a sledovací systémy. Vzhledem k tomu, jak rychle se internet věcí vyvíjí, je pravděpodobné, že většina zařízení bude chytřejší než dříve (původní zařízení bez integrace IoT), což zajistí dobrou domácí bezpečnost.

2.4.2.2 Nositelná elektronika

Wearable devices - jsou inteligentní elektronické zařízení s mikrokontrolery. Zařízení lze nosit na těle jako implantát nebo jako příslušenství. Spotřebitelé po celém světě každý rok čekají

na vydání nového modelu chytrých hodinek od společnosti Apple. Kromě toho existuje mnoho dalších nositelných zařízení IoT, která usnadňují život, například Sony Smart B Trainer, náramek LookSee, gesta Myo, Xiaomi Mi Bands. Implantátem pak může být například mikročip nebo chytré tetování.

Kapitola 3

Analýza

Tato kapitola se zaměřuje na návrh systému. Návrh je klíčovou částí při vývoji. V případě špatného navržení systému se mohou tyto nedostatky projevit během implementace nebo později při rozšiřování systému. Může to zapříčinit nepoužitelnost aplikace nebo neschopnost reagovat na budoucí požadavky a rozšíření. Tato skutečnost je častým problémem a systémy tak mohou přestat být konkurence schopnými vůči ostatním systémům, které na nové požadavky dokážou reagovat [9].

3.1 Požadavky

Veškeré systémové požadavky, které od systému požadujeme lze rozdělit do dvou základních kategorií. Těmito kategoriemi jsou funkční požadavky a nefunkční požadavky. V následujících podkapitolách si je popíšeme.

3.1.1 Funkční požadavky

Funkční požadavky popisují chování systému. Jde o služby a akce, které po systému požadujeme, aby splňoval. Vzhledem k tomu, že je systém složen ze tří částí, rozdělíme požadavky podle toho, do kterých částí systému spadají. V našem případě máme tyto části: hardware, server a webová aplikace.

1. Hardware
a. Musí umet zobrazovat informaci uživateli na displeji
b. Musí mít ovládací tlačítko na hlasitost a jas
c.Musí umet přehrávat melodie s určitou hlasitosti
d. Musí svítit zdrojem: LED nebo žárovka
e. Musí zjistit aktuální teplotu v aktuálně lokalitě
2. Server
a. Ukládat události do databáze
b. Synchronizace události s Google Calendar
3. Webová aplikace
a. Přehrávat melodii ve stanoveny čas
b. Komunikovat se serverem a zpracovávat požadavky

3.1.2 Nefunkční požadavky

Nefunkčními požadavky jsou ty, podle kterých lze měřit kvalitu vyvíjeného produktu. Jsou to určitá pravidla a omezení kladená na samotný produkt nebo na jeho vývoj. Doplňují funkční požadavky.

1. Hardware
a. Displej musí být dotykový
b. Tlačítko musí být otočné
2. Server
a. Backendová část musí být napsaná v JS
3. Webová aplikace
a. Frontendová část musí být napsaná v JS
b. Přehrávat hudbu z YouTube

4. Dostupnost
a. Webové rozhraní musí být dostupné 24/7
5. Bezpečnost
a. Aplikace musí být bezpečné a mít omezeny dostup na internet
b. Zařízeni musí být nastavitelné v rámci sítě

3.2 Doménový model

Rozlišujeme 2 druhy události - alarm a event.

3.2.1 Event

Event je vnitřní událost, která se ukládá do databáze a nese význam připomínky nebo události, která trvá nějakou dobu, např: 1 hodinu, 2 měsíce, 3 týdny. Musí mít následující vlastnosti:

- Datum a čas začátku
- Datum a čas zakončeni
- Název
- Melodie
- Barva

Seznam eventu je možné synchronizovat s Google Calendar pomocí Google Calendar API.

3.2.2 Alarm

Alarm je vnitřní událost, která se ukládá do databáze a má význam klasického buzeni nebo připomínky. Musí mít následující vlastnosti:

- čas
- status
- název
- opakovatelnost
- melodie

3.3 Případy užití

V této části se zaměříme na případy použití, které popisují chování zařízení z pohledu uživatele. Uživatel je považován za aktéra, který interaguje se zařízením určitým způsobem, aby dosáhl určitého cíle. Případy použití ukazují, kteří herci interagují s danými částmi systému a jaké

akce mohou provádět. Případy použití obvykle popisují pouze funkční požadavky. Tato metoda je velmi důležitá a nezbytná při vývoji softwaru. Hlavním účelem těchto případů použití je velmi dobře reflektovat detaily jednotlivých funkcí. Díky nim lze snadno odhadnout, kolik času a práce bude věnováno vývoji. Jsou také velmi užitečné při vytváření vlastní dokumentace a testovacích skriptů (víme přesně, co testovat).

Evidujeme pouze jednoho aktéra – uživatel čili vlastník zařízení.

3.3.1 Obecné funkce

- Zapnout a vypnout zařízení
- Upravit hlasitost melodii
- Upravit jas zdroje světla

3.3.2 Alarm a Event

S každou tuto události (Alarm a Event) uživatel by měl umět provádět následující akce:

- Založit záznam o nadcházející události
- Upravit existující záznam události
- Smazat existující záznam události
- Načíst seznam události

Kapitola 4

Raspberry Pi

Raspberry Pi je rodina levných jednodeskových počítačů původně navržených pro počítačové a elektronické vzdělávání. Díky kombinaci nízkých nákladů, flexibility a široké podpory operačního systému se rodina Raspberry Pi stala jednou z předních počítačových platforem pro fanoušky i pedagogy.

4.1 Úvod a historie

Raspberry Pi je známý jako jednodeskový počítač, což znamená přesně to, jak vypadá: je to počítač, stejně jako stolní počítač, notebook nebo smartphone, ale postavený na jedné desce s plošnými spoji. Jako většina počítačů s jednou deskou je Raspberry Pi malý - přibližně stejně velký jako kreditní karta - ale to neznamená, že není výkonný: Raspberry Pi zvládne vše, co zvládne větší počítač, který má větší spotřebu. Rodina Raspberry Pi se zrodila z touhy podporovat více praktického počítačového vzdělávání po celém světě. Jeho tvůrci, kteří se spojili a vytvořili neziskovou nadaci Raspberry Pi, netušili, že bude tak populární: Několik tisíc postavených v roce 2012 na testování vody bylo okamžitě vyprodáno a miliony byly odeslány do celého světa. V letech od té doby se tyto desky dostaly do domácností, tříd, kanceláří, datových center, továren, a dokonce i samo pilotních lodí a vesmírných balónů. Od původního modelu B byly vydány různé modely Raspberry Pi, každý s vylepšenými funkcemi nebo s funkcemi specifickými pro konkrétní případ použití. Rodina Raspberry Pi Zero je například malou verzí Raspberry Pi v plné velikosti, která postrádá některé funkce - zejména více portů USB a kabelový síťový port - ve prospěch mnohem menšího rozložení a nižších požadavků na napájení [10].

Eben Upton, vývojář Raspberry Pi, zaznamenal při práci v počítačové laboratoři Cambridgeské univerzity v roce 2006 pokles úrovně dovedností. Studenti hlásící se ke studiu informatiky začali mít méně zkušeností s programováním než studenti v minulosti. Upton a jeho univerzitní kolegové měli myšlenku postavit počítač, který byl dodáván se všemi nástroji potřebnými k jeho programování a prodán za cílovou cenu 25 USD (asi 20 GBP). Muselo to umět i jiné zajímavé věci, aby to lidi přitahovalo k používání, a muselo to být dostatečně silné. Tato myšlenka odstartovala šestiletou cestu. První Raspberry Pi byl vydán v únoru 2012 a do konce čtvrtletí se ho prodalo půl milionu kusů. V červenci 2017 bylo v domácnostech, školách a na pracovištích více než 14 milionů malinových velikonočních vajíček, z nichž 10 milionů bylo vyrobeno ve Velké Británii. Je to celkově nejprodávanější britský počítač všech dob.

4.2 ARM versus x86

Jádrem systému Raspberry Pi je multimediální procesor typu SoC (system-on-chip) Broadcom BCM2XXX. To znamená, že většina systémových komponent, včetně jeho hlavního a grafického procesoru spolu se zvukovým a komunikačním hardwarem, je integrována do jediné součástky ukryté pod paměťovým čipem s kapacitou 256 MB až 8 GB uprostřed základní desky.

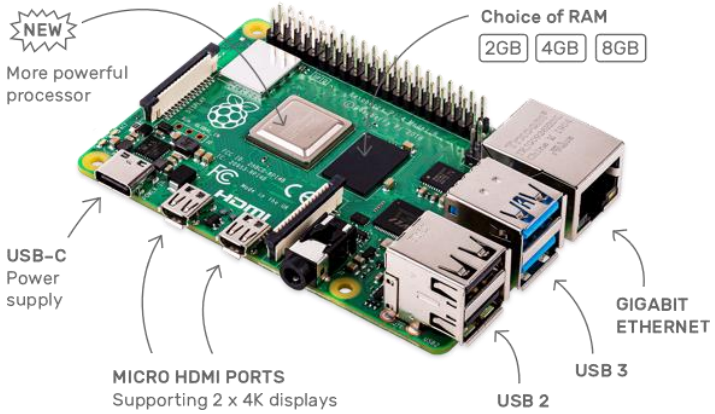
Procesor BCM2XXX se od procesorů, kterými jsou vybaveny stolní počítače nebo notebooky, však neliší jen svým návrhem typu SoC. Používá také jinou architekturu instrukční sady (ISA – instruction set architecture), která se označuje jako ARM. Architektura ARM, kterou již v 80. letech vyvinula společnost Acorn Computers, se v počítačích třídy PC uplatňuje poměrně zřídka. Vyniká však v mobilních zařízeních: telefon, který nosíte v kapse, je téměř určitě vybaven alespoň jedním výpočetním jádrem typu ARM. Vzhledem ke kombinaci jednoduché architektury

s redukovanou instrukční sadou (RISC – reduced instruction set) a nízké spotřeby energie představuje v mobilních zařízeních mnohem výhodnější volbu než procesory stolních počítačů, které se vyznačují vysokými nároky na napájení a architekturou s komplexní instrukční sadou (CISC – complex instruction set).

4.3 Přehled modelu

Nejsou jen čtyři různé generace Pi, ale také řada modelů. Raspberry Pi 4, viz obrázek č. 3, je k dispozici jako model B a 3 se prodává jako model A, model B a model B +, v podstatě přetaktovaný model B s rychlejším Wi-Fi. Mezitím je Raspberry Pi 2 k dispozici jako model B a Raspberry Pi 1 jako model B a model A s nižšími specifikacemi. Model A nemá ethernet, má méně paměti než model B a má pouze jeden port USB. Prodává se však za nižší cenu 25 USD a spotřebovává méně energie [11].

Obecně je Pi 4 Model B lepší volbou než Pi 3, protože nabízí lepší specifikace za stejnou cenu. Přestože je Pi 1 mnohem méně výkonný, je levnější než Pi 3 a je k dispozici také v kompaktnějším a méně energeticky náročném modelu A.



Obrázek 3. Raspberry Pi 4 4 GB

Cena, malá velikost a nízká spotřeba energie Pi Zero znamenají, že má zjevná omezení ve srovnání s většími sourozenci. Má pouze jeden port USB On The Go a původní Pi Zero postrádá připojení k síti. Raspberry Pi Zero W za 10 $ však podporuje Wi-Fi 802.11b / g / n a Bluetooth 4.0. Pi Zero je méně vhodný pro použití jako PC a je vhodnější pro balení do samostatného zařízení IoT nebo automatizovaného zařízení, jako je meteorologická stanice, kde je omezený prostor nebo je vyžadována minimální spotřeba energie. Pokud však je potřeba připojit Zero k DIY PCB a jinému hardwaru DIY, je nutné muset pájet piny k nepoužitému GPIO záhlaví na desce.

Tabulka 1. Přehled a porovnaní všech modelu Raspberry Pi

Generace	Model	SoC	Paměť	Ethernet	Wi-Fi	GPIO	Rok
Raspberry Pi	B	BCM2835	256 MB	Ano	Ne	26pin	2012
	A			Ne		40pin	2013
	B+		512 MB	Ano			2014
	A+			Ne			2014
Raspberry Pi 2	B	BCM2836/7	1 GB	Ano			2015
Raspberry Pi Zero	Zero	BCM2835	512 MB	Ne			2015
	W/WH				Ano		2017
Raspberry Pi 3	B	BCM2837A0/B0	1 GB	Ano	Ano		2016
	A+	BCM2837B0	512 MB	Ne			2018
	B+		1 GB	Ano			2018
Raspberry Pi 4	B	BCM2711	1,2,4,8 GB				2019
	400		4 GB				2020
Raspberry Pi Pico		RP2040	264 KB	Ne	Ne	26pin	2021

4.4 CPU a RAM

Ačkoli většina desek Raspberry Pi Model B vypadá velmi podobně, mezi všemi modely Raspberry Pi existují velké rozdíly v jádrech procesoru, rychlosti procesoru, 32bitové a 64bitové podpoře, velikosti RAM a rychlosti RAM [12].

Všechny modely Raspberry Pi používají technologii SoC, která kombinuje CPU, video a další funkce, které se běžně nacházejí na samostatných čipech, do jednoho kusu křemíku. Na deskách s 256 MB nebo 512 MB RAM je čip RAM naskládán nad SoC. Tedy SoC a RAM jsou instalovány jako jedna podsestava. Na deskách s 1 GB RAM je čip RAM připevněn ke spodní části desky s připojením k SoC procházející deskou. Výsledkem je sendvič se třemi vrstvami při pohledu shora:

- SoC
- Deska Raspberry Pi 2 nebo 3
- RAM

Uživatelé musí vyměnit své desky, aby získali rychlejší výkon CPU nebo RAM. Naštěstí jsou desky Raspberry Pi levné. Desky Raspberry Pi používají čipy Broadcom BCM2xxx SoC s různými jádry procesoru a množstvím paměti RAM v závislosti na úrovni desky nebo revize.

4.5 Porty

Porty zahrnuté v modelech Raspberry Pi:

- USB 2.0
- USB 3.0 pouze v Raspberry Pi 4
- HDMI v1.3
- Audio vystup
- 10/100/1000 Ethernet pouze v některých modelech
- Micro-USB nebo USB-C konektor pro napájení

4.6 Board-Level connectors

Mezi konektory na úrovni desky zahrnuté v modelech Raspberry Pi patří:

- Camera interface (CSI)
- Display interface (DSI)
- GPIO Father connector (vše kromě Raspberry Pi Zero, Zero W) nebo GPIO Mother pinout (Raspberry Pi Zero, Zero W)
- SD card nebo microSD card

CSI používá kamera Raspberry Pi, která je k dispozici ve verzích 5MP nebo 8MP. DSI používají různá zobrazovací zařízení, včetně displeje.

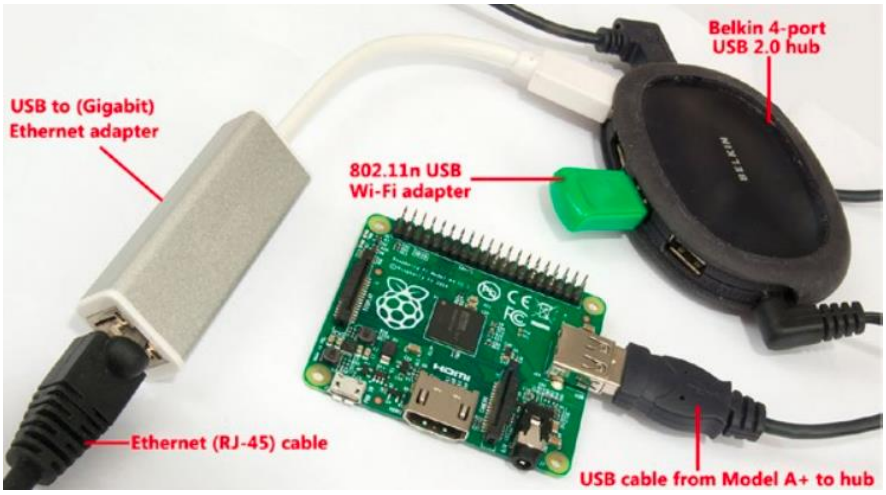
4.7 Integrované síťové funkce

Pokud se uvažuje o připojení k síti, lze desky Raspberry Pi rozdělit do tří kategorií:

1. Desky bez integrované sítě
2. Desky s integrovanou kabelovou sítí

3. Desky s integrovanou bezdrátovou sítí

Raspberry Pi Model A, A+ a Zero neobsahují žádné integrované sítě. S těmito deskami lze použít bezdrátový adaptér USB nebo kabelový ethernetový adaptér USB. Pokud je potřeba používat více než jedno zařízení USB současně, lze použít rozbočovač (HUB) USB 2.0



Obrázek 4. Raspberry Pi A+ používá 4 portový USB rozbočovač Belkin k připojení k USB Wi-Fi adaptéru a USB ethernetovému adaptéru

4.8 Zdroj energie

Minimálně každý Raspberry Pi v plné velikosti by měl používat napájecí zdroj 2 A. Pro větší spolehlivost se však doporučuje napájecí zdroj 3 A, zejména pokud bude použit SPI Pi, rozhraní fotoaparátu nebo piny GPIO.

Kapitola 5

Hardware

Kromě samotné desky Raspberry Pi, IoT zařízeni je třeba vybavit dalšími komponenty a moduly. V této kapitole popíšeme moduly, součástky a komponenty, které byli použité při návrhu řešení.

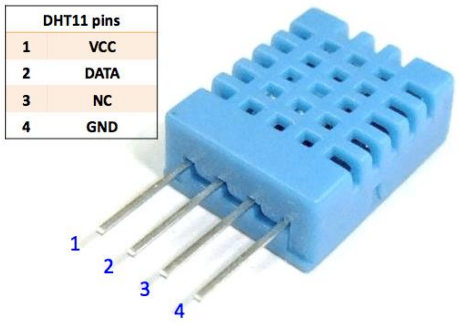
5.1 Moduly a Senzory

Bohužel, Raspberry Pi 4 nemá zadně vybavené senzory a moduly. Má pouze Wifi a Bluetooth modul a další porty, přes které je možné připojit další zařízeni. Ale jsou tam GPIO piny, na které se da instalovat externí moduly, senzory a další komponenty, jako například: dioda, teploměr, mikrofon, reproduktor, displej a další.

Jeden z užitečných senzoru je teploměr.

5.1.1 Teploměr

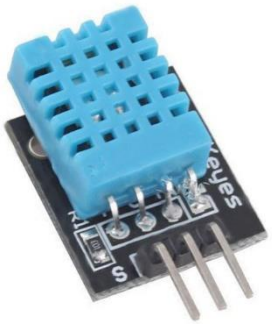
DHT11 je jeden z nejpopulárnějších senzoru na měřeni teploty a vlhkosti. DHT11 lze spojit s mikrokontrolery jako Arduino, Raspberry Pi atd. DHT11 je snímač teploty a vlhkosti s minimálním úsilím. Poskytuje vysokou kvalitu a dlouhodobou spolehlivost [13].



Obrázek 5. DHT11 senzor na měřeni teploty

Senzor DHT11 lze zakoupit buď jako senzor, nebo jako modul. V každém případě je výkon snímače stejný. Senzor bude dodáván jako 4pinový balíček, ze kterého budou použity pouze tři piny, zatímco modul bude mít tři piny, jak je uvedeno výše. Jediným rozdílem mezi snímačem a modulem je to, že modul bude mít vestavěný filtrační kondenzátor a výsuvný odpor a pro snímač je musíte v případě potřeby použít externě [14].

Senzor může měřit teplotu od 0 ° C do 50 ° C a vlhkost od 20 % do 90 % s přesností ± 1 ° C a ± 1 % [15]. Schéma připojení tohoto snímače je uvedeno na obrázku.



Obrázek 6. DHT11 modul

5.2 Display

K tomu, aby IoT zařízeni ukazovalo aktuální čas, datum a další informace je potřeba nainstalovat displej. Existuje mnoho řešeni od různých dodavatelů.

Displej musí být dotykový s úhlopříčkou do 7”. V tabulce č. 2 je vidět porovnání některých displeji od největšího výrobce displejů pro Raspberry Pi – Waveshare a originálního displeje od výrobce Raspberry Pi.

Tabulka 2. Porovnaní displejů pro Raspberry Pi

Displej	Waveshare 4.3”	Raspberry Pi 7”	Waveshare 3.5” LCD (B)	Waveshare 5.5"
Rozlišení	800×480	800x480	320×480	1920x1080
Konektivita	DSI	DSI	SPI	HDMI
Snímání dotyku	Kapacitní	Kapacitní	Rezistivní	Kapacitní
Panel displeje	IPS	IPS	IPS	AMOLED
Cena*	1109,00 Kč	1899,00 Kč	699,00 Kč	3 179,00 Kč

Displej Waveshare 3.5” LCD (B) má SPI rozhraní, což znamená, že se přípoji k Raspberry Pi pomoci GPIO pinu. Vzhledem k tomu, že do některých pinu je potřeba připojit další moduly a seniory, tak tento typ displejů není vhodný.

Displej Waveshare 5.5" má konektor HDMI. Raspberry Pi 4 nemá klasický HDMI vystup, ale má HDMI mini. Znamená to, že bude potřeba to připojovat pomoci adapteru. Vzhledem k tomu, že zařízeni má byt kompaktní, tak tento typ konektoru taky nebudeme brát v úvahu při rozhodovaní.

Po zvážení všech výhod a nevýhod byl vybrán displej Waveshare 4.3” s DSI konektorem. Ten byl vybrán kvůli nízké ceně a dostupnosti na skladě dodavatele.

5.3 Reproduktory

Raspberry Pi 4 nedisponuje reproduktorem. Proto tento prvek bude potřeba doinstalovat. Pro potřeby zvoněni bude stačit 2 reproduktory s výkonem až 0.5 W, které možné zakoupit v e-shopech [16].

Obrázek 7. Reproduktory s výkonem 0.5 W

5.4 Rotary Encoder

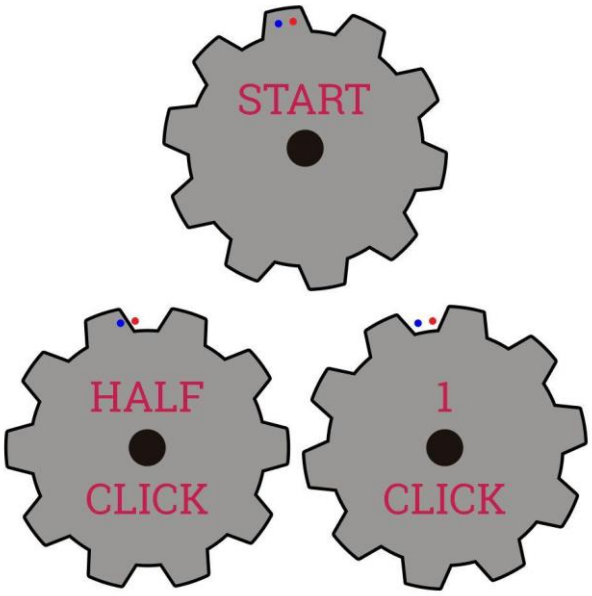
Rotační enkodér je zařízení, které snímá otáčení a směr připojeného knoflíku.

Funguje to tak, že má 2 interní kontakty, které při otáčení knoflíkem rozpojí a rozpojí obvod. Když otáčíte knoflíkem, cítíte, jak „cvaká“, což znamená, že byla otočena jedna poloha. Pokud by vnitřní kontakty byly původně HIGH po jediném kliknutí, nyní by byly oba LOW. S jednoduchou logikou je možné zjistit směr otáčení.

Enkodér ma 3 stavy (viz obrázek č. 9) [17]:

1. Start
2. Half Click
3. Click

Obrázek 8. Rotary encoder



Obrázek 9. Stavy enkoderu

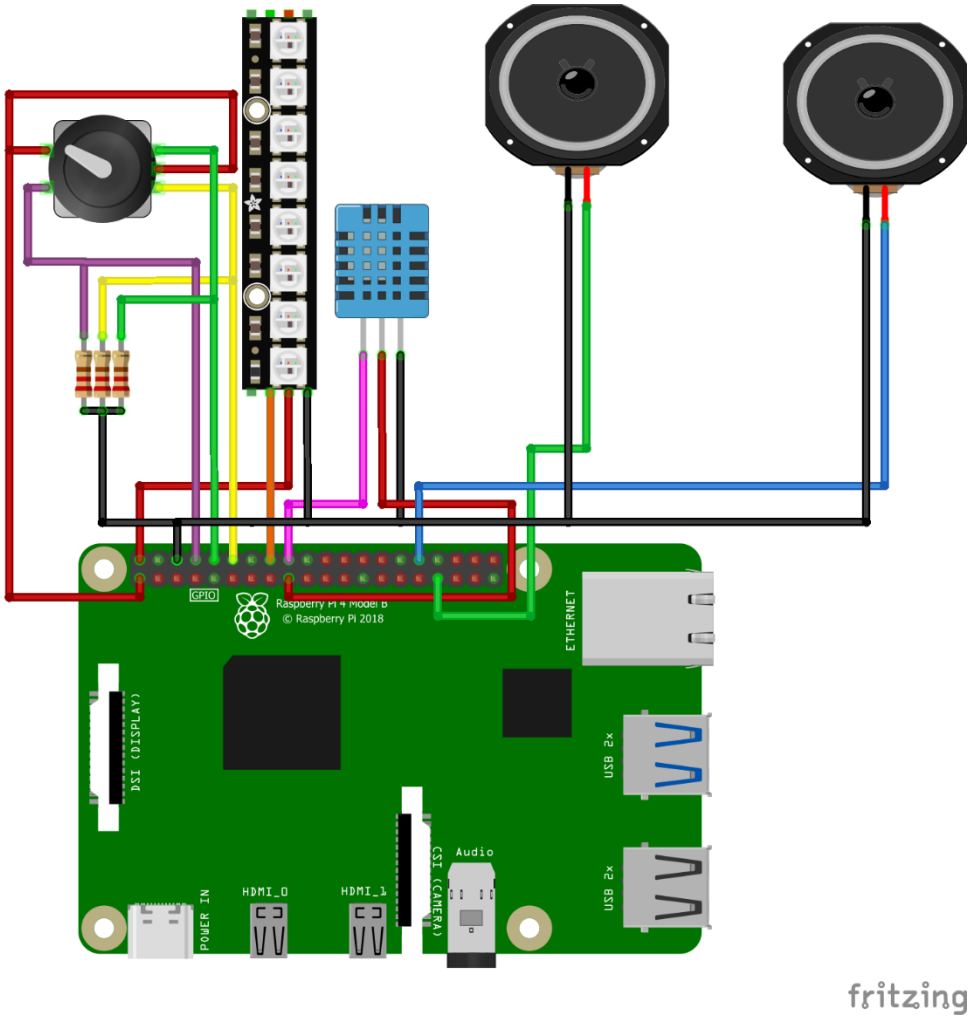
5.5 LED páska

Bylo by dobrý, aby zařízeni mohlo svítit v noci. Proto je třeba instalovat svíticí zdroj, jako například žárovka nebo LEDky. Žárovka je příliš velká pro takové zařízeni, proto budeme uvažovat LEDky, konkrétně několik LEDek propojených mezi sebou. Narazíme tím pádem na LED pásku. Taková LED páska ma vestavené LED diody a 4 piny pro napájení a kontrolu.

Obrázek 10. LED páska s 8 diodami

5.6 Schéma

Hotova schéma je na obrázku č. 11.



Obrázek 11. Schéma propojených součástek pro IoT zařízeni

Kapitola 6

Operační systém

V poslední době se objevilo mnoho služeb a zařízení IoT (internet věcí). Rychlé nasazení služeb a zařízení IoT vyžaduje efektivní a všestrannou softwarovou platformu IoT. Každé zařízení IoT je obvykle navrženo tak, aby vykonávalo určité funkce, a není programovatelné. K používání nových služeb IoT je tedy zapotřebí nové zařízení. Každý den však vychází mnoho nových služeb IoT a uživatel chce využívat více služeb. Zejména chce uživatel dokonce implementovat své vlastní aplikace. Programovatelná softwarová platforma IoT je tedy nezbytná pro distribuci služeb IoT. I když existuje několik programovatelných softwarových platforem pro mobilní zařízení, jako jsou Android, iOS a Tizen, jsou pro zařízení IoT omezená zdroji a obtížně přizpůsobitelná pro zařízení IoT. Kromě toho jsou pro implementaci aplikací vyžadovány vysoké znalosti programování.

6.1 Úvod

Rodina jednodeskových počítačů Raspberry Pi má širokou podporu operačních systémů v prostředí Linux i mimo Linux. Nicméně Raspberry Pi nemá interní úložiště. K bootování je třeba mít SD nebo mikro SD kartu na které bude nahrán obraz operačního systému.

V teto kapitole rozebereme 3 populární operační systému pro Raspberry Pi, její bootování, výhody a nevýhody, a nakonec vybereme tu ten “nejlepší”.

6.2 Raspberry Pi OS

Většina desek Raspberry Pi (s výjimkou Raspberry Pi Zero a Zero W) do roku 2019 je dodávána s Raspbian, distribucí Linuxu, která je založena na Debianu. Raspbian je oficiálně podporován nadací Raspberry Pi. Od roku 2019 s příchodem Raspberry Pi 4 se Raspbian přejmenovali na Raspberry Pi OS.

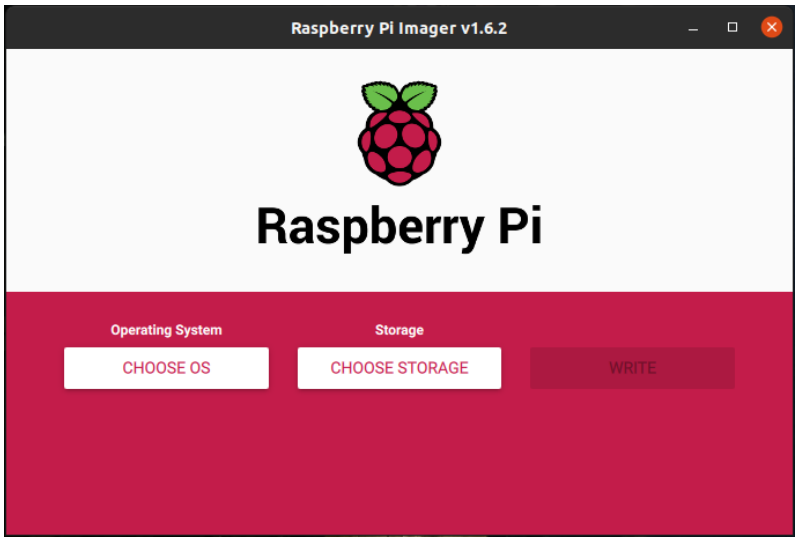
Raspberry Pi OS je k dispozici ve dvou formách: Raspberry PI OS obsahuje pracovní plochu PIXEL a Raspberry Pi OS Lite je minimální verze. Oba jsou k dispozici prostřednictvím instalačního programu NOOBS, který je součástí flash disku Raspberry Pi, který je součástí většiny desek Raspberry Pi. Raspberry Pi OS s PIXELem obsahuje několik programovacích jazyků a nástrojů, takže je dobrou volbou pro programovací prostředí připravené k použití. Obsahuje také LibreOFFICE, webový prohlížeč, e-mail a další kancelářské aplikace. Raspberry Pi OS Lite se načítá jako terminál [10].

6.2.1 Bootování

Tato část obsahuje pokyny k vytvoření bootovací Pi OS s požadovanými balíčky pro poskytování všech internetových služeb. Měla by být použita alespoň 4 gigabitová karta microSD. Aktuální verze Raspberry Pi OS automaticky rozšiřují souborový systém tak, aby používal plnou kartu microSD, což znamená, že bootovací obraz bude tak velký, jak se na kartu vejde. To znamená, že přenosy budou pomalejší a bootovací zařízení zabere obrovské množství místa pro zálohování.

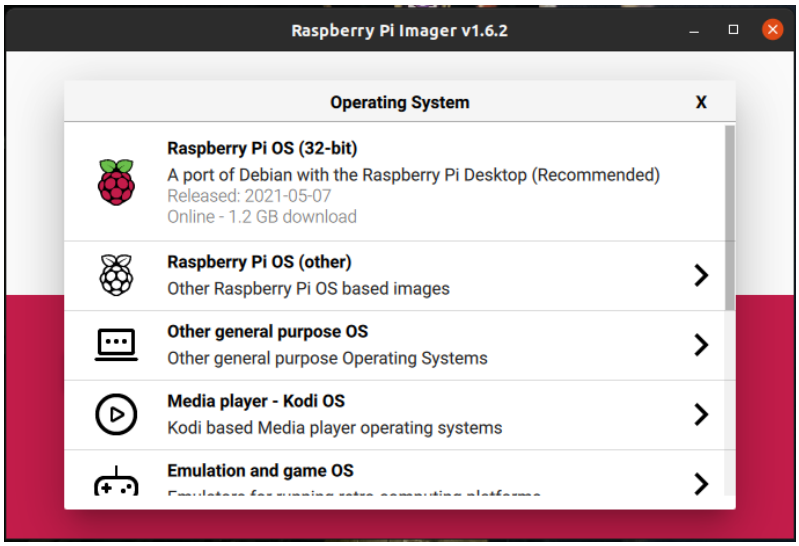
Podle oficiálního návodu od výrobce Raspberry Pi OS instalace je snadná a rychlá. V roce 2020 byl představen nastroj Raspberry Pi Imager. Tento nástroj je snadno použitelný a super rychlý, díky některým zkratkám, které zavedli do algoritmu. Nastroj je možné stáhnout na oficiálních stránkách výrobce.

Po spuštění, za prvé, Raspberry Pi Imager stáhne soubor a.JSON ze svých webových stránek se seznamem všech aktuálních možností stahování, čímž zajišťuje, že vždy jsou dostupné nejaktuálnější verzi.



Obrázek 12. Pi Imager - úvodní obrazovka

Celý proces se skládá z volby operačního systému, výběru karty a spuštění zápisu.



Obrázek 13. Pi Imager - výběr operačních systému

V nabídce Imageru nejsou jen tři základní verze oficiálního Raspberry Pi OS: textový Lite, základní Desktop a Desktop Full s hromadou předinstalovaných aplikací, ale také linuxová distribuce LibreELEC s domácím kinem Kodi, pokud je třeba Raspberry proměnit v multimediální přehrávač připojený skrze HDMI k televizoru, Ubuntu, Manjaro, RISC OS Pi a další. Utilita si poradí i s několika dalšími úkony – umí na microSD nahrát třeba základní recovery firmware v případě, že došlo k poškození systémové EEPROM, chybí ale například konfigurace systému. Hodilo by se například, kdyby aplikace umožnila rovnou povolit SSH a přednastavit Wifi – tedy

tzv. headless konfiguraci pro případy, kdy poběží Raspberry jako server nebo robotická řídící jednotka bez běžného grafického výstupu.

Jakmile z dostupných možností bude vybrán operační systém, nástroj načte příslušný soubor přímo z webových stránek Raspberry Pi a zapíše jej přímo na kartu SD. To ve srovnání se standardním procesem čtení z webu, zápisu do souboru na vašem pevném disku a poté, jako samostatný krok, jeho čtení zpět z pevného disku a zápisu na SD, značně zrychluje tento proces. Tento proces se skládá z dvou částí:

1. Zapisování obrazu na SD kartu

Tento krok může trvat od několika minut do několika hodin a záleží to na velikosti a rychlosti SD karty na kterou se to zapisuje. Například obraz Raspberry Pi OS lite má velikost 0.4 GB a SD karta má třídu 10(10 MB/s), tak proces bude trvat přibližně 1 minutu.

2. Ověřování obrazu na SD kartě

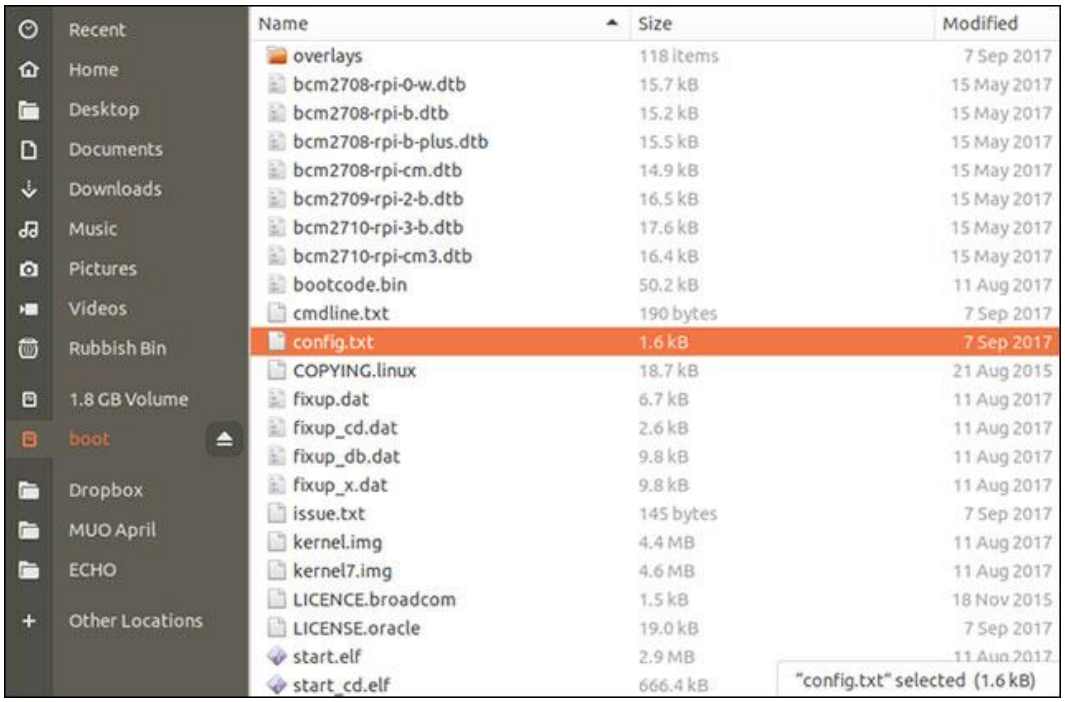
Druhým krokem nastroj Pi Imager musí ověřit správnost a integritu zapsaného obrazu. Ověřování trvá minimálně 2 minuty.

Jakmile obraz je zapsán a ověřen Pi Imager ukáže okno jako na obrázku č. 14:

Obrázek 14. Pi Imager - úspěšný zápis

6.2.2 Konfigurace

Jako většina moderních operačních systémů je Raspberry PI OS ovládán velkým počtem konfiguračních souborů. Při spuštění Raspberry PI OS tyto soubory čte během inicializace v různých časech. Při čtení souborů Raspberry PI OS nastavuje interní příznaky a někdy dokonce vytváří nové konfigurační soubory, které odrážejí požadované chování operačního systému, a tedy i Raspberry Pi. Konfigurační soubory je nutné mnohokrát ručně upravit, aby bylo možné vybrat různé volitelné akce, které má OS provést. To lze očekávat, protože Raspberry PI OS je založen na distribuci Linuxu s názvem Debian, která je navržena tak, aby byla konfigurována buď ručně, nebo prostřednictvím grafického uživatelského rozhraní [10].



Obrázek 15. Umístění konfiguračního souboru config.txt

Hardwarová nastavení počítače Pi závisí na hodnotách v souboru config.txt, který je umístěn v adresáři /boot (viz obrázek č. 15). Na základě tohoto souboru počítač Pi nastavuje různé vstupy a výstupy a taktování čipu BCM2xxx a připojeného paměťového modulu.\

Jsou-li problémy s grafickým výstupem (pokud například obraz nevyplňuje celou obrazovku nebo přesahuje její okraj), je možné to vyřešit změnou parametrů v souboru config.txt. Soubor je obvykle prázdný nebo v některých distribucích vůbec neexistuje. To pouze znamená, že počítač Pi funguje s předem nastavenými výchozími hodnotami. Pokud je třeba provést změny a soubor neexistuje, stačí vytvořit nový textový soubor s názvem config.txt a vyplnit potřebná nastavení.

Soubor config.txt kontroluje téměř všechny aspekty hardwaru počítače Pi. Soubor se načítá pouze při prvním spuštění systému. Případné změny provedené během činnosti počítače Pi se projeví teprve po jeho restartu nebo vypnutí a opětovném zapnutí. Pokud některé změny způsobují problémy, stačí k obnovení výchozích hodnot pouze odstranit soubor z adresáře /boot.

6.3 Ubuntu

Čerstvě vydané Ubuntu 20.10 Groovy Gorilla poprvé přináší podporu plnohodnotného desktopu na Raspberry Pi 4. Ubuntu už v minulých vydáních podporovalo Raspberry Pi, zatím však ve variantách bez desktopového prostředí jako Server nebo Cloud. Současné vydání však uživatelům Raspberry nabízí také verzi Desktop včetně prostředí GNOME a obvyklé řady

desktopových aplikací. Podle tvůrců je to první krok k většímu cíli: desktopovému vydání s dlouhodobou podporou (LTS) [18].

6.3.1 Bootování

Pro instalaci systému jsou dvě možnosti. Společnost Canonical doporučuje používat Imager, oficiální aplikaci Raspberry Pi, odkud je možné stáhnout obrázek a zapsat jej na SD. Způsob je stejný jako při instalaci klasického Raspberry Pi OS, který byl představen v kapitole 6.2.1

6.4 Windows

S Windows 10 přicházející na platformy ARM, jako jsou Qualcomm Snapdragon a Samsung Exynos, víme, že je možné spustit Windows 10 s desktopovými aplikacemi na zařízeních ARM, včetně Raspberry Pi 4. Existuje několik způsobů, jak nainstalovat a spustit Windows 10 na Raspberry Pi. Microsoft vždy povolil uživatelům spouštění „Windows 10 IoT “, což je také verze Windows 10, ale je to určeno pouze pro prototypování, nikoli pro provozování skutečných desktopů. Společnost Microsoft oficiálně neumožňuje spustit skutečnou verzi Windows 10 na tomto počítači s jednou deskou, ale existují i některé technické projekty, které umožní nainstalovat operační systém na Raspberry Pi 4 a zvýšit produktivitu [19].

Microsoft představil dvě edice Windows 10 IoT [20]:

1. Windows 10 IoT Core
2. Windows 10 IoT Enterprise

Tabulka 3. Porovnaní Windows 10 IoT Core a Enterprise

Windows 10 IoT Core	Windows 10 IoT Enterprise
střižená verze Windows 10	Plná verze Windows 10 sdílí všechny výhody celosvětového ekosystému Windows
Optimalizováno pro menší zařízení	Optimalizováno pro komplexní řešení
Dostává méně aktualizací	Získá více aktualizací než IoT Core
Podporuje jednu aplikaci, jednu aplikaci v popředí najednou s podporou aplikace na pozadí	Podporuje jednu aplikaci, jednu aplikaci v popředí najednou s podporou aplikace na pozadí
Podporováno pouze UWP UI	Plná podpora Windows UI

6.4.1 Bootování

Na rozdíl od jiných systémů vyžaduje Windows 10 IoT instalaci Windows 10 pro přístup, programování a stahování. Instalace Windows 10 IoT Core probíhá pomocí nástroje Windows 10 IoT Core Dashboard [21].

Microsoft se snaží instalaci Windows 10 IoT zjednodušit pomocí IoT Core Dashboard. Tento řídicí panel vám umožňuje vytvořit přizpůsobený systém Windows 10 IoT zaměřením na určené procesory, ale zda to ve skutečnosti věci zjednoduší nebo ne, závisí na uživateli [22].

6.5 Alternativní

Na Raspberry Pi může běžet také mnoho dalších operačních systémů. Podporován je také formálně ověřené mikrojádro seL4.

Seznam možných k instalaci operačních systému na Raspberry Pi:

- Broadcom VCOS – Proprietární operační systém, který obsahuje vrstvu abstrakce navrženou pro integraci se stávajícími jádry, jako je například ThreadX (který se používá na procesoru VideoCore4), poskytující ovladače a middleware pro vývoj aplikací. V případě Raspberry Pi to zahrnuje aplikaci pro spuštění procesorů ARM a poskytování veřejně dokumentovaného API prostřednictvím rozhraní poštovní schránky, které slouží jako jeho firmware. Neúplný zdroj linuxového portu VCOS je k dispozici jako součást referenčního grafického ovladače vydaného společností Broadcom [23].
- HelenOS - přenosný multiserverový operační systém na bázi mikrojádra; má základní podporu Raspberry Pi od verze 0.6.0 [24].
- Android Things - integrovaná verze operačního systému Android určená pro vývoj zařízení IoT.
- Tiny Core Linux - minimální operační systém Linux zaměřený na poskytování základního systému pomocí BusyBox a FLTK. Navrženo pro běh primárně v RAM.

6.6 Závěr

Raspberry Pi OS je založen na Debianu GNU/Linux. Nejnovější verze systému Raspberry Pi OS (v době psaní tohoto článku) vychází z Debianu 10 Buster (nejnovější stabilní verze Debianu). Raspberry Pi OS je velmi lehký pro jednodeskové počítače Raspberry Pi. Má podporu pouze pro 32bitové operační systémy, zatímco existuje Beta verze, která přináší 64bitovy operační systém. 32bitové operační systémy podporují maximálně 3 GB RAM.

Vzhledem k tomu, že Raspberry Pi OS je postaven na Debian, stejně jako Ubuntu, a může fungovat bez omezení a oficiálně podporován výrobcem, je favoritem pro IoT zařízení.

Kapitola 7

JavaScript

Tato kapitola poskytuje obecný přehled JavaScriptu; historie a vývoj JavaScriptu, následovaný popisem aktuálních funkcí JavaScriptu. Popisuje stav a historii verzí ECMAScriptu, který je implementován pomocí JavaScriptu, a poté představuje různé varianty JavaScriptu. Je důležité pochopit aktuální strukturu a specifika JavaScriptu, protože všechny diskutované rámce jsou postaveny v základním jazyce JavaScript. Předpokládá se, že čtenář již zná, co je HTML, CSS a jak funguje web [25].

7.1 Historie

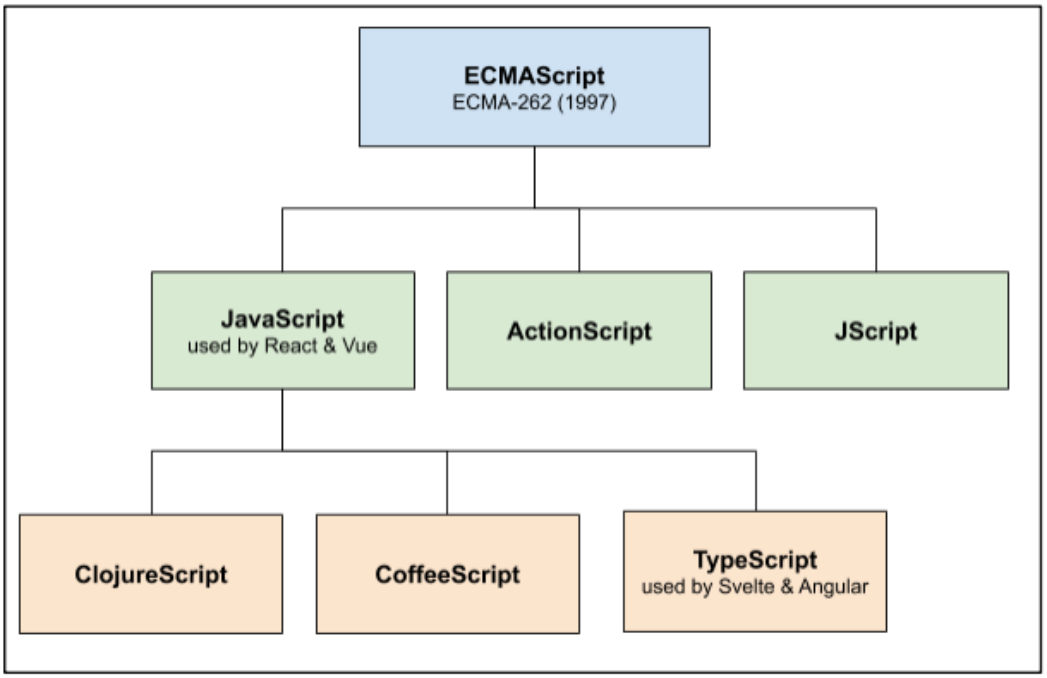
V roce 1995 společnost NetScape Communications, vývojáři tehdy populárního webového prohlížeče NetScape Navigator, najala programátora Brendana Elcha, aby vytvořil nový dynamický skriptovací jazyk pro webové stránky a manipulaci s daty na straně klienta. Poté, co se Netscape etabloval na trhu webových prohlížečů, viděl potřebu vytvářet dynamické webové stránky namísto použití pouze HTML, což byl do té doby standard. Většina webových stránek byla navržena výpočetně neefektivním způsobem pomocí pouze HTML: pro každou akci uživatele nebo kliknutí byl odeslán požadavek na server a poté byla klientovi odeslána nová stránka HTML.

Marc Andreessen, zakladatel NetScape Communications, věřil, že existuje základní potřeba jednoduchého skriptovacího jazyka pro web, zaměřeného na manipulaci s DOM. Skriptovací jazyk byl určen nejen pro zkušené vývojáře, ale také pro designéry a lidi s menšími zkušenostmi s programováním, něco, co by fungovalo jako doplněk HTML. Projekt byl inspirován funkčností a syntaxí Javy, i když se zásadně liší. Brendan Elch a jeho tým zpočátku nazývali projekt Mocha, později přešli na LiveScript, než nakonec přistoupili ke jménu JavaScript, což způsobilo mnoho zmatků, pokud jde o podobnost mezi Javou a JavaScriptem [26].

Projekt byl prototypován v průběhu roku 1995 a oficiálně vydán v březnu 1996. Už při spuštění JavaScript umožňoval nové funkce na webových stránkách, které samotný HTML nezvládal, jako například reakce na vstup uživatele, změna barvy prvků a zobrazování vyskakovacích oken [27]. Jedním z důvodů jeho pozdější popularity bylo, že NetScape koupila společnost America Online (AOL) a později předala kód svého prohlížeče společnosti Mozilla, včetně funkcí založených na JavaScriptu, což přispělo k jejímu růstu [28].

7.2 ECMAScript

Po vydání JavaScriptu v roce 1997 vývojáři v čele s Brendanem Elchem viděli potřebu jazykové standardizace, která by podpořila růst, zabránila fragmentaci komunity vývojářů JavaScriptu a zpřístupnila jazyk napříč prohlížeči. Stalo se tak prostřednictvím jazykového standardu ECMAScript, definovaného v roce 1997 normalizační organizací Ecma International. ECMAScript má standardní ID ECMA-262.



Obrázek 16. Vztahy mezi verzemi JavaScript

Obrázek č. 16 popisuje vztah mezi standardem ECMAScript, různými implementacemi nebo „dialekty“ ECMAScriptu (JavaScript, ActionScript a JScript) a „příchutě“ JavaScript: ClojureScript, CoffeeScript a TypeScript. JavaScript je nejznámější implementací ECMAScriptu, ale kromě JavaScriptu existuje několik dalších implementací nebo „dialektů“ ECMAScriptu. Jedním z nich je JScript, vyvinutý společností Microsoft v roce 1996 jako jejich vlastní interní alternativa k JavaScriptu, používaná především v prohlížeči Microsoft Internet Explorer. Dalším pozoruhodným dialektem je ActionScript, vyvinutý společností Macromedia Inc., společností později koupenou společností Adobe Systems. Standard ECMAScript se neustále vyvíjí od své první standardizace (verze 1) v roce 1997. ECMAScript 6, známý také pod názvem ECMAScript 2015, je šestým vydáním normy ECMA-262 a je často používanou verzí. Od verze ECMAScript 2015 se Ecma přesunula na systém každoročního vydání, který aktualizuje ECMAScript jednou za rok, takže každá verze je poté známá jak podle čísla verze, tak podle roku verze. Nejnovější, dvanácte vydání ECMAScript bylo definováno v červnu 2021 jako ECMAScript 2021 [29].

Když je třeba zajistit kompatibilitu se staršími prohlížeči, musí být modernější verze JavaScriptu převedeny na starší verze. Konverzi lze také provádět mezi verzemi JavaScriptu nebo z jedné „příchuti“ JavaScriptu na jinou, například z TypeScriptu na JavaScript. Tento proces se nazývá transkompilace. Nejvíce podporovaná verze JavaScriptu odpovídá ECMAScript 2015 a toto je běžně používaná cílová verze pro účely překompilování. Různé varianty JavaScriptu, označené na obrázku č. 16 oranžovou barvou, se někdy nazývají překompilované jazyky.

7.3 XML a AJAX

XML je značkovací jazyk podobný HTML, ale častěji se používá k reprezentaci dat místo zobrazení obsahu. XML i HTML jsou dnes na internetu široce používány a oba také pocházejí z dřívějšího značkovacího jazyka SGML (Standard Generalized Markup Language), který byl v 80. letech používán jako dynamický informační jazyk. XML bylo vyvinuto World Wide Web Consortium a původně vydáno v roce 1996, zatímco poslední standardní vydání, páté, bylo definováno v roce 2008. Vývojáři XML zamýšleli, aby byl jazyk použitelný přes internet, snadno se psal a četl, kódování dokumenty ve formátu čitelném jak pro lidi, tak pro počítače. Jak je vidět na obrázku č. 17, XML je podobný HTML, protože oba používají k definování prvků otevírací a zavírací značky a mezi tagy lze definovat obsah nebo text. Další podobností s HTML je vnoření vlastností, protože **city** a **country** jsou vnořené vlastnosti **location** na obrázku č. 17. Tento příklad kódu představuje osobu s vlastnostmi **firstName**, **lastName** a **location** a vlastnosti **city** and **country** jsou vnořené vlastnosti **location** [30].

XML se mělo stát relevantní v oblasti JavaScriptu s vynálezem technologické kolekce AJAX,

```
<person>
        <firstName>Mattias</firstName>
        <lastName>Levlin</lastName>
        <location>
                <city>Espoo></city>
                <country>Finland></country>
        </location>
</person>
```

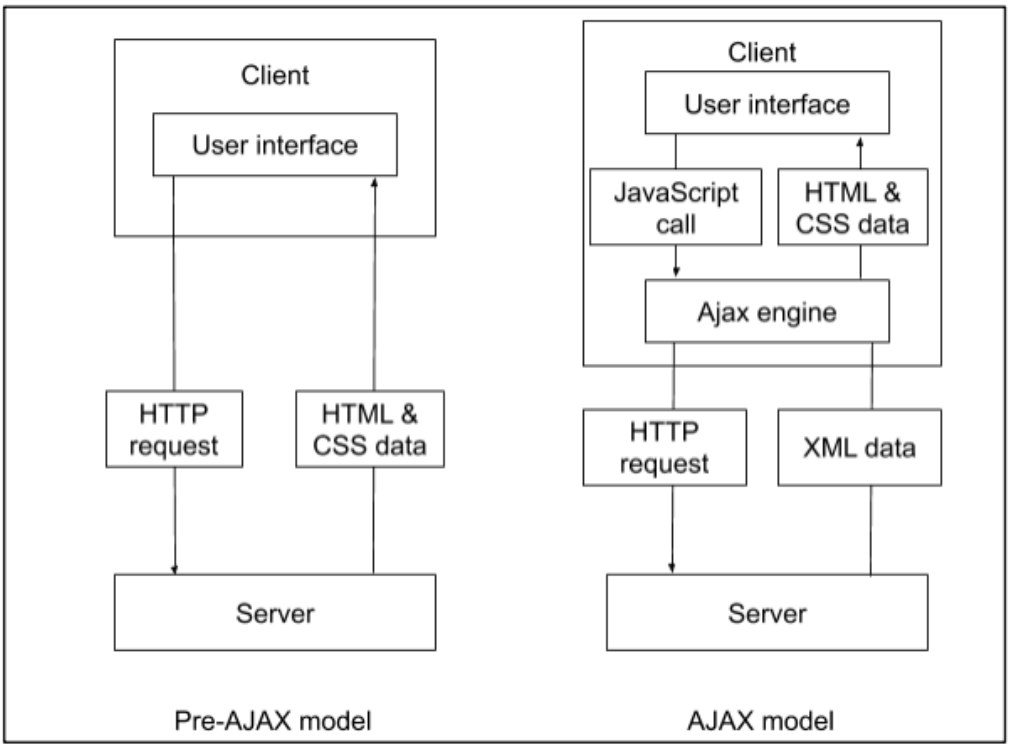
Obrázek 17.Ukázka kódu XML

což je zkratka pro asynchronní JavaScript a XML. AJAX poskytuje vývojářům způsob, jak aktualizovat části stránky HTML bez stahování celého jejího obsahu. AJAX není jedinou technologií nebo nástrojem, ale spíše souborem několika technologií spojených dohromady jako celek; Garrett definuje tyto technologie jako XHTML, CSS, DOM, XML, XLST (eXtensible Stylesheet Language Transformations), XMLHttpRequest a JavaScript [31].

K prvnímu kroku směrem k AJAX a vzniku asynchronních prvků na webu došlo zavedením značky HTML iframe (vložený rámec), zavedené v roce 1996 v aplikaci Microsoft Internet Explorer. Další důležitou technologií, která je ústředním prvkem technologického zásobníku AJAX, je XMLHttpRequest, vyvinutý v roce 1998. XMLHttpRequest je skriptovací objekt, který se používá k odesílání dat XML na server a ze serveru místo HTTP dat. Systém AJAX byl

zprototypován v roce 1999. Používání webových aplikací před AJAXem obvykle vyžadoval dlouhé čekání na konci uživatele: pokaždé, když byl požadavek uživatele odeslán kliknutím na rozhraní, musel uživatel nevyhnutelně čekat na odpověď synchronního serveru a přenos dat. V systému AJAX je každá uživatelská akce, která by normálně vyžadovala požadavek serveru, směrována místo toho na modul AJAX, který je umístěn na straně klienta. Některé jednoduché akce lze zpracovat výhradně na straně klienta a u věcí, které vyžadují komunikaci se serverem, se místo synchronních dat HTTP používají asynchronní data XML. Webová aplikace tedy funguje bezproblémově z pohledu uživatele a eliminuje čekací doby. Na obrázku č. 18 jsou klíčovými inovacemi systému AJAX volání JavaScriptu a modul AJAX, oba umístěné na straně klienta, a komunikace prostřednictvím dat XML, nahrazující data HTML/CSS ze serveru.

Všechny frameworky JavaScriptu uvedené v této práci přijaly jednostránkové aplikační principy, filozofii designu, která se stala dobře známým standardem mezi programátory



Obrázek 18. Porovnaní technologii AJAX a přímého přenosu dat HTTP

webových aplikací. Jednostránková aplikace (SPA) je aplikace složená z jednotlivých komponent, načtená do paměti při návštěvě první stránky, kterou lze poté vyměnit nebo aktualizovat nezávisle, takže se při každé akci uživatele nemusí znovu načítat celá webová stránka. Další výhodou jednostránkových aplikací je, že komponenty lze znovu použít, a tím lze výrazně snížit množství potřebného kódu. Jednostránková aplikace byla poprvé implementována a patentována v roce 2002, přičemž patent konkrétně uváděl JavaScript jako příklad cílového jazyka pro implementaci. Jednostránkové aplikace lze porovnávat s alternativní vícestránkovou

aplikací, která může mít některé okrajové výhody, včetně snazší optimalizace pro vyhledávače, protože vyhledávače považují každou stránku vícestránkové aplikace za samostatnou stránku. Úspěch jednostránkových aplikací, které budou později propagovány, byl z velké části díky předchozí technologii AJAX a jejím inovacím v komunikaci se serverem [32].

Kapitola 8

Backend

Webové aplikace, jako softwarové systémy, se tradičně dělí na dvě části: Backend (server), která je v mnoha případech zodpovědná za zpracování dat, a Frontend (klient), která je nezbytná k zajištění pohodlného rozhraní pro interakci mezi uživatelem a systémem.

Backend lze navrhnout pomocí různých architektonických přístupů, jedním z nich jsou mikroslužby, které jsou nejvhodnější pro škálovatelné systémy. Myšlenka přístupu vychází z konceptu rozdělení logicky nezávislých částí systému. Používá se místo monolitických architektonických stylů, protože je obtížné je škálovat (a není možné škálovat pouze jejich jednotlivé části) a současně vyvíjet jejich různé části, přičemž tyto problémy jsou řešeny v mikroslužbách.

8.1 GIT

Správa verzí je systém, který zaznamenává změny souboru nebo sady souborů v čase tak, abyste se mohli později k určité verzi vrátit. Umožní vrátit soubory zpět do předchozího stavu, vrátit celý projekt do předchozího stavu, porovnávat změny provedené v průběhu času, zjistit, kdo naposledy upravil něco, co nyní možná způsobuje problémy, kdo a kdy vytvořil diskutabilní část a mnoho dalšího.

Git se narodil, stejně jako mnoho velkých věcí v životě, z kreativní destrukce a vášnivého sporu.

Jádro Linuxu je celkem rozsáhlý softwarový open source projekt. Po většinu životního cyklu údržby jádra Linuxu (1991–2002) se změny přenášely formou záplat a archivních souborů. V roce 2002 začal projekt linuxového jádra využívat komerční distribuovaný systém správy verzí s názvem BitKeeper [33].

Vztahy mezi komunitou, která vyvíjela jádro Linuxu, a komerční společností, která vyvinula BitKeeper, se v roce 2005 zhoršily a nástroj přestal být poskytován zdarma. To přimělo komunitu vývojářů Linuxu (a zejména Linuse Torvaldse, tvůrce Linuxu), aby vyvinula vlastní nástroj, založený na poznatcích, které nasbírala při užívání systému BitKeeper. Některé z cílů nového systému byly následující:

- Rychlost,
- jednoduchý návrh,
- silná podpora nelineárního vývoje (tisíce paralelních větví),
- plně distribuovaný,
- schopnost efektivně spravovat velké projekty, jako je linuxové jádro (rychlost a objem dat).

Od svého vzniku v roce 2005 se Git vyvinul a vyzrál v snadno použitelný systém, který si dodnes uchovává své prvotní kvality. Je extrémně rychlý, velmi efektivně pracuje s velkými projekty a nabízí skvělý systém větvení pro nelineární způsob vývoje.

8.2 Node.JS

Při rostoucí popularitě JavaScriptu, a stále větším možnostem jeho použití, ale stále existovala při webovém vývoji jedna oblast, pro kterou bylo nutné využít jiný jazyk, ačkoliv ji museli vývojáři na pozici full-stack developerů obsáhnout. Jedna se sice o oblast backendu, tedy serverové části aplikace. K tomuto se využívalo různých jazyků, ovšem po čase se zrodila myšlenka na využití JavaScriptu i na straně serveru. Vývojář tak v podstatě musí umět hlavně jeden jazyk pro psáni kompletní webové aplikace. Node.js je postaven na V8 jádře, které používá Google Chrome a jedná se o open source prostředí pro server. Umožňuje tedy využití JavaScriptu na straně serveru a je prakticky nezávislý na vybrané platformě (Windows, Linux, Unix, macOS a další). Jeho hlavní výhodou, tedy kromě možnosti využití stejného jazyku jako na frontendu, je

asynchronní programování a velmi efektivní práce s pamětí. Právě díky jeho rychlosti a snadnosti použití se jedná o jedno z nejoblíbenějších rozšíření vůbec [34].

Vlastnosti Node.JS platformy, jako asynchronní přístup ke vstupně-výstupním operacím a její snadná škálovatelnost, jsou velice zajímavé z hlediska jejího výběru pro vývoj aplikací na straně serveru. Platforma byla zveřejněna v roce 2009 a od té doby zažívá vzestup. Platforma je podporována a využívána významnými firmami, jako jsou Microsoft, eBay, LinkedIN, Yahoo a další [35].

8.2.1 Express.JS

Express.js je minimalistický, flexibilní a open source webový framework pro Node.js. Díky němu je vývoj webových aplikací a API mnohem jednodušší než při použití samotného Node.js. Především vyřizování požadavků a routování lze s Express.js spravovat mnohem efektivněji. Express.js je v dnešní době možné označit jako standard pro vývoj webových aplikací používajících na backendu Node.js [36].

8.3 Firebase

Firebase je kombinací mnoha služeb Google v cloudu, včetně ověřování uživatelů, databáze, úložiště, hostování atd. Cloudový systém Firebase poskytuje přenos dat šifrování SSL.

Jednou z hlavních funkcí internetu věcí je sběr dat. Firebase poskytuje přístup ke své databázi v iOS, Android, JavaScript SDK, REST API, Admin SDK. iOS a Android SDK jsou pro vývoj mobilních aplikací. JavaScript SDK je pro webové aplikace. A REST API je obecná síťová služba, kterou lze použít pro obecný programovací jazyk. Admin SDK se používá v prostředí Node k implementaci aplikací JavaScript. Všechna tato připojení jsou šifrována.

Důvodem, proč byl vybrán Google Cloud Firebase je, že má službu Authentication, která zároveň umožní pracovat s Google Calendar API out-of-box. To byl hlavní důvod, proč jsem ani neuvažoval o jiných službách od dalších dodavatelích, jako např. Amazon (AWS) nebo Microsoft (Azure).

8.3.1 Firestore

Tato služba zastupuje databázi. Firestore je NoSQL databáze, která je velice podobná databázi MongoDB. Na rozdíl od SQL jde tedy o dokumentovou databázi. Její základní strukturou jsou kolekce. V těchto kolekcích jsou dokumenty a v těchto dokumentech jsou naše finální data, která potřebujeme uložit. To je základní struktura NoSQL databází. Samozřejmostí je vnořování kolekcí a vytváření tak složitějších databázových struktur. Služba umožňuje správu dat pomocí několika programovacích jazyků. Mezi nimi jsou např. C++, Unity, Node.js, Java, Python nebo Go [37].

8.3.1.1 Snapshot

Místní zápisy v aplikaci okamžitě vyvolají posluchače snímků. Důvodem je důležitá funkce, která se nazývá “latency compensation”. Když se provede zápis, budou posluchači upozorněni na nová data před odesláním dat do backendu [38].

8.3.1.2 Implementace

Pro splnění serverových požadavku, podle datového modelu byly navrženy následující kolekce:

- **Alarms**

Tato kolekce obsahuje seznam objektu, které reprezentuji objekt Alarm. Má pole: label, time, status, repeat, sound.

- **Events**

Kolekce Events obsahuje seznam Event objektu. Každý objekt ma stejné vlastnosti, jako stejnojmenný objekt z datového modelu.

- **Next**

Kolekce Next obsahuje seznam objektu, které reprezentuji notifikace, buďto pro alarm nebo event. Tato kolekce se plni ve třech případech:

1. Uživatel založí novy event nebo alarm

V tomto případě backend má na starosti vytvořit seznam nadcházejících upozorněni pro tento typ události. Pokud jde o alarm, tak kolekce se naplní o N záznamu, kde N je počet opakovaní alarmu v týdnu. Jde-li o Event, tak se vytvoří notifikace pro každé připomenuti.

2. Uživatel upraví již existující alarm nebo event.

Při takovém scénáři, backend musí smazat již založené události a přezaložit je tak, jak to bylo popsáno v bodě 1.

3. Uživatel smazal event nebo alarm.

Backend musí odstranit všechny notifikace o smazané události.

- **Settings**

Pomocná kolekce pro udržovaní key-value záznamu o různých nastavení: Například: hlasitost zvuku, jas LED pásky a další.

8.3.2 Authentication

Služba sloužící ke správě uživatelských účtů a jejich identit. Součástí je zakládání účtů, ověřování uživatelů, obnova hesla a další akce spojené s uživatelským účtem. Firebase Authentication umožňuje i přihlašování pomocí služeb třetích stran. Těmi jsou např. Google, Facebook, Twitter, Github, Yahoo, Microsoft či Apple. V mém případě však využívám pouze možnost ověření pomocí Google pro následnou synchronizaci událostí s Google Calendar.

8.3.3 Functions

Cloud Functions for Firebase je bez serverový framework, který umožňuje, aby se kód na straně serveru automaticky spustil v reakci na události spouštěné funkcemi Firebase a požadavky HTTPS. Kód JavaScript nebo TypeScript je uložen v cloudu Google a běží v kontrolovaném prostředí. Není třeba spravovat a škálovat vlastní servery [39].

8.3.4 Emulatory

Firebase Local Emulator Suite je sada pokročilých nástrojů pro vývojáře, kteří chtějí vytvářet a testovat aplikace lokálně pomocí Cloud Firestore, Realtime Database, Cloud Storage, Authentication, Cloud Functions, Pub/Sub a Firebase Hosting. Poskytuje bohaté uživatelské rozhraní, které vám pomůže rychle spustit a prototypovát [40].

Místní vývoj s Local Emulator Suite může být vhodný pro prototypování, vývoj a nepřetržitou integraci.

Obrázek 19. Přehled existujících Google Firebase Emulátoru

Sada Firebase Local Emulator Suite se skládá z jednotlivých emulátorů služeb vytvořených tak, aby přesně napodobovaly chování služeb Firebase. To znamená, že je možné aplikaci připojit přímo k těmto emulátorům a provádět integrační testování nebo QA bez dotyku produkčních dat.

Je možné například připojit aplikaci k emulátoru Cloud Firestore a bezpečně číst a zapisovat dokumenty při testování. Tyto zápisy mohou spouštět funkce v emulátoru Cloud Functions.

Kapitola 9

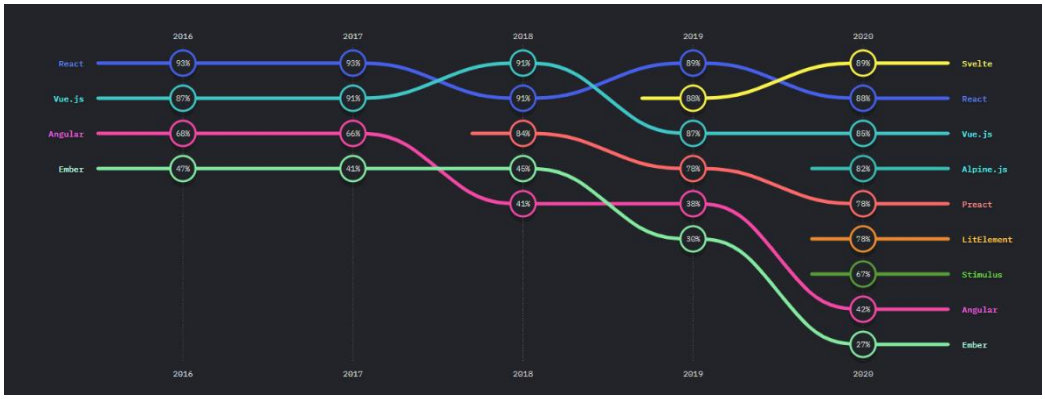
Frontend

V této kapitole se zaměříme na používání JavaScriptu k programování zařízení IoT na platformě Raspberry Pi 4. Přestože se JavaScript již dlouho těší velké oblibě při psaní webu, nyní je sám o sobě vyspělým a bohatým jazykem. Je také stále více převládající ve světě IoT díky své přenositelnosti na širokou škálu zařízení a také díky velké instalované základně knihoven a vývojářů, kteří znají jazyk. Asynchronní povaha jazyka z něj navíc dělá hlavního kandidáta pro aplikace IoT, které jsou často řízený událostmi, a proto musí být vysoce citlivé.

9.1 JavaScript frameworky

Po přezkoumání JavaScriptu jako celku je v této kapitole zhodnocena celková funkce a architektura obecného rozhraní JavaScript pro front-end. React, Vue, a Angular odpovídají charakteristikám definovaným v této kapitole, a to buď prostřednictvím svých základních funkcí, nebo prostřednictvím běžně používaných rozšiřujících knihoven. Některé z nejběžnějších funkcí rámců front-end jsou synchronizace stavu a zobrazení, směrování, systém šablon a opakovaně použitelné komponenty [41].

Protože existuje velký počet JavaScript frameworku pro vývoj frontendu, je důležité definovat některá kritéria výběru. Spolehlivá a kvantitativní data průzkumů se nacházejí zejména v průzkumech The State of JavaScript, publikovaných každoročně od roku 2016 [42].



Obrázek 20. Porovnaní populárností JavaScript frameworku v rocích 2016 až 2020

9.1.1 React

React.js nebo jednoduše React je knihovna JavaScript vyvinutá společností Facebook. Byl popsán jako deklarativní, účinný a flexibilní framework. První verze React byla vydána v květnu 2013. React má užší rozsah než ostatní frameworky v tomto seznamu a zobrazuje pouze uživatelské rozhraní aplikace. Výhodou je lehká struktura knihovny, jejíž učení a používání je levnější. To však také znamenalo, že React byl v určitých kontextech nazýván knihovnou uživatelského rozhraní, nikoli frameworkem. Obecně jej však lze považovat za framework, protože se používá ke stejným účelům jako Vue a Angular 2+ [43]. React byl původně vyvinut jako port JavaScript pro XHP, knihovnu PHP vytvořenou Facebookem. XHP byla modifikace PHP, která vám umožnila vytvořit si vlastní komponenty, čehož je React také schopen. Tento vývoj lze považovat za důležitý krok v celkovém posunu ve vývoji webu, přičemž jako základní webovou technologii byl zvolen JavaScript, nikoli PHP, což byl v roce 2000 dominantní standard. XHP byla knihovna, která byla zaměřena na prevenci škodlivých útoků uživatelů se zlými úmysly, a projekt portování JavaScriptu se rozrostl do JSX (JavaScript XML), který se stal běžným standardním jazykem pro React spolu se standardním JavaScriptem [44].

9.1.2 Angular

Angular je framework, který existuje ve dvou verzích, běžně nazývaných AngularJS a Angular 2+. AngularJS je starší verze založená na JavaScriptu, která již není v aktivním vývoji, zatímco Angular 2+ je novější a vychází z TypeScriptu. Angular 2+ byl vydán v roce 2016 a liší se od svého předchůdce AngularJS a většiny ostatních rámců v tom, že je založen na TypeScriptu. Je možné použít Angular bez TypeScriptu, ale tato volba byla nazývána obtížná a obecně se nedoporučuje [45]. Angular 2+ je v současné době populárnější verzí Angular s vylepšeními výkonu a dalšími výhodami oproti AngularJS. Angular, navržený pro vývoj větších aplikací, je jedním z největších a plně funkčních frameworku JavaScriptu, a to jak z hlediska funkcí programování, tak z hlediska velikosti souboru. V recenzi State of JavaScript 2018 byly nejčastěji citovanými pozitivními aspekty Angular počet funkcí, styl programování a dokumentace. Nejčastěji uváděnými negativními aspekty Angular byly jeho zdánlivé nafouknutí, složitost a těžký vývojový styl, který se nedoporučuje pro malé developerské projekty. Bylo také zaznamenáno, že má docela strmou křivku učení [46].

9.1.3 Vue

Vue.js, nebo jen Vue, vytvořil bývaly zaměstnanec Google - Evan Yu, který se inspiroval AngularJS, ale chtěl vytvořit jeho jednodušší a vylepšenou verzi. Vue lze tedy považovat za odlehčenou verzi AngularJS. Hlavní knihovna Vue je zaměřena pouze na prezentační vrstvu. První verze Vue byla vydána v roce 2014. Od té doby se Vue rozrostl a vedle React a Angular se stal jedním ze tří nejpopulárnějších JavaScript frameworku. V recenzi State of JavaScript 2020 byly nejčastěji citovanými pozitivními aspekty Vue to, že se snadno učí, je lehký, má příjemný styl kódování, dokumentaci a vysoký výkon. Celkově nejběžnějším negativním aspektem byla jeho neobratnost. Vývojáři Vue vydali svůj framework v době, kdy v JavaScriptu dominovaly React a první verze Angular. Z tohoto důvodu se vývojáři Vue rozhodli přidat stránku porovnávající jejich vlastní rámec s ostatními. Článek pojednává především o rozdílech a podobnostech s Reactem, ale také poskytuje srovnání s oběma verzemi Angular, stejně jako s jinými frameworky. Tato dokumentace se zaměřuje na podobnosti mezi React a Vue; autoři dokumentace poznamenávají, že oba používají virtuální DOM a poskytují komponenty reaktivního a složeného pohledu. Další podobností je, že jádro obou frameworku je poměrně úzké, aby se udrželo zaměření a umožnilo uživatelům využít modularitu frameworku. Prvky, které mohly být zahrnuty do základního frameworku, jako je směrování a správa globálního stavu, jsou místo toho zpracovávány doprovodnými knihovnami (oblíbené alternativy jsou Vue-router a VueX). Vue podporuje všechny prohlížeče kompatibilní s ECMAScript 5 [45].

9.1.4 Výběr technologie

V této prací bude používán framework Vue z řadu příčin:

1. Je lehký pro začátečníky a na učeni
2. Je reaktivnější než Vue nebo Angular.

Vue narozdíl od Reactu a Angularu ma obousměrnou reaktivnost, což znamená, že se při každé změně dat provede přerenderování šablony a naopak.

Já, jako autor, již mam zkušenost s Vue.JS, a rad bych svoje znalosti použil v praxi.

9.2 Datum a Čas

Při implementaci IoT časovače je důležité, aby zařízeni bylo schopné pracovat s časem rychle a korektně. Práce s objekty Date je pro většinu vývojářů často bolestivou zkušeností, protože postrádají podporu místního a časového pásma. V důsledku toho se mnoho vývojářů spojilo a vytvořilo vlastní moderní a užitečné knihovny data a času. Důrazně je doporučeno používat již existující knihovnu, která vyhovuje vašim potřebám než znovu objevovat kolo. Podívejme se blíže na základy a rozdíly mezi následujícími knihovnami JavaScript:

- **Datejs**
- **Moment.js**
- **Luxon**
- **Dayjs**

9.2.1 Datejs

Datejs je open-source JavaScriptová knihovna pro analýzu, formátování a zpracování objektů DateTime. Je to obálka pro vestavěnou funkci JavaScript Date Constructor s vylepšenými možnostmi. Ačkoli poslední oficiální vydání bylo docela dávno a od té doby nebylo aktualizováno, stále je to výkonná knihovna pro formátování data a času. Knihovna je vhodná pro ty, kteří se teprve začali učit vývoj webových aplikací.

9.2.2 Moment.js

Moment.js je lehká knihovna JavaScriptu pro analýzu, ověřování, manipulaci a formátování dat. Docela populární v komunitě, dostává časté aktualizace a má podrobnou dokumentaci, jak ji používat.

9.2.3 Luxon

Luxon je další knihovna, která usnadňuje práci s formáty data a času v JavaScriptu. Podle oficiální dokumentace poskytuje mimořádné funkce pro přidání, odečtení a analýzu data a času do formátu, který potřebujete. Ačkoli si z Moment.js vypůjčuje docela dost nápadů, má své vlastní výhody, díky kterým se od Moment.js odlišuje:

- Objekty Luxonu jsou neměnné, zatímco objekty v Moment.js jsou proměnlivé. Na rozdíl od Moment Luxon nevyžaduje konstruktory kopií ani metody klonování.
- Moment.js používá 0-indexované měsíce, zatímco měsíce v Luxonu jsou 1-indexované.
- Lokalizace a časové pásmo jsou implementovány pomocí rozhraní ECMAScript Internationalization API namísto samotné knihovny. Dodává se také s podporou poly-fill pro starší prohlížeče.
- Luxon má něco, co se nemá Moment - Duration a Intervaly.

9.2.4 Day.js

Day.js je minimalistická knihovna JavaScript, která analyzuje, ověřuje, manipuluje a zobrazuje datum a čas. Podporuje většinu moderních prohlížečů a API je podobné Moment.js. Nicméně, jak je vysvětleno v jeho dokumentaci, Day.js nabízí další výhody:

- Objekty jsou neměnné
- Je to minimalistická knihovna o velikosti max. 2 kB.

9.2.5 Závěr

Dobrým výběrem pro práce s datumem s časem je Moment.JS, který má podrobnou dokumentaci a docela populární v komunitě. Nicméně Luxon nabízí stejnou funkcionalitu, a navíc další výhody, které přebije Moment.JS.

Pro implementace datumu a času v prací bude použita knihovna Luxon.

9.3 API

Application Programming Interface (API) je soubor procedur, funkcí, protokolů a knihoven, který je využíván programátory v rámci tvorby aplikací a softwaru. Volným překladem do češtiny se jedná o rozhraní sloužící k vývoji mobilních a webových aplikací. V praxi jednotlivá API slouží k rozšíření funkcionality webu a k automatizaci určitých procedur.

API je rozhraní využívané při vývoji mobilních i webových aplikací a tvorbě internetových stránek. Smyslem API je zajištění komunikace mezi dvěma platformami, které si vzájemně vyměňují data. Umožňují využívat již naprogramovaná řešení a integrovat je do vlastních webů či softwaru, díky čemuž šetří programátorský čas a tím i peníze [47].

9.3.1 Google Calendar API

Jedním z požadavku na implementaci je synchronizace události s Google Calendar. Toho je možné docílit pouze jedním způsobem – využitím oficiálního Google Calendar API.

Google Calendar API umožňuje pracovat s kalendáři a událostmi. Ten ma knihovnu v JavaScript, Java, Go, PHP, Python a dalších programovacích jazycích [48].

Budeme potřebovat API pouze pro práce s eventy: vytvářet, mazat, upravovat a čist.

Kapitola 10

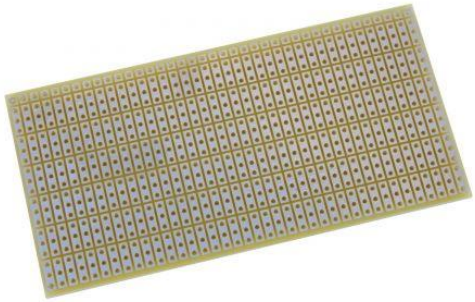
Sestavení

V této kapitole projdeme, co všechno je potřeba mít a udělat k tomu, aby možné bylo cílové IoT zařízeni postavit.

9.1 Hardware

Hlavni část IoT zařízeni je platforma, na které to bude fungovat. V našem případě to je Raspberry Pi 4 4 GB. Připravíme další moduly, senzory a komponenty:

- microSD karta
- Waveshare 4.3“ displej s DSI konektorem
- DSI konektor
- 2 reproduktory 28 mm s výkonem 0.5 W
- DHT11 teploměr
- Rotary encoder
- 3 rezistory v hodnotě 1kOm
- Univerzální spoj vrtaný 100x50mm [49]
- Drátové propojky
- LED páska
- Páječka, kapalina a cínová pájka



Obrázek 21. Univerzální spoj vrtaný 100x50mm

Dále provádíme sadu kroku:

1. Nahrát obraz operačního systému na microSD kartu podle návodu v kapitole 6.2.1
2. Připojit microSD kartu do Raspberry PI 4
3. Připojit displej k Raspberry Pi pomoci DSI konektoru
4. Sestavit modul s enkodérem a LED páskou:
   a. Připojit enkodér do spojů
   b. Připojit LED pásku do spojů
5. Připojit nový vytvořeny modul do Raspberry Pi 4
6. Připojit reproduktory do Raspberry Pi 4

9.2 Software

Ted by mělo by vše přepraveno k tomu, aby bylo možné spustit software. Připojíme Raspberry Pi 4 do zdroje napětí a pomoci externí klávesnici nainstalujeme potřebný software.

9.3 Environment

Nainstalujeme následující programy do systému Raspberry Pi OS:

- Node.JS a npm

curl -fsSL https://deb.nodesource.com/setup_14.x | bash -

sudo apt-get install -y nodejs

- Python a pip3

sudo apt-get install python3

sudo apt-get install python3-pip

- GIT

sudo apt-get install git

Nainstalujeme lokální emulátor pro Firebase služby:

npm install -g firebase-tools

9.4 Budik

Nyni stáhneme a nainstalujeme připraveny balíček s kódem který je dostupný na adrese:

git clone https://github.com/tashpjak/budik.git

cd budik

npm install

Spustíme emulátory:

firebase emulators:start

Spustíme aplikace:

npm run serve

Zapneme aplikace v kiosk modu:

/usr/bin/chromium-browser --noerrdialogs --disable-infobars --kiosk http://localhost:8081

Závěr

V této prací jsem dokázal vyrobit plně funkční IoT zařízeni na bázi Raspberry Pi 4 s využitím modulu pro měřeni teploty, tlačítka a LED pásky. Logika tohoto zařízeni je napsaná v JavaScript na bázi frameworku Vue a funguje jako klasická JS aplikace.

Zkratky a pojmy použité v práci

IoT - Internet of Thing
JS - JavaScript
Cisco IBSG - Cisco Internet Business Solutions Group
DDoS - distributed denial of service stránky
AI - artificial intelligence
ML - Strojové
SIEM - Security Information and Event Management
PMU - Power Management Unit
CPU - central processing unit
RAM - Random-access memory
Raspbian – starý název Raspberry Pi OS
Ubuntu – operační systém – distributiv Linuxu, potomek Debianu
Debian - operační systém – distributiv Linuxu
USB - Universal Serial Bus
GPIO - General-purpose input/output

Reference

[1] Weissmannová, V. (2019). Internet věcí – kultura sdílení v době nových médií [Magisterská diplomová práce]. Masarykova univerzita.

[2] Информационная безопасность устройств IoT с использованием аппаратной поддержки. (2020, December 21). Информационная безопасность устройств IoT с использованием аппаратной поддержки. https://habr.com/ru/post/534300/

[3] DoubleAgent: Taking Full Control Over Your Antivirus. Cybellum. (2017, March 22) https://cybellum.com/doubleagent-taking-full-control-antivirus/

[4] Intel® Software Guard Extensions (Intel® SGX). (2020). Intel. https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html

[5] Gregg, B. (2021). Linux perf Examples. Perf Examples. https://www.brendangregg.com/perf.html

[6] V. Jyothi, X. Wang, S. K. Addepalli and R. Karri, "BRAIN: BehavioR Based Adaptive Intrusion Detection in Networks: Using Hardware Performance Counters to Detect DDoS Attacks," 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), 2016, pp. 587-588, doi: 10.1109/VLSID.2016.115.

[7] Statista. (2021, January 22). Internet of Things - number of connected devices worldwide 2015–2025. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[8] Team, D. (2021, May 9). IoT Applications | Top 10 Uses of Internet of Things. DataFlair. https://data-flair.training/blogs/iot-applications/

[9] Bc. Václav ŠTemberg, ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE, Fakulta elektrotechnická, & Katedra počítačů. (2020). Mobilní aplikace poskytující informace o obcích. https://dspace.cvut.cz/bitstream/handle/10467/88082/F3-DP-2020-Stemberg-Vaclav-Mobilni_aplikace_poskytujici_informace_o_obcich.pdf

[10] Berg, C. (2020). Raspberry Pi 4 For Beginners And Intermediates: A Comprehensive Guide for Beginner and Intermediates to Master the New Raspberry Pi 4 and Set up Innovative Projects. Independently published.

[11] Heath, N. & TechRepublic Staff. (2020, September 3). Raspberry Pi: A cheat sheet. TechRepublic. https://www.techrepublic.com/article/raspberry-pi-the-smart-persons-guide/

[12] Heath, N. (2019, June 24). Raspberry Pi 4 Model B review: This board really can replace your PC. TechRepublic. https://www.techrepublic.com/article/raspberry-pi-4-model-b-review-this-board-really-can-replace-your-pc/

[13] F. Margret Sharmila, P. Suryaganesh, M. Abishek and U. Benny, "Iot Based Smart Window using Sensor Dht11," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), 2019, pp. 782-784, doi: 10.1109/ICACCS.2019.8728426.

[14] Meher, Chandra & Sahoo, Adyasha & Sharma, Suraj. (2019). IoT based Irrigation and Water Logging monitoring system using Arduino and Cloud Computing. 1-5. 10.1109/ViTECoN.2019.8899396.

[15] DHT11–Temperature and Humidity Sensor. (2021). Components101. https://components101.com/sensors/dht11-temperature-sensor

[16] 8 Ohm 2Watts 28 mm Dia Metal Shell Round Internal Magnet Speaker Loudspeaker 2pcs. (2020). EBay. https://www.ebay.com/itm/312647620516?hash=item48cb4017a4:g:C6AAAOSwiAJhAqp8

[17] Elektormagazine. (2020, June 29). Rotary encoder(s) on a single MCU pin. Elektor. https://www.elektormagazine.com/labs/rotary-encoder-on-a-single-mcu-pin

[18] P. (2021, May 25). Vyzkoušel jsem Ubuntu 21.04 na Raspberry Pi 4 a, bohužel, ne. Ubunlog. https://ubunlog.com/cs/Zkou%C5%A1el-jsem-ubuntu-21-04-na-Raspberry-Pi-4-a-omlouv%C3%A1m-se%2C-ale-ne/

[19] Jak nainstalovat Windows 10 na Raspberry Pi 4. 2021. https://websetnet.net/cs/jak-nainstalovat-windows-10-on-raspberry-pi-4/

[20] Venu, S. (2020). Asp.Net Core and Azure with Raspberry Pi 4: .Net Core Applications in Raspbian OS (1st ed.). Apress.

[21] T. (2019, May 22). Setting up a Raspberry Pi - Windows IoT. Microsoft Docs. https://docs.microsoft.com/en-us/windows/iot-core/tutorials/rpi

[22] Staff, M. (2021, May 19). How to Get Windows 10 IoT Core on the Raspberry Pi. DigiKey. https://www.digikey.com/en/maker/blogs/2019/how-to-get-windows-10-iot-core-on-the-raspberry-pi

[23] Broadcom releases SoC graphics driver source [LWN.net]. (2014). Https://Lwn.Net/Articles/588950/

[24] ReleaseNotes/0.6.0 – HelenOS. (2014). Helen OS. http://www.helenos.org/wiki/ReleaseNotes/0.6.0

[25] Mattias Levlin & Åbo Akademi University. (2020). DOM benchmark comparison of the front-end JavaScript frameworks react, angular, vue, and svelte.

[26] Flanagan, D. (2006). JavaScript: The Definitive Guide (Fifth ed.). O'Reilly Media.

[27] Lform Design. (2021, July 8). How JavaScript became the dominant language of the web. https://lform.com/blog/post/how-javascript-became-the-dominant-language-of-the-web

[28] Brown, K. (2021, May 5). JavaScript: How Did It Get So Popular? Codecademy News. https://www.codecademy.com/resources/blog/javascript-history-popularity/

[29] ECMA-262. (2021, July 6). Ecma International. https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

[30] Extensible markup language (XML) 1.0 (fifth edition). (2019). W3. https://www.w3.org/TR/REC-xml/#sec-origin-goals

[31] Jesse James Garrett. (2005). Ajax: A New Approach to Web Applications. Adaptive Path. https://courses.cs.washington.edu/courses/cse490h/07sp/readings/ajax_adaptive_path.pdf

[32] Birdeau, Lucas, et al. "Delivery of data and formatting information to allow client-side manipulation." U.S. Patent No. 8,136,109. 13 Mar. 2012.

[33] Git - správa verzí. (2017). GIT. https://git-scm.com/book/cs/v2/%C3%9Avod-Spr%C3%A1va-verz%C3%AD

[34] Fiala, Daniel & Vysoká škola ekonomická v Praze. (2020). Vytvoření technologické platformy pro výuku programovacího kursu node.js. Fakulta informatiky a statistiky. https://vskp.vse.cz/81202

[35] Vysoká škola ekonomická v Praze, & Kočárek, Michal. (2013). Srovnání vývoje webových aplikací v Nette frameworku (PHP) a Node.JS. Fakulta informatiky a statistiky.

[36] Vysoká škola ekonomická v Praze, & Kvítek, Karel. (2020). Moderní technologie a nástroje pro vývoj webových aplikací. Fakulta informatiky a statistiky. https://vskp.vse.cz/79787

[37] ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE, & Václav ŠTemberg. (2020). Mobilní aplikace poskytující informace o obcích. Fakulta elektrotechnicka. https://dspace.cvut.cz/bitstream/handle/10467/88082/F3-DP-2020-Stemberg-Vaclav-Mobilni_aplikace_poskytujici_informace_o_obcich.pdf

[38] Get realtime updates with Cloud Firestore |. (n.d.). Firebase. Retrieved August 11, 2021, from https://firebase.google.com/docs/firestore/query-data/listen

[39] Cloud Functions. (2020). Firebase. https://firebase.google.com/docs/functions

[40] Google. Introduction to Firebase Local Emulator Suite. Firebase. Retrieved August 11, 2021, from https://firebase.google.com/docs/emulator-suite

[41] Luke Joliat. (2019). Do we still need JavaScript frameworks? Free Code Camp. https://medium.freecodecamp.org/news/do-we-still-need-javascript-frameworks-42576735949b/

[42] State of JS 2020: Front-end Frameworks. (2020). State of JS. https://2020.stateofjs.com/en-us/technologies/front-end-frameworks/

[43] Hannah, J. (2018, March 12). The Ultimate Guide to JavaScript Frameworks. JavaScript Report. https://jsreport.io/the-ultimate-guide-to-javascript-frameworks/

[44] Dawson, C. (2021, June 6). JavaScript's History and How it Led To ReactJS. The New Stack. https://thenewstack.io/javascripts-history-and-how-it-led-to-reactjs/

[45] Comparison with other frameworks Vue.js. (2021). Vue.JS. https://vuejs.org/v2/guide/comparison.html

[46] The State of JavaScript 2018. (2018). State of JS. https://2018.stateofjs.com/

[47] Koďousková, B. (2021, July 15). Co je to API a jaké jsou možnosti jeho využití? Rascasone.com. https://www.rascasone.com/cs/blog/co-je-api

[48] Google Calendar API Overview. (2021). Google Developers. https://developers.google.com/calendar/api/guides/overview

[49] S.R.O, W. R. (2021). Univerzální spoj vrtaný, 100x50mm, FR4 TA 020 | GM electronic, spol. s.r.o. GME. https://www.gme.cz/univerzalni-spoj-100x50-rm-2-54-pasove-spoje-tri-body

[50] F. Rahman, M. Farmani, M. Tehranipoor and Y. Jin, "Hardware-Assisted Cybersecurity for IoT Devices," 2017 18th International Workshop on Microprocessor and SOC Test and Verification (MTV), 2017, pp. 51-56, doi: 10.1109/MTV.2017.16

Přílohy

Obrázky

1. F. Rahman, M. Farmani, M. Tehranipoor and Y. Jin, "Hardware-Assisted Cybersecurity for IoT Devices," 2017 18th International Workshop on Microprocessor and SOC Test and Verification (MTV), 2017, pp. 51-56, doi: 10.1109/MTV.2017.16.
2. TPM-M R2.0 card https://www.asus.com/Motherboards-Components/Motherboards/Accessories/TPM-M-R2-0/
3. The Raspberry Pi Foundation. (2020). Buy a 4 Model B –. Raspberry Pi. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
4. Použiti Raspberry Pi s Ethernet adapterem
5. DHT11 senzor. https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/
6. DHT11 modul. https://components101.com/sensors/dht11-temperature-sensor
7. Reproduktory. https://www.ebay.com/itm/312647620516?hash=item48cb4017a4:g:C6AAAOSwiAJhAqp8
8. Rotary Encoder. https://www.elektormagazine.com/labs/rotary-encoder-on-a-single-mcu-pin
9. Stavy enkoderu. https://thepihut.com/blogs/raspberry-pi-tutorials/how-to-use-a-rotary-encoder-with-the-raspberry-pi
10. LED páska s 8 diody. https://www.laskarduino.cz/8x-inteligentni-rgb-led-neopixel-pasek--ws2812b--5050--5v/
11. Schéma propojených součástek pro IoT zařízeni. Fritzing
12. Pi Imager - úvodní obrazovka
13. Pi Imager – výběr operačních systému
14. Pi Imager – úspěšný zápis
15. Umístění konfiguračního souboru config.txt
16. Vztahy mezi verzemi JavaScript. https://www.doria.fi/bitstream/handle/10024/177433/levlin_mattias.pdf
17. Ukázka kódu XML.
18. Ajax. https://www.doria.fi/bitstream/handle/10024/177433/levlin_mattias.pdf
19. Firebase Emulatory. https://firebase.google.com/docs/emulator-suite
20. Popularita JS frameworku. https://2019.stateofjs.com/front-end-frameworks/
21. Univerzalni spoj vrtany. https://www.gme.cz/univerzalni-spoj-100x50-rm-2-54-pasove-spoje-tri-body

Tabulky