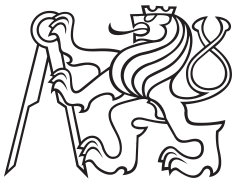
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Systém pro řízení pěstebního prostředí

Jiří Štengl

Vedoucí práce: Ing. Lukáš Zoubek
Květen 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Štengl** Jméno: **Jiří** Osobní číslo: **499132**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Otevřená informatika**

Specializace: **Základy umělé inteligence a počítačových věd**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Systém pro řízení pěstebního prostředí

Název bakalářské práce anglicky:

System for Controlling of Growing Enviroment

Pokyny pro vypracování:

Cílem práce je vytvořit prototyp informačního systému pro řízení pěstebního prostředí (zahrady). IS se bude skládat z webové aplikace určené ke správě uživatelů a konfiguraci řídících jednotek a ze samotného systému pro řízení prostředí.
1. Definujte obecné požadavky na systém a konkretizujte ho na použití v zahradě.
2. Proveďte rešerši existujících řešení.
3. Vymyslete architekturu systému a jednotlivých komponent.
4. Navrhněte webovou aplikaci včetně zabezpečené komunikace mezi webovou aplikací a moduly.
5. Navrženou webovou aplikaci implementujte.
6. Navrhněte architekturu modulů a jejich komunikace.
7. Navrhněte schémata pro jednotlivé moduly.
8. Vytvořte z připravených schémat desky plošných spojů, desky osaďte.
9. Vytvořte firmware pro jednotlivé moduly.
10. Celé řešení otestujte.

Seznam doporučené literatury:

[1] J. Canete, Cipriano Galindo, and Inmaculada Moral. Introduction to Control Systems, pages 137–165. 01 2011.
[2] George F Coulouris, Jean Dollimore, and Tim Kindberg. Distributed Systems. International Computer Science Series. Addison-Wesley Edu- cational, Boston, MA, 4 edition, June 2005.
[3] Chang-Le Zhong, Zhen Zhu, and Ren-Gen Huang. Study on the iot archi- tecture and gateway technology. In 2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), pages 196–199, 2015.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Lukáš Zoubek Centrum znalostního managementu FEL

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

Datum zadání bakalářské práce: **22.01.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Lukáš Zoubek	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	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

Poděkování

Děkuji rodičům za oporu při studiu a při psaní této práce. Dále bych pak chtěl poděkovat mému bratrovi Ing. Václavu Štenglovi za rady při implementaci backend služby. Dále také děkuji svému školiteli Ing. Lukáši Zoubkovi za věnovaný čas a možnost navštěvovat kancelář firmy Procesment. Děkuji také doc. Ing. Stanislavu Vítkovi, Ph.D, za přístup do laboratoře 720 na katedře radioelektroniky a za rady při vývoji systému. V neposlední řadě chci poděkovat Ing. Ladislavu Kaiserovy, který mi předal mnoho znalostí o tvorbě a implementaci enterprise systémů.

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.

V Praze dne 26.5.2023

Abstrakt

Práce se zabývá analýzou, návrhem a implementací IoT systému pro správu venkovního pěstebního prostředí. Systém se skládá z webové aplikace a z distribuovaných modulů založených na modulech ESP32. Systém je navržený pro čtyři druhy modulů, kde jeden slouží jako master ostatních (gateway) a ostatní se k němu připojují pomocí bezdrátového protokolu ESP-NOW. Uživatel si může gateway připojit ke své lokální Wi-Fi síti a pomocí webové aplikace si nastavovat závlahu zahrady.

Klíčová slova: IoT, internet věcí, zahrada, informační systém, pěstování

Vedoucí práce: Ing. Lukáš Zoubek
Centrum Znalostního Managementu
Technická 2
13393 Praha 6

Abstract

In this thesis you can find analysis, draft and implementation of IoT system for controlling of growing enviroment. System is composed out of web application and from distributed modules based on ESP32 modules. System is designed for four types of modules, which one of them serves as master (gateway) to other modules, which are connecting as slaves via wireless protocol ESP-NOW. User can connect gateway to his local Wi-Fi network and via web application set automatic irrigation rules for his garden.

Keywords: IoT, internet of things, garden, information system, cultivation

Title translation: A system for managing the growing environment

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V dnešní době se stále častěji setkáváme se zařízeními, které jsou schopny komunikovat jak s okolním světem tak i se světem IT. Tato zařízení nám mohou pomáhat a zefektivňovat dennodenní činnosti či zastávat nějaké povinnosti za nás. Jednou z těchto činností je pěstování rostlin, které s aktuálním přístupem je dlouhodobě neudržitelné a to jak z důvodu nadměrného používání hnojiv tak neefektivního využití vody.

Cílem této práce je navrhnout a vytvořit informační systém, který se bude skládat z IoT systému a webové aplikace běžící v cloudu. Systém umožňuje uživateli nastavit automatické zalévání zahrady podle hodin synchronizovaných z internetu, dále bude umožňovat uživateli dívat se na stav zařízení na své zahradě. Nastavování systému probíhá pomocí webové aplikace, která slouží jako uložiště nastavení a můstek mezi uživatelem a zařízeními na zahradě.

Webová aplikace se skládá ze dvou částí a to backend a frontend, které mezi sebou komunikují pomocí rozhraní REST. Webová aplikace dále poskytuje REST rozhraní pro gateway, která periodicky stahuje nastavení zalévání zahrady a ukládá si to do své flash paměti pro případ výpadku webové služby či výpadku spojení.

Hlavním zařízením zodpovědným za chod zařízení rozmístěných na zahradě je gateway, která slouží jako spojovací bod mezi webovou aplikací a moduly na zahradě a zároveň i jako orchestrátor jednotlivých úkonů prováděných zařízeními. Gateway funguje ve dvou módech, kde první mód se stará o nastavení přihlašovacích údajů pro lokální síť a druhý se stará o celkovou funkcionalitu systému. V neposlední řadě systém řeší problém použití více gateway na jednom místě díky nastavitelným mac adresám, pomocí kterých se jednotlivé moduly připojí ke gateway. Další moduly, které se v systému nacházejí jsou čidla vlhkosti a teploty vzduchu, čidla vlhkosti půdy a řadič pro ovládání až 6 nezávislých solenoidových ventilů sloužící pro ovládání přívodu vody k jednotlivým záhonkům.

S ohledem na rozsah této bakalářské práce je implementován pouze PoC (Proof of Concept) s gateway a řadičem aktuátorů. Moduly pro sběr dat teploty a vlhkosti půdy jsou pouze navržené a osazené, ale firmware pro ně není implementován.

1.1 Členění práce

Práce je členěna celkem do tří tematických celků. Prvním celkem je teoretická čast, kde je shrnuto jak jsou známe architektury softwarových a distribuovaných systémů, internetu věcí a druhy databází. Druhým tematickým celkem je analýza požadavků na systém, kde jsou shrnuty jak uživatelské požadavky tak technické požadavky jako například propustnost systému. Všechny požadavky na systém lze najít v kapitole osm, kde se nachází FURPS+ analýza. Do druhého tematického celku patří také rešerše existujících řešení, kde jsou zhodnoceny klady a zápory jednotlivých řešení. V poslední části práce lze najít návrh řešení analyzovaného problému, jeho implementaci a následné otestování naimplementovaného řešení.

Kapitola 2
Způsoby pěstování rostlin

"Zemědělství je jedním z odvětví materiální výroby, jehož finální výsledky vycházejí z bezprostředního působení společnosti na přírodu." [27] S použitím moderních technologii se také začaly vyskytovat nové způsoby pěstování rostlin, které tak pomáhají vylepšit jeho produktivitu. V dnešní době se nejvíce využívá klasické polní zemědělství, hydroponie a akvaponie. Ve městech se také v menší míře můžeme setkat s alternativními způsoby pěstování rostlin.[27]

2.1 Polní zemědělství

Polní zemědělství je v dnešní době nejvíce rozšířeným způsobem zemědělství v České republice. V České republice k roku 2015 zabírala obdělávaná půda 53% celé rozlohy státu [8] z toho 71% orná půda na které se podle pěstebních plánu vysazují jednotlivé plodiny, které jsou následně obdělávany. Tento typ zemědělství má nevýhodu ve výskytu škůdců, kteří mohou ničit úrodu. Další nevýhodou je obtížná kontrola vnějších podmínek pro pěstování rostlin. [7]

2.2 Hydroponie

Hydroponie je technologie pro pěstování rostlin v živném roztoku s a nebo bez přidaných umělých podloží v podobě písku, štěrku, kamennou vatou atd... pro ukotvení rostlin. [16] Dále se hydroponické systém dělí na otevřené a zavřené. Otevřené systémy po doručení živného roztoku kořenům rostlin dále tento roztok již nepoužívají. Naopak oproti tomu uzavřené systémy roztok obohatí o živiny a znovu použijí. Oproti polnímu zemědělství jsou zde rostliny pěstovány v uzavřeném systému a lze tak dokonale ovládat podmínky jako jsou například teplota a vlhkost vzduchu, světlo, poměr živin v půdě atd... Dalšími výhodami hydroponie jsou velká efektivita využití vody a šetrnost k životnímu prostředí.

2.3 Akvaponie

Tato ve světě hojně využívaná metoda spojuje produkci vodních živočichů s pěstováním rostlin."[13] Tato metoda spojuje funkce akvakultury a hydroponie,

kde akvakultura je chov ryb pro konzumaci člověkem. Akvaponie využívá faktu, že v akvakultuře ryby produkují biologický odpad jak v podobě pevného skupenství tak plynů rozpuštěných ve vodě, zejména pak dusík v podobě amoniaku, který je pro ryby silně toxický a tudíž musí být z akvakultury filtrován pryč. Zde pak přichází na řadu nitrifikační bakterie v kořenech rostlin, díky kterým je amoniak přeměňován na dusičnany, které jsou výborným hnojivem pro rostliny. [13]

Kapitola 3

Kontrolní systémy

Kontrolní systémy se zabývají úpravou stávajících systému pro požadavky určitého modelu chování [10]. Kontrolní systémy dělíme na systémy se zpětnou vazbou a bez zpětné vazby.

3.1 Kontrolní systémy bez zpětné vazby

Nejjednodušším možným kontrolním systémem, je systém bez zpětné vazby. Systém kvůli chybějící zpětné vazbě není schopen kontrolovat zadanou veličinu přímo a proto je zde potřeba se spoléhat na jinou veličinu a správné nastavení systému obsluhou. Příkladem takovéhoto systému může být zavlažování, které nekontroluje přímo vlhkost zahrady, ale pouze má nastavené, že má danou zahradu zalévat 5 minut. Nevýhody systému bez zpětné vazby je velké náchylnost na chyby jak způsobené okolím (celý den pršelo a tudíž zahrada je již dostatečně vlhká) tak způsobené špatnou obsluhou (5 minut na zavlažení může být moc a nebo příliš málo).

3.2 Kontrolní systémy se zpětnou vazbou

Kontrolní systémy se zpětnou vazbou již obsahuje zpětnou vazbu a tudíž nemusí využívat jinou veličinu ke svému běhu. Systém si je schopen sám změřit hlídanou veličinu, určit odchylku od cílové hodnoty a podle velikosti odchylky reagovat na stav systému pomocí výstupů. Oproti kontrolním systému bez zpětné vazby je zpětnovazební systém schopen reagovat na zásah do systému okolím a přizpůsobit aktuálnímu stavu svůj výstup (celý den pršelo, zahrada je již zalita a není potřeba závlaha). Zpětnovazební kontrolní systémy rozdělujeme do dvou skupin podle typu zpětné vazby a to na systémy s pozitivní zpětnou vazbou a s negativní zpětnou vazbou. [10]

3.2.1 Kontrolní systémy s pozitivní zpětnou vazbou

Jak již název napovídá, systém reaguje na změnu zpětné vazby stejným směrem jako proběhla změna (pokud se veličina snížila, sníží se výstup a vice versa). Nevýhodou těchto systému je limitní chování po dosažení určité

hranice dané veličiny. Celý systém se tedy nakonec chová jako bistabilní. Tohoto jevu se velmi často využívá v digitální elektronice. [10] [21]

3.2.2 Kontrolní systémy s negativní zpětnou vazbou

Kontrolní systémy s negativní zpětnou vazbou fungují přesně obráceně než systémy s pozitivní zpětnou vazbou (pokud se veličina sníží, výstup se zvýší a vice versa). Výhody negativního zpětnovazebního systému oproti systému s pozitivní zpětnou vazbou jsou větší stabilita a škálovatelnost. Nevýhodou zpětnovazebního systému je, že může velmi výrazně ovlivňovat chování celého systému. [10]

Kapitola 4

Architektura softwárového systému

Architektura softwárového systému je zabývá rozložením systému na jeho hlavní komponenty. [9] Komponenty se obvykle skládají z jednoho či více modelů a plní vlastnosti nějaké služby (například v bankovním systému máme komponentu/službu starající se o platby kartou, další komponentu starající se o platbu mezi bankami atd...). Komponenty mohou mezi sebou komunikovat a být spolu provázané ruznými vazbami. Model je zjednodušující popis reálného světa, který pomáhá softwárovým inženýrům lépe zachytit okolní svět do svého programu. Na celou problematiku se dá pohlížet jako na sérii několika kroků.

4.1 Komponenty a konektory

K použití komponentovému návrhu se přistupuje zejména na začátku vývoje systému. Zde se definují funkcionality jednotlivých komponent s minimálním ohledem na následnou implementaci. Jednotlivé komponenty jsou na sobě nezávislé a tím dochází k enkapsulaci celého systému. Konektory slouží k definování API mezi jednolivými komponentami a nastavení vlastnostním jednotlivým konektorům (zdali je konektor pouze read-only, nebo zdali může i zapisovat). Konektory nemusí vést pouze z komponenty A do komponenty B, ale mohou se libovolně větvit. Díky tomu dochází k dobrému zobrazení toku informací za běhu programu.[9]

4.2 Rozdělení na logické bloky

Dalším krokem je rekurzivní rozdělování problému na menší a menší logické celky, které jsou snadno implementovatelné. Tento model architektury se zejména hodí pro statickou analýzu problému a požadovaného běhu daného programu. Důležité u rozdělení na logické bloky je, aby nikdy v modelu nevznikl cyklus, který by zničil stromovou architekturu a aby žádný uzel neměl více jak jednoho předka.[9]

4.3 Alokace zdrojů

K alokaci zdrojů dochází až po připravení návrhu pomocí dříve zmíněných metod, kdy je potřeba návrh začít implementovat na reálných zařízeních. Zde je nutné vymyslet jak bude vypadat celková podoba programu, z jakých souborů a souborových struktur se bude program skládat, kam se umístí konfigurační soubory atd... Dalším prvkem alokace zdrojů je příprava prostředí a hardwaru na nasazení aplikace. K mírným úpravám dochází i za běhu/implementace programu, kde se můžou ukázat některé body jako úzká hrdla celého systému.[9]

4.4 Definice chování aplikace

Poslední částí návrhu architektury softwáru je definice chování jednotlivých komponent a chování komponent navzájem mezi sebou. Dochází zde k návrhu sledu jednotlivých kroků pro zhotovení určité akce. Zde je nutné se soustředit na možné krajní situace, které mohou v systému nastat jako jsou například uváznutí, přetečení zásobníku se zprávami a jiné.[9]

Kapitola 5
Distribuované systémy

Distribuované systémy se skládají z více hardwárových či softwárových komponent umístěných ve společné počítačové síti, které spolu komunikují a koordinují své akce pouze pomocí posílání zpráv [12]. Jednotlivé komponenty mohou být libovolně umístěny v prostoru a mohou se tedy nacházet na stejném zařízení, ve stejné budově a nebo třeba i na druhé straně světa. Distribuované systémy mají díky svým vlastnostem velké spektrum využití. Od MMO her, přes big data distribuované databáze až po síť senzorů.[12]

5.1 Druhy distribuovaných systémů

Distribuované systémy jsou děleny na distribuované výpočetní systémy, distribuované informační systémy a pervazivní/IoT systémy.

5.1.1 Distribuované výpočetní systémy

Mezi distribuované výpočetní systémy se řadí počítačové clustery, cloudy a počítačové gridy. Úkolem těchto systémů je poskytnou vysokou výpočetní kapacitu pro velmi náročné aplikace. Cílem je reprezentovat skupinu distribuovaných počítačů jako jeden superpočítač a odstranit tak špatně škálovatelné modely pro paralelní výpočty. [26]

5.1.2 Distribuované informační systémy

Distribuované informační systémy se zejména hodí pro velké aplikace stávající z několika rozlišných celků/služeb, ke kterým lze přistupovat přes společnou bránu. Výhoda takového systému je, že může pokrývat služby několika rozdílných společností, vysoká spolehlivost a konzistence. [26]

5.1.3 Pervazivní/IoT systémy

Narozdíl od předchozích dvou druhů jsou pervazivní systémy běžně umístěny v našem okolí a uživatel neinteraguje s těmito systémy pouze pomocí jednoho rozhraní (osobní počítač), ale i svými dalšími akcemi. Jednotlivá koncová zařízení jsou obvykle vybaveny senzory/aktuátory pro komunikaci

s okolím. Dalším rozdílem oproti předchozím typům je nestálost počítačové sítě a nutnost vypořádat se s možnými výpadky, které v datových centrech nenastávají. [26]

5.2 Výzvy při designu distribuovaných systémů

Definice distribuovaných systému s sebou přináší i vlastnosti, na které je potřeba si při designu distribuovaných systémů dát pozor.[12]

5.2.1 Sdílení zdrojů

Distribuovaný systém musí být dostatečně robustní, aby zvládl obsluhovat všechny klienty a zároveň přístup ke sdíleným datům byl spolehlivý. Přístup ke sdíleným zdrojů se dá řešit pomocí tokenových a beztokenových algoritmů. Tokenové algoritmy mají vytvořený jeden token pro celý systém, který umožňuje uzlu vlastnící tento token přistupovat ke společnému zdroji. Mezi tokenové algoritmy můžeme zařadit algoritmus s centrálním uzlem na předávání tokenu a nebo kruhový algoritmus, kde token postupně v cyklu cestuje mezi jednotlivými servery. Na druhou stranu Ricart-Agrawalův algoritmus patří mezi beztokenové algoritmy, kde se procesy mezi sebou domlouvají kdy chtějí přistoupit do společné sekce.[17]

5.2.2 Nepřítomnost globálních hodin

V reálném světě v dnešní době neexistuje způsob jak přesně synchronizovat všechny komponenty distribuované sítě na stejný čas. Problém se synchronizací nastává, protože jednotlivé komponenty společně komunikují pomocí zpráv, které mají nějaké zpoždění, které není přesně známo. V případech, kdyby se mohli reálné hodiny jednotlivých zařízení velmi lišit se používají logické hodiny, které se inkrementují nikoliv podle tiků oscilátoru na fyzickém zařízení, ale nýbrž podle proběhlých akcí na zařízení. Příkladem takových hodin mohou být Lamportovy logické hodiny. [12] [15]

5.2.3 Odolnost vůči selhání

U distribuovaných systému je snaha co nejvíce omezit možné selhání systému a v případě selhání co nejdříve oznámit administrátorovy systému selhání. Selhání se dělí na havárie, kdy dojde k zastavení vykonávání procesu a na byzantské selhání, kde proces stále pracuje, ale pracuje chybně. Dále se u distribuovaných systému můžeme setkat se selháním komunikačních kanálů a to zejména se ztrátami jednotlivých zpráv nebo rozdělením procesů na dvě disjunktní množiny, které spolu nekomunikují. V neposlední řadě se můžeme setkat s časovými selháními, kde proces neodpoví do nějakého předem definovaného časového úseku. [12] [14]

Kapitola 6
Internet věcí

Internet věcí je označení scénářů, kdy se počítačová síť a výpočetní prostředky rozšiřují na běžné předměty a senzory a umožňuje těmto zařízením generovat, vyměňovat mezi sebou a spotřebovavát data s minimálním zásahem člověka [24]. Hlavním cílem Internetu věcí (Internet of Things - IoT) je usnadnit, automatizovat nebo monitorovat procesy, které by si jinak žádaly lidského pracovníka. [24][19]

6.1 Architektura internetu věcí

V internetu věcí se nejčastěji používá tří nebo pětivrstvá architektura. V obou těchto architekturách vyšší vrstva vždy obaluje všechny vrstvy pod ní a přidává k nim další funkcionalitu.

6.1.1 Třívrstvá architektura

Třívrstvá architektura se skládá ze tří vrstev. První je vstupně/výstupní vrstva, ve které se nachází všechna fyzická zařízení, která sbírají data nebo nějakým jiným způsobem interagují s prostředím. Druhá vrstva je síťová kde dochází ke komunikaci mezi zařízeními navzájem a mezi webovou službou a zařízeními. Poslední vrstvou je aplikační, pomocí které může uživatel přistupovat k celému systému a interagovat s ním. [23]

6.1.2 Pětivrstvá architektura

Pětivrstvá architektura oproti třívrstvé architektuře rozděluje síťovou a aplikační vrstvu na dvě vrstvy. Síťovou vrstvu rozděluje na vrstvu lokální sítě a na přenosovou vrstvu. Vrstva lokální sítě se stará o lokální komunikaci mezi zařízeními a o komunikaci mezi gateway (lokální master zařízení, které komunikuje s cloudovou službou) a zařízeními. Přenosová vrstva je komunikační vrstva mezi gateway a službami v cloudu. Aplikační vrstva je rozdělena na podpůrnou aplikační vrstvu a na prezentační vrstvu. Podpůrná aplikační vrstva se stará o sběr dat a o komunikaci s gateway, vyhodnocování dat, ukládání dat a další. Zatímco prezentační vrstva se stará o uživatelské rozhraní a zobrazení dat. Celá architektura potom vypadá takto: vstupně/výstupní

vrstva, vrstva lokální sítě, přenosová vrstva, podpůrná aplikační vrstva a prezentační vrstva. [28]

6.2 Komunikační modely

U internetu věcí se můžeme setkat s více druhy komunikace. Jak již mezi zařízeními tak i mezi zařízeními a webovou službu a webovou službou a uživatelem. [24]

6.2.1 Model komunikace mezi zařízeními

Model komunikace mezi zařízeními reprezentuje dvě nebo více zařízení, které mezi sebou přímo komunikují [24]. Komunikace může probíhat pomocí různých technologií v závislosti na požadavcích na velikost dat, rychlost přenosu a energetických potřebách zařízení. Nejčastěji se používají technologie jako je Wi-Fi, Bluetooth, ZigBee, Z-Wawe, Lo-Ra a další. Bohužel v dnešní době neexistuje žádný ucelený standard, který by udával protokol na komunikaci mezi zařízeními.

6.2.2 Model komunikace cloudová služba a zařízení

V tomto modelu se zařízení rovnou připojuje ke cloudové službě přes internet. Tento model se používá u zařízení, která mají obvykle pouze jeden modul, jako jsou například termostaty, zámky u dveří nebo chytré televize. Výhoda tohoto modelu spočívá ve využívání již vytvořené počítačové sítě a možnosti přenosu velkého množství dat mezi cloudovou službou a zařízením. [24]

6.2.3 Model komunikace gateway a zařízení

Tento model využívá společnou gateway pro více lokálních zařízení, které s gateway komunikují a následně gateway komunikuje se službou v cloudu. Gateway se stará kompletně o zabezpečení celého lokálního systému k přístupu na internet. Tento model komunikace využívají zejména zařízení, která se sama nemohou připojit na internet jako jsou například chytré hodinky, kde chytrý telefon uživatele slouží jako gateway, nebo chytré domácnosti jako je například Amazon Alexa. [24]

6.2.4 Model komunikace webových služeb

Model komunikace webových služeb je doplňujícím modelem k předchozím modelům. Jeho cílem je spojit různé služby od různých dodavatelů a tím vytvořit jedno ucelené řešení. Zde se zejména využívají webové technologie jako jsou https, REST API, oauth2 a další. [24]

6.3 Zabezpečení internetu věcí

Internet věci přináší do zabezpečení aplikace před možnými kyberútoky nové výzvy, které je potřeba řešit. Narozdíl od běžných webových služeb internet věcí obsahuje fyzické komponenty, ke kterým se útočník může dostat a proto je důležité chránit všechny vstupní/výstupní body systému. Možnou hardwarovou ochranou může být ochrana proti elektromagnetickému rušení, uzamknutí EEPROM/flash paměti proti přepsání a při výpadku napájení automatické uvedení zařízení do bezpečné polohy (například automatické zamknutí domu). Důležitá je ochrana i před softwarovými útoky, které mohou být zaměřené jak na fyzické zařízení tak i na službu v cloudu s daty o uživatelích. U fyzických zařízení jsou zranitelná zejména zařízení připojující se na internet a komunikující se službou v cloudovém prostředí. Zde je potřeba zejména vhodně ošetřit zabezpečení komunikace mezi zařízeními a webovou službou, dále pak je potřeba správně nastavit zařízení pro obsluhu neznámých požadavků, aby nedošlo například k přetečení zásobníku a tím k nedefinovanému chování zařízení. V neposlední řadě je důležité správně vytvořit zabezpečení pro webovou službu, která komunikuje se zařízeními a s uživateli. Zde může dojít k úniku citlivých dat uživatelů. Zabezpečení komunikace mezi webovou službou a gateway není v práci implementováno.[24]

Kapitola 7
Databáze

Databáze je strukturovaná kolekce dat [22]. V dnešní době se můžeme setkat se dvěma typy databázových systému a to jsou SQL a NoSQL. Databázové systémy se zaobírají třemi základními požadavky a to jsou: konzistence, dostupnost a tolerance oddílů. Konzistence se týká toho, že data po úspěšné vykonání vkládací nebo updatovací operace jsou bezpečně uložena a všichni uživatelé databáze si je mohou přečíst. Dostupnosti systému lze dosáhnout tím, že systém pracuje bezchybně a vždy je připraven reagovat na příkazy od uživatelů. Dostupnosti lze dosáhnout tím, že databáze může být nasazena na cluster nezávislých uzlů, kde nevadí když jednotlivé uzly vypadnou. Tolerance oddílů je funkce, kdy samotná databáze je schopna rozpoznat spadlé uzly a požadavky mířené na tyto uzly zpracovat jinde. [22]

7.1 Relační databáze

SQL (Structured Query Language) databáze ukládají data v podobě tabulek. Data v nich uložená musí mít předem daný tvar a typ dat. Tabulky je možné propojovat pomocí relací a to v podobě N:M, 1:N a 1:1 vazbám mezi jednotlivými řádky databáze. Kvůli těmto operacím mohou být zapisovací a čtecí operace pomalejší, protože je potřeba napočítat správné provázání dat podle podmínek z různých tabulek. Předností SQL databázi je konzistence dat. Každá SQL databáze splňuje takzvané ACID podmínky. [22]

- Atomicita - všechny transakce musí být splněné celé. Pokud dojde k přerušení transakce v půli jejího běhu celá transakce je zrušena a databáze vrácena do původní podoby.
- Konzistence - databáze přechází pouze mezi stabilními stavy. Nesmí dojít k poškození dat přerušenými transakcemi.
- Izolace - v SQL databázi je možné nastavit více druhů izolace jednotlivých transakcí, které se nesmí navzájem ovlivňovat.
- Odolnost - všechny úpravy dat jsou ukládany do logů, aby v případě výpadku systému mohlo dojít ke správnému doplnění dat.

7.2 NoSQL databáze

NoSQL (Not only SQL) jsou databáze kde podoba dat není přesně dána předem pomocí relačního modelu. NoSQL databáze se převážně zaměřují na dostupnost a toleranci oddílů (takzvaně jsou škálovatelné do šířky - lze je rozdělit na více nezávislých strojů), zatímco konzistence zde není tolik absolutní jako u SQL databází. Jedná se o takzvanou případnou konzistenci, kdy systém po nějaké upravující akci potřebuje čas na zpropagování operace ke všem uživatelům. Jedná se zejména o následek dobré škálovatelnosti do šířky, kdy je potřeba informaci propagovat na všechny servery. NoSQL databáze jsou oproti SQL databázím novější a proto může docházet k častějším nálezům různých chyb a vad v implementaci, které nebyly zatím odhaleny. NoSQL databáze se dělí na několik typů podle jejich funkcionality. [20][22]

- Key-Value databáze se podobají hashovacím tabulkám. Klíč je řetězec a hodnota může být libovolný objekt. Mezi implementované řešení lze zařadit Amazon DynamoDB, RIAK atd...
- Column-Oriented databáze jsou nejvíce podobné tradičním relačním databázím. Narozdíl od SQL databází nejsou data uložena v tabulce, ale v přenosném formátu (například JSON), který lze distribovat dobře na více serverů. Tyto databáze se především hodí na data minning a analytické operace na daty. Příklady těchto databázi jsou Google BigQuery, MongoDB, CouchDB a další.
- Grafové databáze ukládají data do grafů, kde jednotlivé objekty jsou vrcholy grafu a relace mezi objekty jsou hrany grafu. Grafové databáze narozdíl od ostatních NoSQL databází splňují ACID podmínky. Jejich využití je zejména pro data s velkým množstvím relací jako jsou například sociální sítě. Příklady implementací je například Neo4j.
- Objektově orientované databáze ukládají data jako objekty. U těchto objektů podporuje dědičnost, polymorfii a enkapsulaci. Každý objekt má své jednoznačné ID. Nevýhodou těchto databází je především omezená škálovatelnost. Hojně jsou využívány pro počítačově asistovaný návrh a ve vědecké sféře. Příklad tohoto typu je db4o.

7.3 Porovnání SQL a NoSQL databází

Jako hlavní výhody NoSQL databází oproti SQL databázím lze považovat velkou rozmanitost použitelných datových modelů, dobrou podporu škálovatelnosti, agilní vývoj, flexibilitu, efektivitu a rychlost. Na druhou stranu nejsou NoSQL databáze dostatečně odladěny jako jsou SQL databáze dále pak chybí standart pro komunikaci s NoSQL databázemi. V neposlední řadě je potřeba při práci s daty si dát pozor na možné nesplnění ACID podmínek, které mohou vyústit v krátkodobou nekonzistenci dat. V závěru lze říci, že NoSQL databáze se specializují na ukládaní velmi velkých dat, které nemusí

mít předem pevně danou strukturu a následné práci s nimi. Zatímco SQL databáze se specializují na relace mezi objekty a definují přesnou podobu dat.[20]

Kapitola 8
FURPS analýza

FURPS je model kvality navržený firmou HP v roce 1987. Model dekomponuje požadavky/charakteristiky systému do dvou skupin a to na funkcionální a nefunkcionální. Funkcionální požadavky definují vstupy a k nim očekávané výstupy softwaru zatímco nefunkcionální požadavky jsou kladeny na systém jako takový. Mezi nefunkcionální požadavky se řadí použitelnost, spolehlivost, výkon a superoperabilita. Dále v textu budou zmíněny požadavky na systém pro správu zahrad.[25]

8.1 Funkcionalní požadavky

V této kapitole se zaměříme na funkcionální požadavky systému pro správu zahrad.

8.1.1 Požadavky na webovou aplikaci

Požadavky pro webovou aplikaci lze rozdělit do dvou logických celků. První požadavek se týká na správu uživatelů a jejich přístupu do webové aplikace. Druhá část se skládá z komunikace aplikace s koncovými moduly.

Uživatelské požadavky na webovou aplikaci

F1. Možnost vytvoření uživatelského účtu a propojení ho s Google účtem.

F2. Uživatel si bude moci nechat resetovat heslo v případě zapomenutí.

F3. Možnost vytvoření zahrady a rozmístění jednotlivých záhonů na zahradě.

F4. Možnost vlastnictví více zahrad na jednom účtu.

F5. Možnost umístění jednotlivých záhonů na určitá místa ve svojí zahradě.

F6. Možnost zobrazení dat v grafu za vybraný časový úsek.

F7. Pojmenování jednotlivých senzorů a možnost umístění v zahradě.

F8. Možnost zapnout/vypnout závlahu manuálně.

F9. Možnost nastavení periody sběru dat z jednotlivých čidel.

F10. Možnost udání polohy zahrady a zaslání upozornění uživateli na případné změny počasí.

F11. Možnost smazání účtu.

F12. Odhlášení uživatele ze systému.

Systémové požadavky na webovou aplikaci

F1. Možnost propojení čidel a gateway s webovou aplikací.

F2. Kominukace s gateway u uživatele, sběr informací z gateway a ukládání informací do databáze.

F3. Promazávání starých záznamů nasbíraných z koncových modulů uživatelů.

F4. Možnost nastavení podmínek pro závlahu, podle času, podle vlhkosti půdy a nebo podle času a vlhkosti půdy.

F5. Odeslání emailu/upozornění při chybě některého ze senzorů/aktuátorů nebo gateway.

8.1.2 Požadavky na koncové moduly

F1. Odolnost proti přírodním živlům.

F2. Automatické připojení modulů ke gateway.

F3. Vstupní moduly budou napájené z baterie/akumulátoru.

F4. Možnost vybrání si z n možných kanálů po kterých může gateway se zařízeními komunikovat.

F5. Možnost hard resetu nastavení gateway pomocí tlačítka.

F6. Pokud gateway nemá nastavenou wifi síť tak poskytne uživateli možnost připojit gateway na nějakou z lokálních sítí, které najde v okolí.

F7. Gateway umožní změnit své nastavení i po připojení k wifi. Například umožní uživateli změnit na jinou wifi síť atd.

F8. Zařízení budou periodicky v uživatelem nastavených periodách komunikovat s gateway a posílat jí svá data. Gateway odpoví zařízení, na jeho zprávu svou zprávou. Pokud zařízení zprávu nedostane bude se pokoušet zprávu odeslat znovu.

F9. Gateway bude hlídat stav zařízení a pokud uplyne timeout od zprávy z nějakého zařízení gateway to ohlásí webové aplikaci.

F10. Komunikace mezi gateway a webovou aplikací bude zabezpečená.

F11. Při výpadku napájení si gateway i koncová zařízení budou pamatovat uživatelské nastavení.

F12. Při výpadku internetového spojení si bude gateway pamatovat posledních n počet naměřených hodnot, které po konci výpadku odešle webové aplikaci.

F13. Systém bude schopný správně fungovat i bez přístupu k internetu a správně se řídit podle nastavení od uživatele.

F14. Systém bude obsahovat senzory na měření vlhkosti a teploty ovzduší.

F15. Závlahovací moduly budou kontrolovat průtok vody a v případě příliš nízkého průtoku vody oznámí uživateli chybu?

8.2 Použitelnost

P1. Správná funkčnost frontend části aplikace na moderních prohlížečích.

P2. Jednoduché připojení modulů ke gateway.

P3. jednoduché připojení gateway k lokální síti wifi.

P4. Vstupní moduly budou pracovat na baterie.

P5. Jednoduché připojení zavlažovacího modulu k závlažnému systému zahrady.

P6. Odolnost modulů proti přírodním živlům.

8.3 Spolehlivost

S1. Aplikace bezchybně upozorní uživatele na poruchu v jeho lokálním systému.

S2. Webová aplikace bude dostupná alespoň 95% času pro uživatele.

S3. Webová aplikace si bude bezpečně a spolehlivě ukládat data od uživatelů a data naměřená ze zahrad uživatelů.

S4. Moduly budou pracovat spolehlivě i za zhoršených přírodních podmínek.

S5. Gateway bude fungovat i při výpadku internetového připojení.

8.4 Výkon

V1. Tisíce až desítky tisíc záznamů ze senzorů denně.

V2. Řádově jednotky až desítky uživatelů.

V3. Obsluha řádově jednotek až desítek gateway.

8.5 Podporovatelnost

P1. Systém poběží v cloudovém prostředí, které umožňuje snadnou správu systému.

P2. Volba správného build systému pro snadou správu verzí použitých knihoven a frameworků.

P3. Možná správa uživatelských účtů pomocí administrátorského účtu.

P4. Cloudová aplikace bude implementována v jazyce Java.

P5. Frontendová část webové aplikace bude implementována za použití knihovny React.

Kapitola 9

Rešerše existujících řešení

V této kapitole je provedený průzkum komerčních řešení řešící automatickou závlahu zahrady a možnost správy celého systému pomocí webové/mobilní aplikace.

9.1 Gardena zavlažovací systém Micro-Drip se závlahovým počítačem

Jedná se o systém kapkové závlahy, který funguje na principu dávkování vody přesně k rostlinám v pravidelných intervalech. [1] [2]

Výhody:

- Zpětně kompatibilní
- Velmi efektivní využití vody
- Možnost přidělat ke kohoutku
- Možnost nastavení automatického zalévání podle času
- Dobře přizpůsobitelný konkrétní zahradě
- Umí měřit vlhkost půdy

Nevýhody:

- Není možné nastavit množství závlahy u jednotlivých rostlin
- Vhodné pouze pro menší zahrady
- Absence propojitelnosti s webovou aplikací

9.2 Immax NEO LITE Smart zavlažovací systém

Chytrý zavlažovací systém, který se skládá z hlavní řidicí jednotky, která se připojuje do sítě uvnitř domu a až do čtyř venkovních zařízení, které slouží pro závlahu zahrady. Systém je možný nastavovat pomocí mobilní aplikace. [3]

Výhody:

- Mobilní aplikace až v šedesáti jazycích
- Dlouhá výdrž baterií ve venkovních čidlech
- Dlouhý dosah bezdrátové komunikace
- Funkce měření spotřebované vody

Nevýhody:

- Pouze možnost nastavení z mobilní aplikace
- Rozměrná vnější jednotka
- Pouze 2 zalévací kanály

9.3 Rachio 3

Rachio 3 je řídící počítač pro chytré zálévání zahrad. Je možné ho ovládat z mobilní aplikace, Amazon Alexy nebo Google asistenta. Rachio 3 je možné objednat ve třech variantách a to podle požadovaného počtu zalévaných ploch a to na čtyři, osm a šestnáct. [6]

9.3.1 Výhody:

- Možnost propojení s domácími asistenty a ovládání pomocí hlasu
- Díky lokální předpovědi je možné nastavit šetření vody při dešti, sněhu a dalších
- Možnost ovládat až šestnáct nezávislých ploch

9.3.2 Nevýhody:

- Nutnost dokoupit venitly pro ovládání jednotlivých kanálů
- Nutnost přímého spojení kabelů a ventilů jednotlivých kanálů
- Poměrně rozměrné (23 cm * 3.5 cm * 14 cm) a těžké (0.45kg)
- Absence zpětné vazby pomocí měření vlhkosti půdy

9.4 Orbit 57946

Orbit 57946 je řídící počítač pro zavlažování zahrady. Vyrábí se ve dvou variantách a to ve varinatě s šesti kanály a ve variantě s dvanácti kanály. Systém lze připojit k lokální síti pomocí Wi-Fi a ovládat pomocí mobilní aplikace nebo webové aplikace. V neposlední řadě lze Orbit připojit k Amazon Alexa verze 6 a vyšší. [5]

Výhody:

- Možnost ovládat zařízení i mimo lokální síť
- Certifikovaný společností EPA pro svou schopnost šetřit vodou
- Možnost dokoupit senzor pro detekci deště
- Možnost nastavit automatické přerušení zálévání podle lokální předpovědi počasí
- Možnost nastavení jednoduchých funkcí pomocí tlačítek na zařízení
- Možnost připojit k chytré domácnosti a ovládat pomocí hlasu

Nevýhody:

- Nutnost dokoupit venitly pro ovládání jednotlivých kanálů
- Nutnost přímého spojení kabelů a ventilů jednotlivých kanálů
- Absence čidel pro měření vlhkosti půdy
- Nemožnost sledovat hodnoty v průběhu času

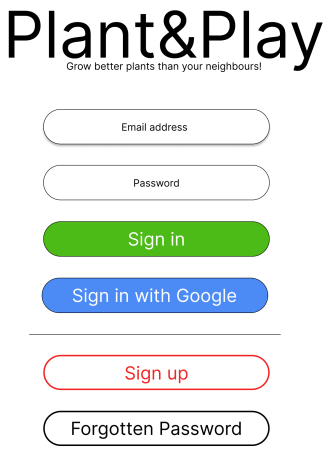
9.5 Shrnutí rešerše

Z rešerše je patrné, že žádný z prozkoumaných systémů neumožňuje skladovat naměřená data a nějakým způsobem je procházet či analyzovat. Další nevýhodou systému je chybějící rozšiřitelnost do systémů pro hydroponii a akvaponii. Naopak silným prvkem systémů bývají mobilní aplikace, které poskytují komfortní řešení ovládání celého systému, jak přes Bluetooth, tak přes Wi-Fi síť. U dražších systému k tomuto trendu můžeme pozorovat i možnost propojení s domácími asistenty jako je Amazon Alexa či Google smart home.

Kapitola 10

Návrh zpracování systému

Dalším krokem v návrhu informačního návrhu je pokrytí požadávků specifikovaných ve FURPS analýze a představení pro ně návrh řešení. Pro lepší představu pokrytí problému byly vytvořeny wireframy ("skicy webu"), které slouží jako předloha pro samotnou webovou aplikaci. Všechny wireframy lze nalézt v příloze této práce.



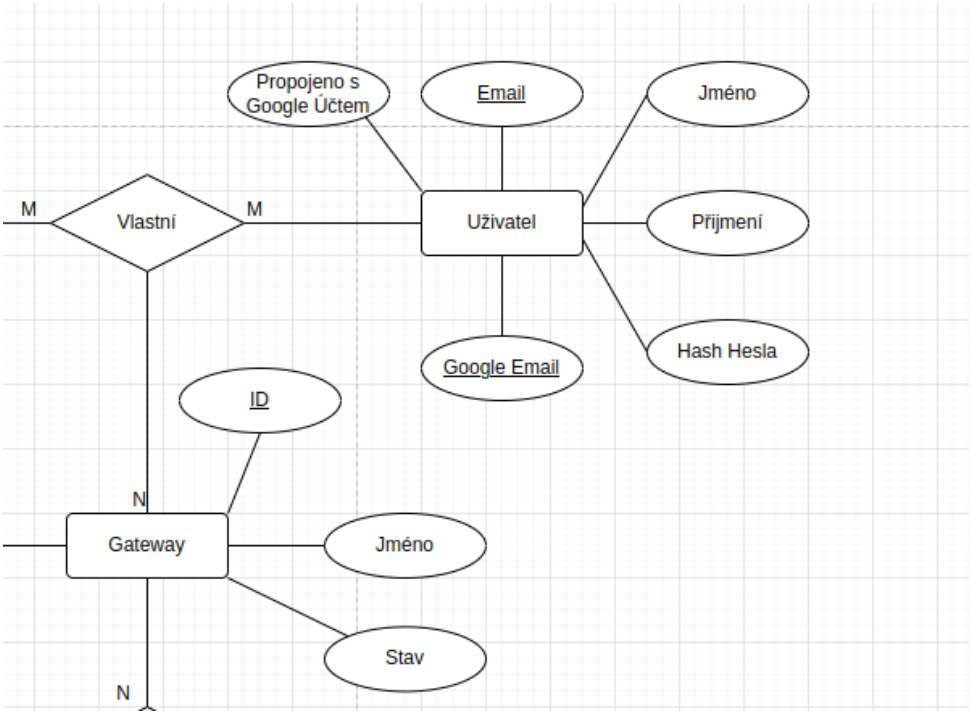
Obrázek 10.1: Příklad návrhu úvodní stránky pro uživatele na přihlášení

Požadavky	Wireframy									
	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
F1	X	X								
F2	X		X	X						
F3							X			
F4							X		X	
F5							X			
F6									X	
F7							X			
F8							X			
F9							X	X		
F10					X				X	
F11						X				
F12										X

Tabulka 10.1: Tabulka obsloužení jednotlivých požadavků pomocí wireframů

10.1 Databázový model webové aplikace

K návrhu a lepšímu porozumění problému byl navržen konceptuální databázový model, který modeluje jednotlivé entity a jejich vlastnosti, primární klíče a relace mezi nimi. Klíčovou entitou celého modelu je entita uživatel, která má jako primární klíč email a jsou na ni vázany přímo a nebo nepřímo všechny další entity. Dále se entity dělí podle toho, zdali reprezentují reálná zařízení a nebo pouze slouží jako pomocné prvky pro uživatele v orientaci v systému a abstrakci nad reálný svět. Jako pomocné entity lze označit entity záhon a zahrada, které mají za úkol pomoci uživateli s rozmístěním jednotlivých senzorů a aktuátorů do systému tak aby to odpovídalo jeho reálné zahradě. Naopak entity gateway, aktuátor, senzor a kanál jsou entity reprezentující reálná zařízení. Databázový model lze najít v příloze této práce.



Obrázek 10.2: Příklad entit uživatel a gateway v databázovém konceptuálním modelu

10.2 Architektura komunikace mezi gateway a moduly

Pro komunikaci mezi gateway a moduly byla zvolena architektura master-slave, kde gateway slouží jako master a ostatní moduly jsou slaves. Dále je třeba rozdělit moduly na ty, které jsou napájeny ze sítě a ty které jsou napájeny pomocí baterii. U modulů napájených ze sítě lze zprávy z gateway posílat přímo, protože při napájení ze sítě dané moduly nemusí vypínat

rozhraní pro komunikaci a uspávat své čipy, které je nutné při napájení z baterií. Na druhou stranu moduly, které jsou napájeny z baterii, budou většinu času uspané a proto kdyby zprávy vysílala gateway tak může dojít k jejich ztrátě z důvodu, že daný modul bude právě uspaný. Z tohoto důvodu je nutné, aby se bateriemi napájené moduly periodicky samy hlásily gateway a vyčkali na odpověď gateway. Tímto způsobem je zajištěno, že gateway bude mít přehled o všech připojených zařízeních a jejich aktuálním stavu.

10.3 Architektura komunikace mezi gateway a webovou aplikací

Požadavky pro komunikaci mezi gateway a webovou aplikací jsou takové, že webová aplikace musí být schopna předat informace ze serveru gatewayi. K tomuto účelu byly zvažovány 2 technologie. První je long-polling (vytvoření REST requestu, který má prodlouženou dobu vypršení) a druhý je vytvoření web socketu, který slouží pro dvoucestnou komunikaci. Pro účel této práce byl zvolen přístup long-pollingu, který umožňuje díky své konečné odezvě lépe detekovat výpadky heartbeatů z gateway a upozornit tak uživatele na možný problém. Další výhodou long-pollingu oproti socketu je větší robustnost řešení, protože výpadek socketu může být občas špatně detekovatelný. V neposlední řadě nástroje pro směrování požadavků na služby na serveru dokáží lépe operovat s HTTP requesty, než s otevřenými sockety, které by se musely zavřít a znovu otevřít při přidání/odebrání instance webové služby.

10.4 Zabezpečení komunikace mezi gateway a webovou aplikací

V dnešní době čelí IoT systémy velké výzvě v podobě zabezpečení a to jak na úrovni fyzické tak na úrovni informační. V této sekci bude probrána informační bezpečnost. Problém informační bezpečnost lze rozdělit podle druhů útoků, které mohou nastat.

10.4.1 Denial-of-service útok

Jedním z nejčastějších útoků je takzvaný útok DOS (Denial-of-service), nebo jeho distribuovaná verze DDOS. DOS/DDOS je útok, který se snaží blokovat přístup legitimních uživatelů do systému pomocí snížení dostupnosti systému jeho přetížením [11]. Tento druh útoku se jeví problematický zejména pro zařízení, která jsou volně dostupná z internetu. V případě tohoto systému může být tímto útokem zasažena pouze webová aplikace, protože gateway jakožto jediný modul připojený k internetu, je schována za routerem, který neposkytuje statickou IP addresu a tudíž není možné k zařízením za routerem přistupovat přímo z internetu. V případě výpadku webové služby je nutné, aby gateway byla schopna pracovat i autonomně a nedošlo k neočekávanému chování, kvůli kterému by mohlo dojít k poškození majetku.

10.4.2 Man-in-the-middle útok

Man-in-the-middle je druh kybernetického útoku, kde neoprávněný prostředník vkročí do online komunikace mezi mezi dva uživatele, kteří o jeho přítomnosti nevědí. Útočník poté odposlouchává nebo mění obsah komunikace podle svých požadavků [18]. Obvyklou ochranou proti tomuto druhu útoků je šifrování komunikace. IoT systémy mají v tomto určitou výhodu i určitou nevýhodu. Výhoda spočívá v tom, že obě strany komunikace, jak server tak modul jsou implicitně považovány za důvěryhodné a tudíž není potřeba si předávat klíč pro symetrickou šifru pomocí handshaku. Na druhou stranu je nutné, aby klíč z modulu nikdo nemohl přečíst a kompromitovat bezpečnost komunikace ostatních modulů pomocí známosti klíče. V tomto případě je nutné klíč periodicky měnit. Druhá možnost je použít například protokol SSL, který používá pro předání klíče asymetrickou šifru a po předání klíče komunikuje pomocí symetrické šifry. Nevýhoda tohoto řešení spočívá v náročnosti a požadavcích na výpočetní prostředky, protože asymetrické šifry nejsou tak odolné na útok hrubou silou jako šifry symetrické a proto je potřeba používat delší klíče. Toto může být limitující faktor pro mikrořadiče s nízkým výkonem a nedostatečnou velikostí RAM.

Kapitola 11

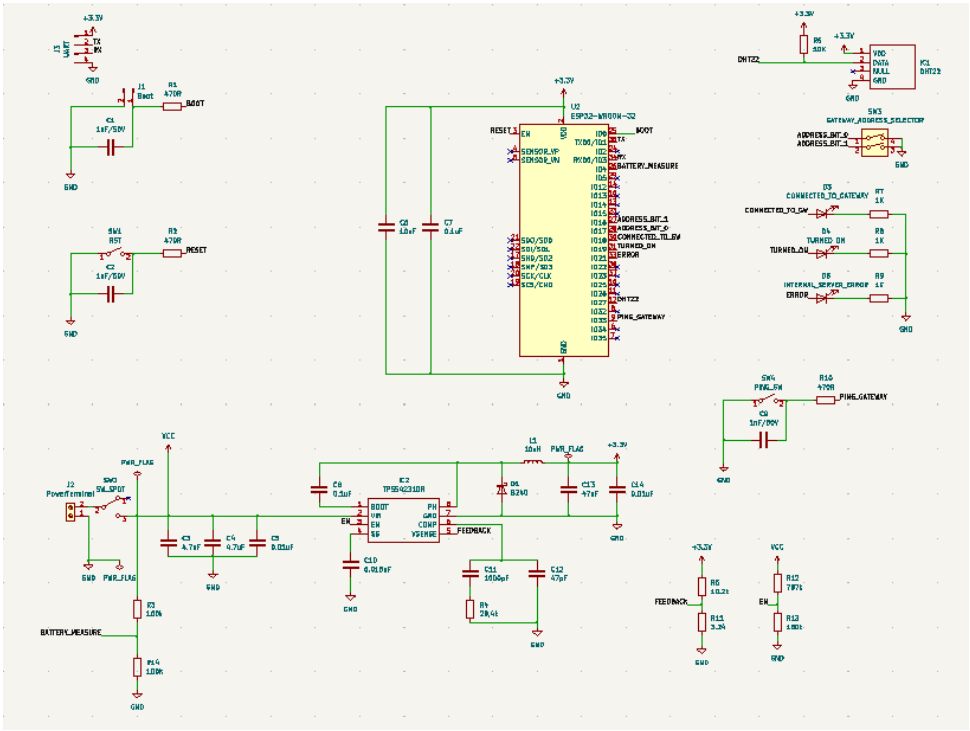
Návrh a výroba desek plošných spojů

V této kapitole bude probráno jaký byl postup při návrhu a výrobě plošných spojů pro jednotlivé moduly.

11.1 Návrh schémat a výběr elektronických součástek

Před návrhem schémat zapojení plošných spojů bylo nutné zvolit správné součástky, které budou použity pro výrobu modulů. Nejtěžší volbou bylo zvolení mikrořadiče a modulu pro komunikaci. Po dohodě s vedoucím práce byl zvolen modul ESP32, který prokazoval nejlepší vlastnosti v poměru cena/výkon. Výběrem ESP32 bylo rozhodnuto, že výpočetní logika bude soustředěna na 3.3V a bylo potřeba tomu přizpůsobit návrh schémat. Velmi důležitý prvek, který byl tímto výběrem ovlivněn byl regulátor a rozsah napájecího napětí. Po dohodě s vedoucím byl zvolen jako zdroj napájení 4 AAA baterie pro moduly napájené bateriemi což odpovídá napětí 3.6-6V a 5-12V pro moduly napájené ze sítě. K tomuto účelu byl zvolen step-down buck regulátor TPS54231DR, který je velmi efektivní a odpovídá napěťovému rozsahu pro obě použití.

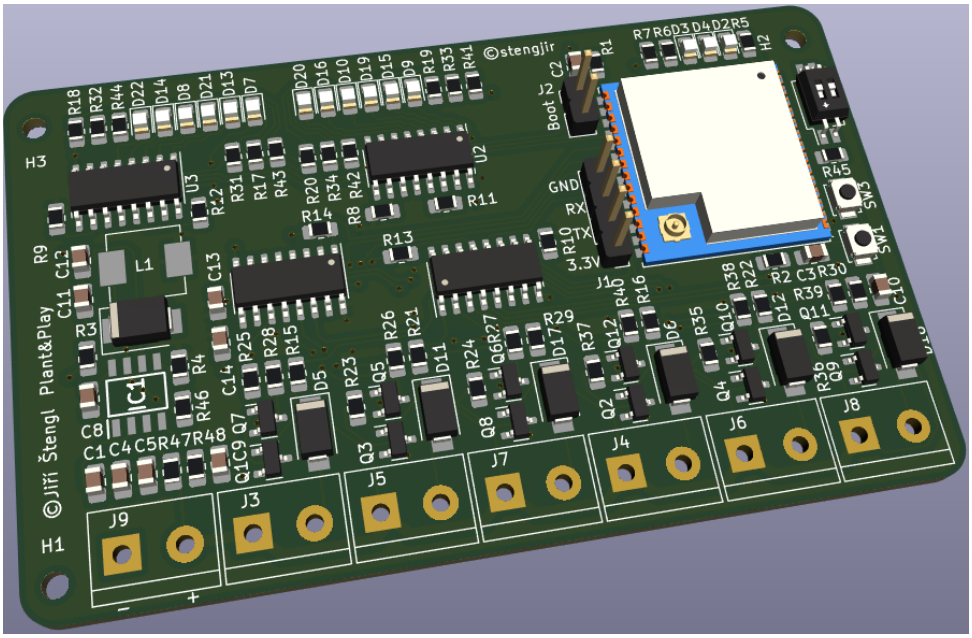
Schémata byla navržena podle funkčních schémat jednotlivých desek, které byly vytvořeny na základě FURPS+ analýzy a schváleny vedoucím práce. K návrhu schémat byl použit program KiCad 6.0. Schémata vycházela ze základního zapojení modulů ESP32, které se dají nalézt v dokumentaci od výrobce. Tato schémata byla dále doplněna o součástky potřebné k vykonávání činnosti jednotlivých funkcí zařízení. Kompletní schémata i se seznamy součástek lze nalézt v příloze.



Obrázek 11.1: Příklad schématu pro teplotní čidlo

11.2 Návrh desek plošných spojů

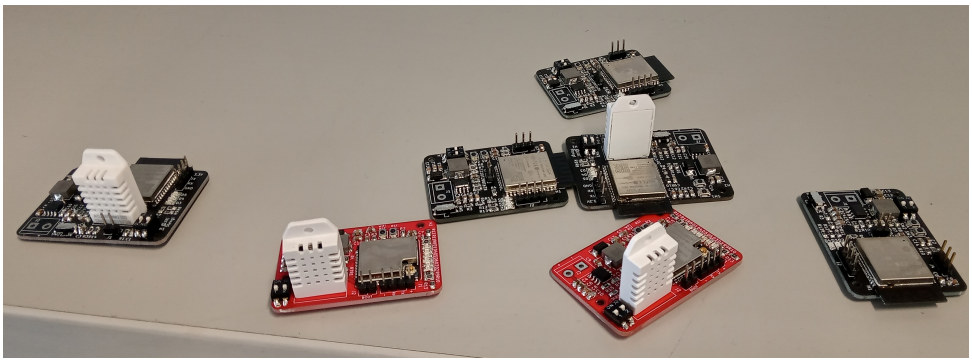
Pro návrh DPS bylo potřeba nejdříve obstarat všechny potřebné footprinty pro jednotlivé součástky, některé byly staženy z volně dostupných zdrojů, jiné bylo potřeba vyrobit. Požadavek na desky byl, aby byly co nejmenší a proto byla zvolena technologie SMD. Desky byly navrženy v programu KiCad 6.0.



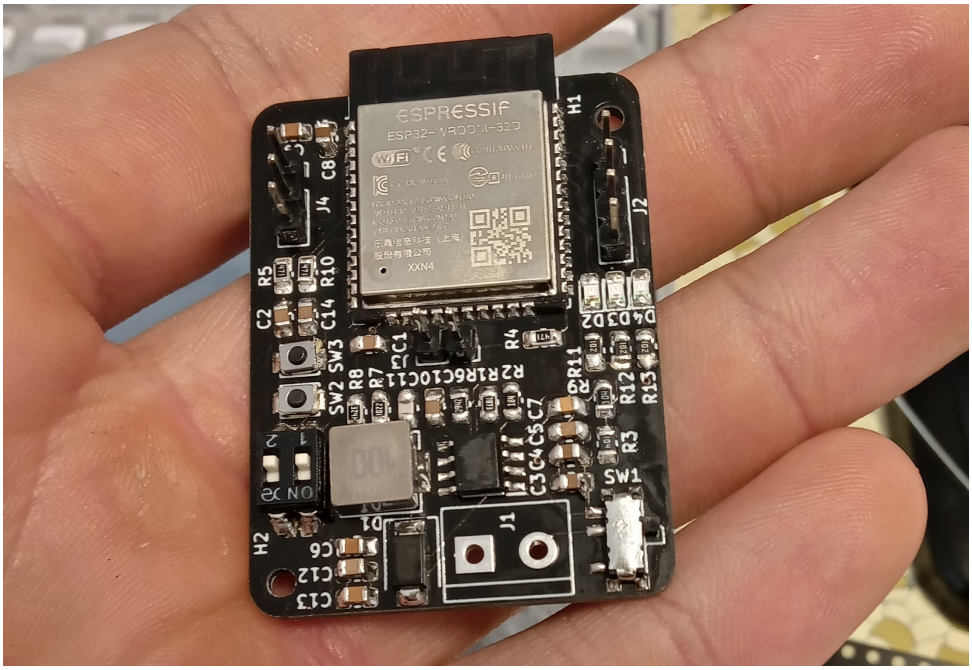
Obrázek 11.2: 3D model desky driveru ventilů vygenerovaný v programu KiCad 6.0

11.3 Výroba a osazení desek plošných spojů

Desky plošných spojů byly vyrobeny Čínskou společností JLCPCB, která se specializuje na výrobu prototypů elektronických desek. Desky byly osazeny pomocí ruční mikropáječky, za použití tavidla pro hůře pajitelné součástky.



Obrázek 11.3: Fotografie osazených desek v laboratoři v místnosti 720 na katedře radioelektroniky



Obrázek 11.4: Fotografie desky čidla pro měření vlhkosti půdy v mé dlani bez osazených svorkovnic

11.4 Nalezené hardwarové chyby

Po správném osazení desek a při implementaci firmware byly objeveny chyby jak v jednotlivých schématech, tak i v samotných součástkách. Opravené chyby ve schématech lze najít v příloze. Zde je výčet chyb a popis potřebných úprav pro jejich odstranění:

- Chybějící pull-up rezistor u resetovacího pinu. Zde je potřeba přidat 10kΩmezi +3.3V a resetovací pin.
- U desek, které jsou poháněny méně jak 7V DC, je špatně zvolen rezistor o hodnotě 787kΩ, který je potřeba vyměnit za hodnotu 330kΩnebo podobnou.
- U gateway, je potřeba přidat další vyhlazovací kondenzátor k ESP-32, protože při použití 10µF a 0.1µF (doporučených výrobcem) v paralelním zapojení dochází k opětovnému brownout restartu celého modulu, kvůli příkonové proudové špičce.
- Chybějící propojení na desce řadiče ventilů mezi pinem parallel load (PL) na čipu SN74HC165 a pinem modulu ESP32. Bez tohoto propojení nemůže být čip SN74HC165 přepnut do stavu načítání/vyčítání dat přes sériový pin a pořád tak načítá data asynchroně do paralelních pinů. Tím dochází k poškození čtených dat. Řešení zde bylo přidat vodič mezi GPIO 15 na modulu ESP32 a PL.

Kapitola 12

Zvolené technologie pro implementaci

V této kapitole budou vybrány a zdůvodněny technologie pro implementaci informačního systému.

12.1 Webová aplikace

Analýza bude rozdělena na části pro backend, frontend a databázový systém. Tyto 3 skupiny by se daly označit jako základní pilíře všech informačních služeb.

12.1.1 Backend

Serverová část aplikace implementována ve frameworku SpringBoot 2.7.5 a za pomocí programovacího jazyka Java 17. Verze 17 byla zvolena z důvodu, že se jedná o nejnovější verzi LTS. Jako build system bude použit Maven.

12.1.2 Frontend

Na vývoj frontendové části bude použita JavaScriptová knihovna React, která byla vyvinuta společností Meta a díky řadě knihoven poskytuje vývojáři velkou škálu možných předpřipravených komponent. Vývoj projektu bude v jazyce TypeScript, který poskytuje typovou podporu pro jazyk JavaScript a je kompilován do jazyka JavaScript při buildu aplikace.

12.1.3 Databázový systém

Jako databázový systém byl zvolen PostgreSQL. Jedná se o open-source enterprise level SQL databázový systém. Relační databáze byla zvolena z důvodu znalosti struktury dat a možnosti je propojit pomocí relací a vytvořit tak efektivní dotazovací příkazy. Další velkou výhodou je možnost zavedení indexů, které mohou velmi zrychlit práci s daty ze senzorů, jejichž objem s časem bude růst.

12.2 Firmware

Pro implementaci firmware byl zvolen MicroPython. MicroPython je zmenšená a efektivní implementace programovacího jazyka Python 3, která poskytuje k dispozici malou podmnožinu funkcí ze standartní knihovny a je optimalizovaná pro běh na mikrořadičích [4]. U MicroPythonu je nutné nejdříve na mikrořadič nahrát interpreter/operační systém, který poskytuje uživateli MicroPython REPL (Read-Eval-Print Loop) se kterou může standartně komunikovat přes sériovou linku a kde si uživatel může psát příkazy stejně jako v Python REPLu. MicroPython byl zvolen z důvodu velmi rychlého a efektivního vývoje, který se hodí pro rychlé prototypování.

Kapitola 13
Testování

Testování software a zařízení je důležitým prvkem každého vývojového procesu. Testování je možné provádět několika možnými způsoby.

13.1 Testovací scénaře

Testovací scénáře slouží pro testování aplikace z uživatelského pohledu pomocí předem definovaných kroků a kontroly, že všechny body scénáře byly v pořádku splněny.

13.1.1 Zařízení je schopno se připojit k webové službě a zaregistrovat se

Scénář se zabývá připojením gateway k webové službě pomocí REST rozhraní a webová služba si gateway uloží do databáze pokud tam ještě není, pokud se již gateway nachází v databázi tak pouze aktualizuje časovou známku posledního přístupu gateway k webové aplikaci.

Postup scénáře

P1. Spuštění webové aplikace a databáze v dockeru

P2. Spuštění gateway

P3. Vyčkání na připojení gateway k lokální síti (signalizováno LED)

P4. Vyčkání na odeslání požadavku gateway na webovou aplikace (signalizováno logem)

P5. Kontrola logu a databáze, že nová gateway byla přidána do systému/byla úspěšně aktualizována časová známka posledního připojení gateway k webové službě

P6. Kontrola logů gateway, že odpověď byla v pořádku přijata

13.2 Vyhodnocení testovacích scénářů

Testovací scénáře proběhly bez chyb.

13.3 Automatické testy

Automatické testy se v softwarovém inženýrství používají pro automatické testování naprogramovaných řešení. Rozlišujeme několik druhů testů pomocí jejich rozsahu. Nejmenší svým rozsahem jsou jednotkové (unit) testy, které testují funkcionalitu jednotlivých funkcí nebo metod. Poté jsou aplikační testy, které testují funkcionalitu celé aplikace jako celku. Posledním používaným typem testů jsou integrační testy, které testují integraci jednotlivých systémů.

13.3.1 Testování webové služby

V práci byly využity jednotkové a aplikační testy ve webové aplikaci. Testy jsou spouštěny při stavbě programu pomocí build systému Maven. Aplikace je testována jak na funkcionalitu jednotlivých metod tak na správnou rozhraní, které poskytuje pro připojení klientů. Rozhraní je testováno tak, že je spuštěna celá aplikace i s databází, která se spouští pomocí Spring containerů. Rozhraní je testováno pro validní i nevalidní vstupy a je kontrolováno, že vrací správnou odpověď a zároveň provede validní kroky v databázi systému. Pro zajištění kvality webové aplikace byla vytvořena i pipeline na Gitlabu, která automaticky při každé změně webové služby danou službu postaví, otestuje a výsledek zobrazí.

13.3.2 Testování firmware zařízení

Firmware zařízení je obvykle poměrně složité testovat z důvodů, že stav zařízení velmi často nezáleží pouze na stavu softwaru, ale i na stavu hardwarových vstupů jako takových. Dále pak může o funkčnosti celého zařízení rozhodovat schéma zapojení jednotlivých součástek a jejich správné osazení. Z těchto důvodů je možné firmware testovat standartními postupy pouze ve velmi omezené míře a to buď v simulátoru, nebo na samotném zařízení ve spuštění konkrétních procedur, které nepotřebují interagovat s okolním hardwarem. Z tohoto důvodu byly pro testování řešení zvoleny postupy, kde kód zařízení byl otestován s ohledem na funkci celého zařízení.

Kapitola 14

Závěr

V práci jsem postupně uvedl většinu druhů pěstování rostlin a to jak zemní hospodářství, tak akvaponii a hydroponii. U zemního hospodářství jsem uvedl požadavky na systém. Modelování a sběr požadavků jsem prováděl pomocí FURPS+ analýzy, kde jsem nejdříve vymyslel jednotlivé požadavky na systém jak funkcionální tak i systémové. Po zjištění uživatelský požadavků na systém byl čas se podívat na komerční řešení, které se zaobírají stejným, nebo podobným problémem. Zde bylo potřeba daná řešení analyzovat z pohledu jejich kladů a záporů a najít společný problém jímž je nemožnost systém dobře rozšířit.

Dále v práci jsem se zaobíral architekturou IoT systému a to jak z teoretické části, kde jsem provedl rešerši na stávající postupy při vývoji IoT systému tak i z praktické části, kde jsem vymyslel model architektury systému, pro tento konkrétní problém. Narženou architekturu lze shrnout jako síť modulů v konfiguraci Master-Slave, kde master (gateway) komunikuje s webovou aplikací v cloudu a orchestruje svoje slaves (konkrétní moduly). Po vymyšlení architektury systému bylo nutné vymyslet i architekturu jednotlivých komponent systému jako je například backend webové aplikace, jednotlivé moduly a další. Zde návrh probíhal s cílem splnit všechny nadefinované požadavky na systém ve FURPS+ analýze. Ke splnění tohoto úkolu jsem mimo jiné použil i navrhlé wireframy webové aplikace, které poskytují představu rozložení uživatelského rozhraní. Dále pak bylo potřeba nadefinovat jak spolu jednotlivé části systému budou komunikovat. Zde jsem zvolil komunikaci pomocí REST rozhraní, jak pro frontend část aplikace, tak i pro komunikaci mezi gateway a webovou službou.

Po navržené architektuře komunikace bylo potřeba se zamyslet nad možným zabezpečním toku dat, mezi backendem webové služby jejím uživatelským rozhraním a gateway. Zde jsem analyzoval, které ze známých kybernetických útoků by mohly být problematické a ty jsem se pokusil vyřešit na teoretické úrovni. Po dohodě s vedoucím práce jsme odstoupili od nápadu je do praktické části systému dávat kvůli nízkému výkonu gateway modulu, který by měl velké problémy upočítat potřebné šifry pro zabezpečený přenos dat. Po vyřešení problematiky ohledně zabezpečení toku dat bylo potřeba se podívat více do hloubky na rozhraní, které bude vystavovat webová aplikace jak pro uživatelské rozhraní tak pro gateway.

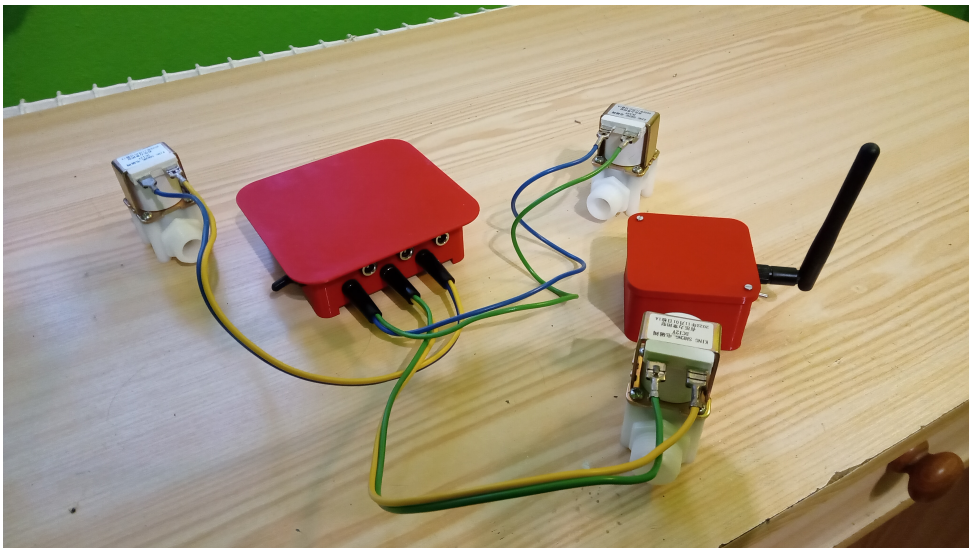
Z důvodů dlouhé dodací lhůty pro dodání potřebných součástek a desek jsem se před implementací webové služby pustil do návrhu desek plošných spojů, které jsem navrhl v programu KiCad 6.0. U desek bylo důležité vybrat správné součástky takovým způsobem, aby odpovídaly požadavkům jak na rozsah napájecího napětí, tak na efektivní využití energie. Po vybrání všech součástek přišla složitější část rozmisťování součástek na desku plošných spojů a jejich routování. U některých součástek bylo potřeba si vytvořit vlastní footprinty, protože pro ně neexistovaly knihovny, které by se daly přidat do KiCad. Následně bylo potřeba desky osadit k tomu jsem využil laboratoř 720 na katedře radioelektroniky.

Dalším krokem bylo pustit se do implementace webové služby. Zde jsem po dohodě s vedoucím práce rozhodl neimplementovat původní aplikaci v plné verzi jak byla navržena, ale pouze proof of concept celé problematiky a to zejména kvůli velkému rozsahu bakalářské práce. Upustili jsme od implementace ovládání závlahy podle vlhkosti země, ale zvolil jsme pouze možnost zavlažovat podle hodin. K implementaci backendu webové služby byla použita Java 17 a framework Spring Boot 2.7.5. K implementaci uživatelského rozhraní byla použita knihovna Javascriptová knihovna React a programovací jazyk Typescript.

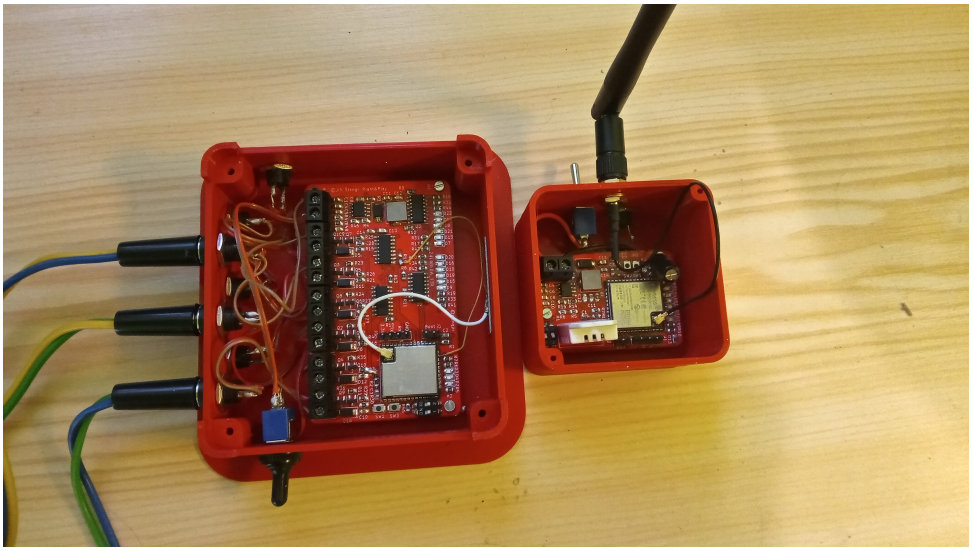
Nejtěžším krokem celé práce byla jednoznačně implementace firmware pro moduly řadiče ventilů a gateway. Firmware pro ostatní moduly jsem po dohodě s vedoucím práce nedělal. Firmware jsem implementoval v MicroPythonu, což je implementace programovacího jazyka Python pro mikrořadiče. Zpětně hodnotím toto rozhodnutí jako chybné, protože zejména firmware pro gateway je poměrně rozsáhlý a je problém, aby se na čip vůbec vešel a bylo tak potřeba optimalizovat kód, aby byl menší. Jednalo se zejména o spojení dto, packetů a vyjímek do jednoho souboru, aby se snížil overhead při importaci, dále pak bylo potřeba importovat pouze lokálně a velmi často volat garbage collector. I přes tato omezení kód pro gateway funguje, ale bohužel utrpěla jeho kvalita. Dalším potenciálním problémem MicroPythonu je fragmentace operační paměti, která víceméně znemožňuje alokování větších souvislých bloků dat.

V neposlední řadě bylo potřeba celé řešení otestovat. Zde jsem využil možnosti fakultního gitlabu a pujčil jsem si fakultní gitlab runnery pro stavbu a testování backendu webové aplikace. Tímto jsem zajistil kvalitu poskytovaného rozhraní a jeho konzistenci. Dále jsem pak provedl testovací scénář, který prošel bez chyby.

Pro gateway i pro řadič ventilů jsem navrhl v programu Autodesk Fusion 360 a vytiskl na 3D tiskárně krabičky. Krabičky jsou vytisknuty z materiálu PLA. Navržené 3D modely lze najít v příloze práce.



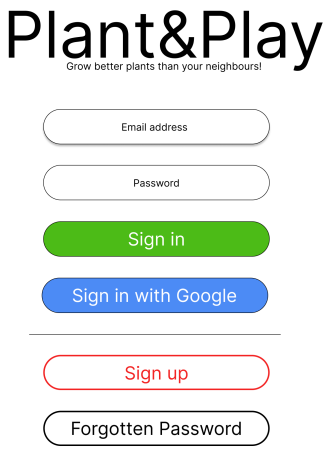
Obrázek 14.1: Fotografie hotové gateway (menší krabička) a řadiče ventilů s připojenými ventily



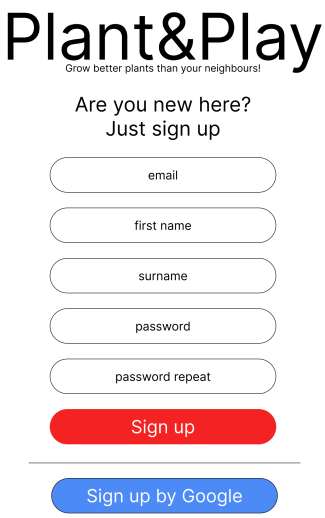
Obrázek 14.2: Fotografie hotové gateway (menší krabička) a řadiče ventilů s odhaleným vnitřkem

Příloha A

Wireframy webové aplikace

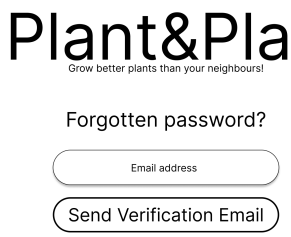


Obrázek A.1: Wireframe úvodní stránky pro uživatele na přihlášení

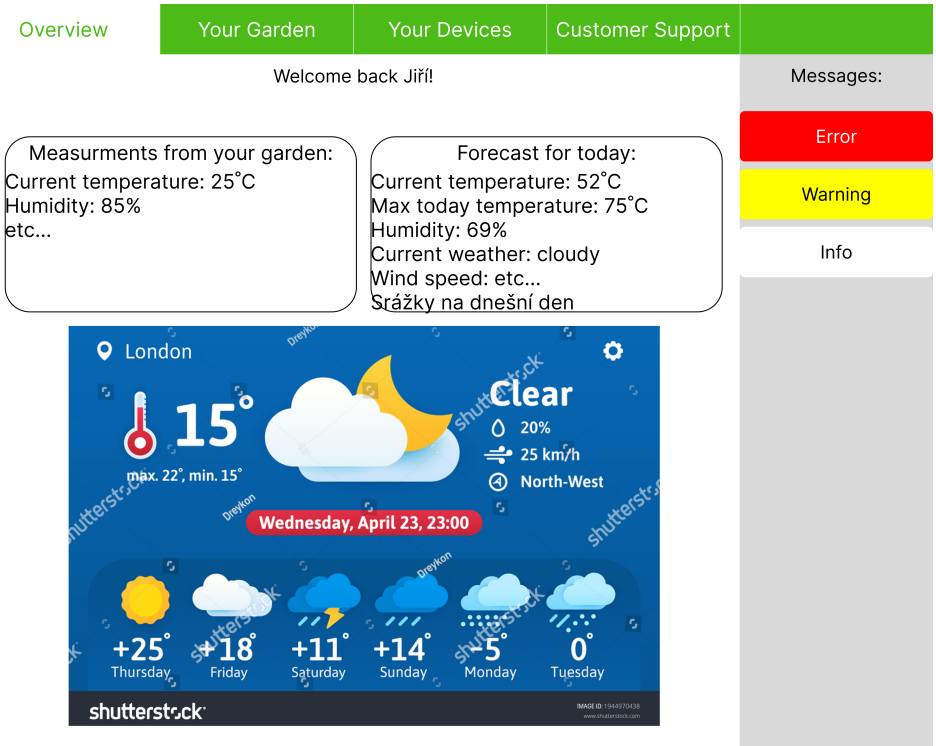


Obrázek A.2: Wireframe registrační stránky pro uživatele

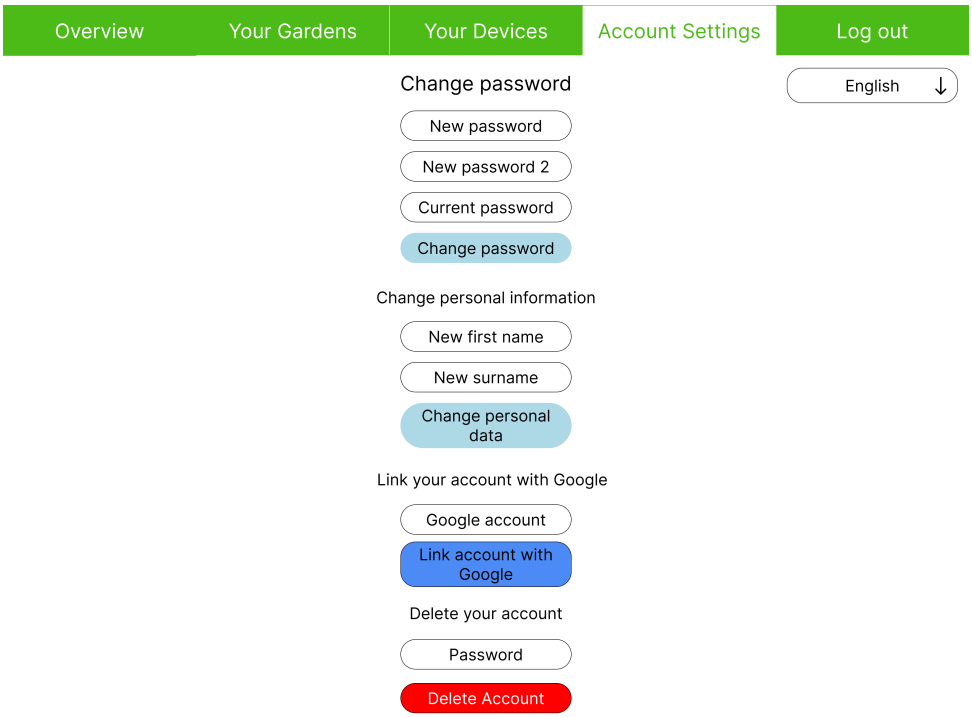
Obrázek A.3: Wireframe prvního kroku při zapomenutí hesla



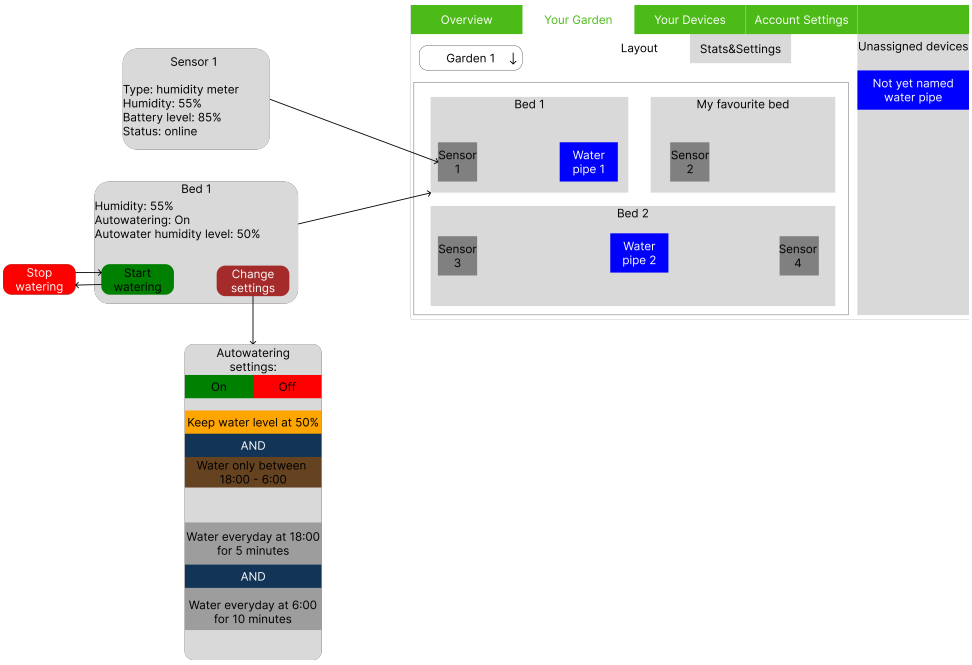
Obrázek A.4: Wireframe druhého kroku při zapomenutí hesla



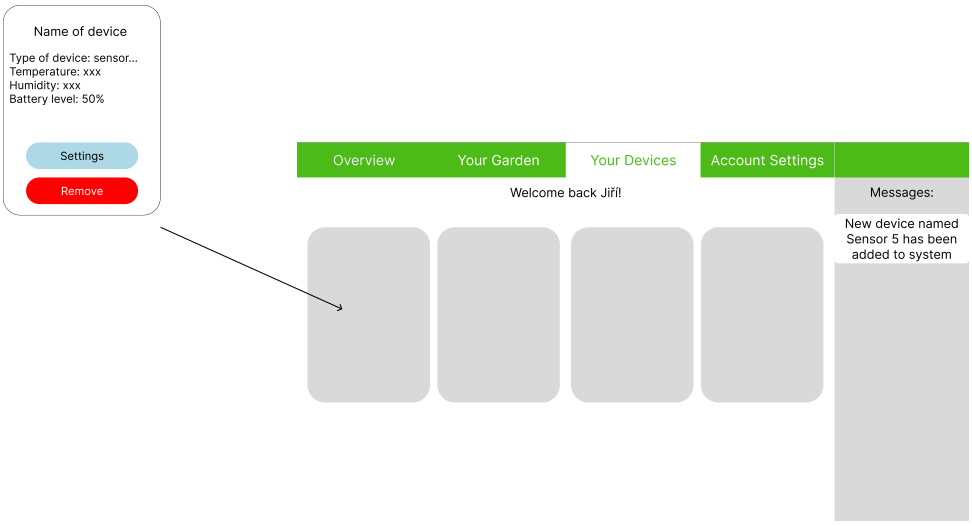
Obrázek A.5: Wireframe přehledu pro uživatele při přihlášení



Obrázek A.6: Wireframe nastavení účtu v aplikaci



Obrázek A.7: Wireframe rozložení zahrady



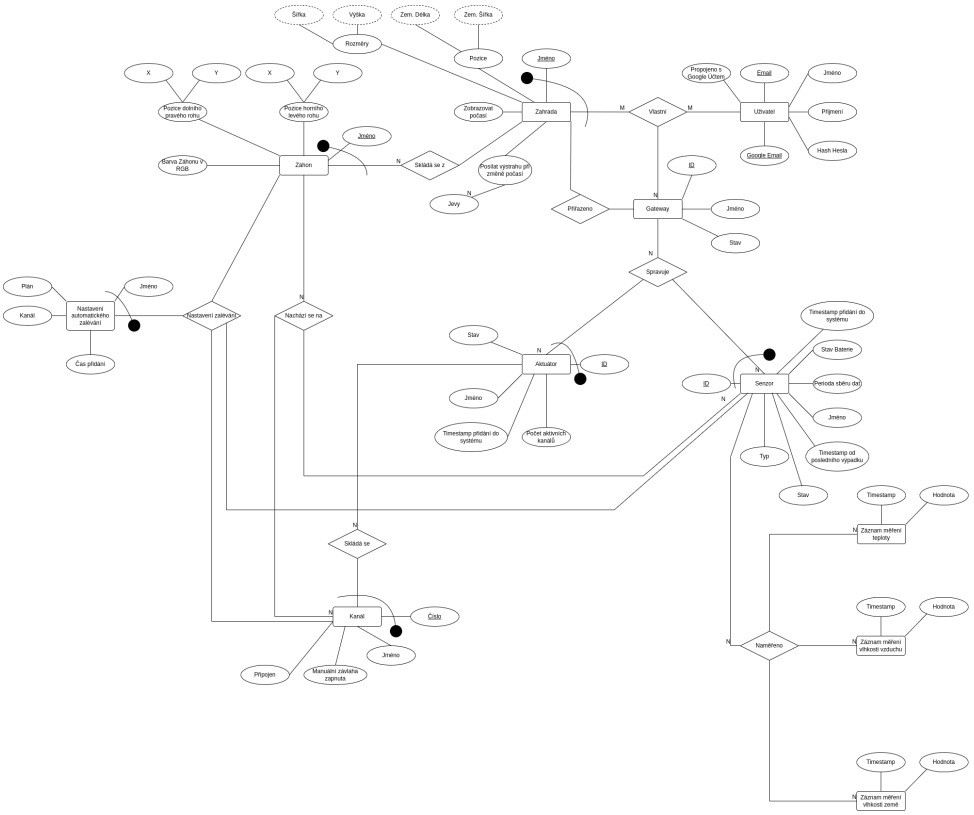
Obrázek A.8: Wireframe nastavení zařízení

Obrázek A.9: Wireframe pro nastavení vlastností zahrady

Obrázek A.10: Wireframe pro odhlášení uživatele

Příloha B

Databázový model



Obrázek B.1: Konceptuální databázový model webového infomačního systému

Příloha C

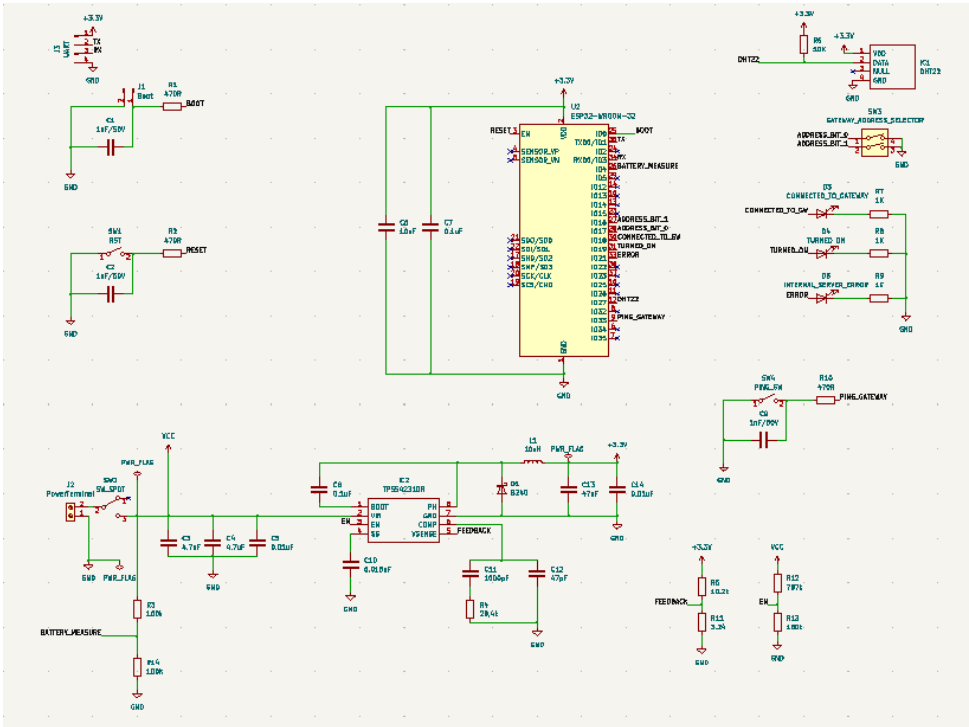
Schémata modulů a seznamy součástek potřebných k jejich osazení

C.1 Čidlo měření teploty

C.1.1 Seznam součástek

- 1x ESP-WROOM-32
- 1x Svorkovnice
- 1x Posuvný mikrospínač
- 1x TPS54231DR
- 1x B240
- 1x 10µH cívka
- 1x DHT22
- 1x 2-bitový přepínač
- 2x mikrospínač
- 1x 2 kolíková lišta
- 1x 4 kolíková lišta
- 3x LED
- Rezistory s následujícími hodnotami: 3x 1kΩ, 3x 470Ω, 2x 100kΩ, 1x 29.4kΩ, 1x 10.2kΩ, 1x 3.24kΩ, 1x 787kΩ, 1x 180kΩ, 1x 10kΩ
- Kondenzátory s následujícími hodnotami: 2x 4.7µF, 3x 10nF, 1x 15nF, 1x 100nF, 4x 1nF, 1x 47µF, 1x 47pF, 1x 10µF

C.1.2 Schéma



Obrázek C.1: Schéma čidla teploty

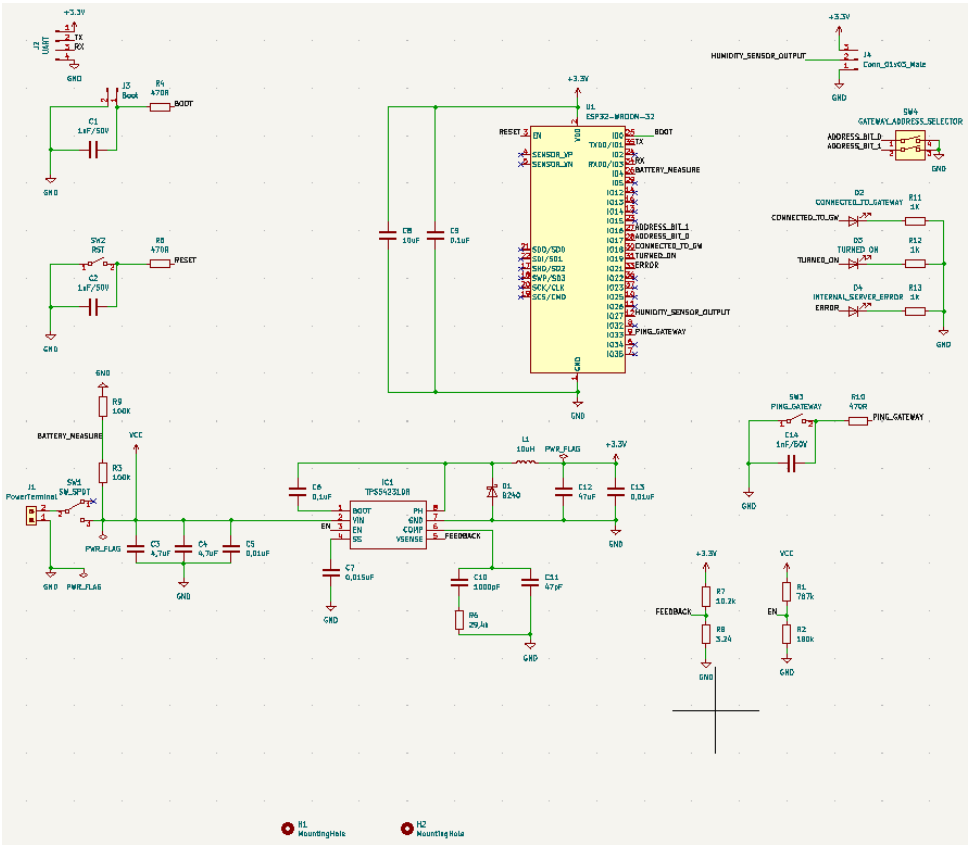
C.2 Čidlo měření vlhkosti

C.2.1 Seznam součástek

- 1x ESP-WROOM-32
- 1x Svorkovnice
- 1x Posuvný mikrospínač
- 1x TPS54231DR
- 1x B240
- 1x 10µH cívka
- 1x 2-bitový přepínač
- 2x mikrospínač
- 1x 2 kolíková lišta
- 1x 3 kolíková lišta
- 1x 4 kolíková lišta
- 3x LED

- Rezistory s následujícími hodnotami: 3x 1kΩ, 3x 470Ω, 2x 100kΩ, 1x 29.4kΩ, 1x 10.2kΩ, 1x 3.24kΩ, 1x 787kΩ, 1x 180kΩ
- Kondenzátory s následujícími hodnotami: 2x 4.7µF, 3x 10nF, 1x 15nF, 1x 100nF, 4x 1nF, 1x 47µF, 1x 47pF, 1x 10µF

C.2.2 Schéma



Obrázek C.2: Schéma čidla vlhkosti

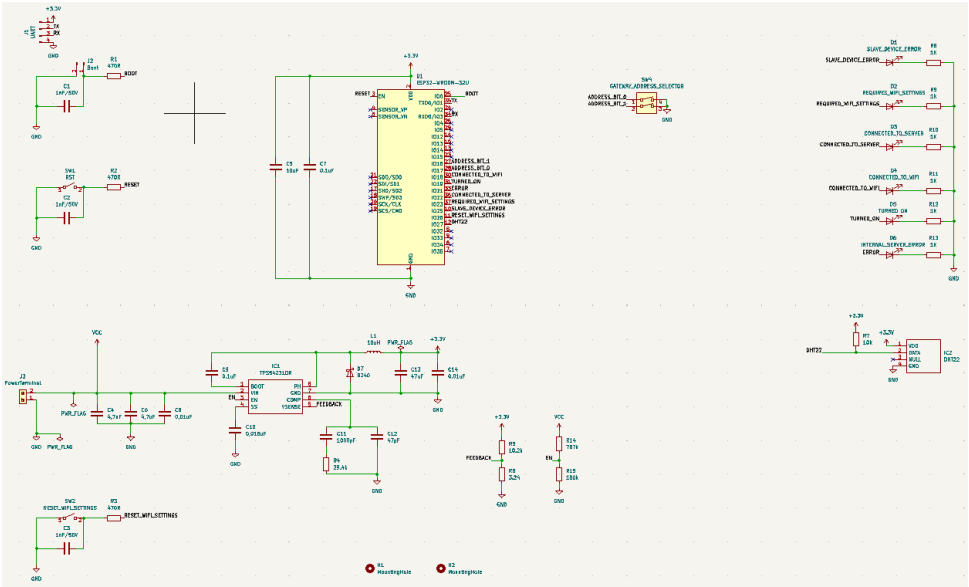
C.3 Gateway

C.3.1 Seznam součástek

- 1x ESP-WROOM-32U
- 1x Svorkovnice
- 1x Posuvný mikrospínač
- 1x TPS54231DR

- 1x B240
- 1x 10µH cívka
- 1x DHT22
- 1x 2-bitový přepínač
- 3x mikrospínač
- 1x 2 kolíková lišta
- 1x 4 kolíková lišta
- 6x LED
- Rezistory s následujícími hodnotami: 6x 1kΩ, 4x 470Ω, 2x 100kΩ, 1x 29.4kΩ, 1x 10.2kΩ, 1x 3.24kΩ, 1x 787kΩ, 1x 180kΩ, 1x 10kΩ
- Kondenzátory s následujícími hodnotami: 2x 4.7µF, 3x 10nF, 1x 15nF, 1x 100nF, 5x 1nF, 1x 47µF 1x 47pF, 1x 10µF

C.3.2 Schéma



Obrázek C.3: Schéma gateway

C.4 Řídící jednotka aktuátorů/ventilů

C.4.1 Seznam součástek

P1. 1x ESP-WROOM-32U

P2. 7x Svorkovnice

P3. 1x TPS54231DR

P4. 1x B240

P5. 1x 10µH cívka

P6. 1x 2-bitový přepínač

P7. 3x mikrospínač

P8. 1x 2 kolíková lišta

P9. 1x 4 kolíková lišta

P10. 15x LED

P11. 6x 2N7002

P12. 6x DMG2302U

P13. 6x MRA4003T3G

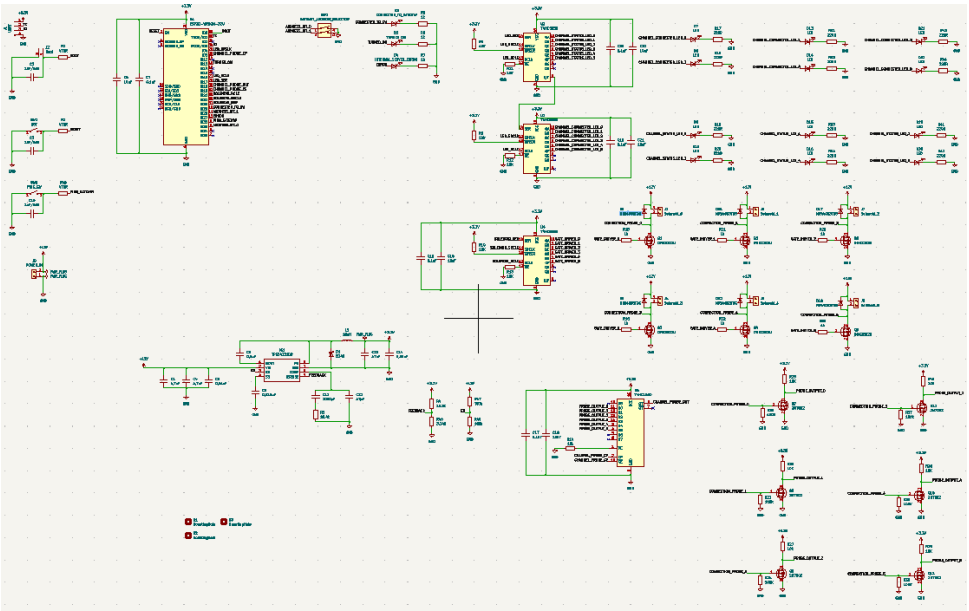
P14. 1x 74HC165D

P15. 3x 74HC595D

P16. Rezistory s následujícími hodnotami: 9x 1kΩ, 3x 470Ω, 2x 100kΩ, 1x 29.4kΩ, 1x 10.2kΩ, 1x 3.24kΩ, 1x 787kΩ, 1x 180kΩ, 1x 10kΩ, 12x 220Ω, 6x 100kΩ, 11x 10kΩ

P17. Kondenzátory s následujícími hodnotami: 2x 4.7µF, 7x 10nF, 1x 15nF, 1x 100nF, 4x 1nF, 1x 47µF, 1x 47pF, 5x 10µF

C.4.2 Schéma



Obrázek C.4: Schéma řídící jednotky aktuátorů

C.5 Použité obrázky třetích stran

C.5.1 Ikonky

Obrázek C.5: Ikonka pro značení informační zprávy ve webové službě. Odkaz na autora

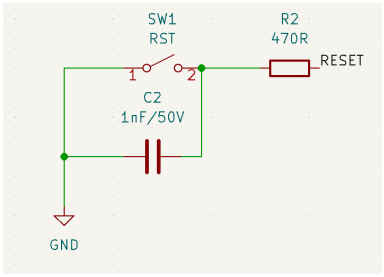
Obrázek C.6: Ikonka pro značení varovné zprávy ve webové službě. Odkaz na autora

Obrázek C.7: Ikonka pro značení errorové zprávy ve webové službě. Odkaz na autora

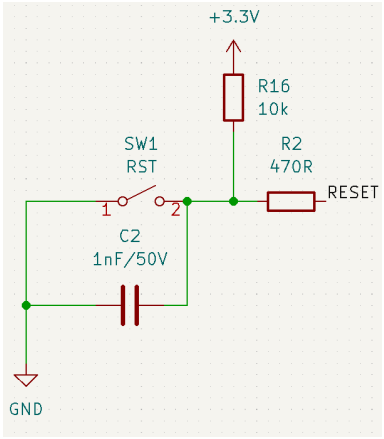
Příloha D

Nalezené hardwarové chyby

D.1 Chybějící pull-up rezistor mezi resetovacím pinem a napájecím napětím

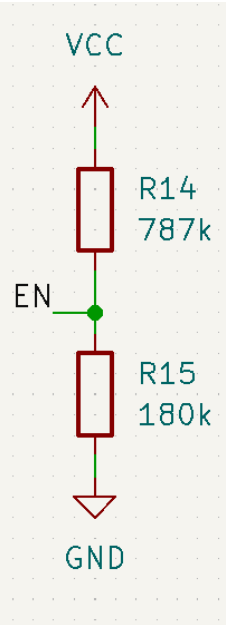


Obrázek D.1: Aktuální schéma gateway s chybějícím pull up rezistorem

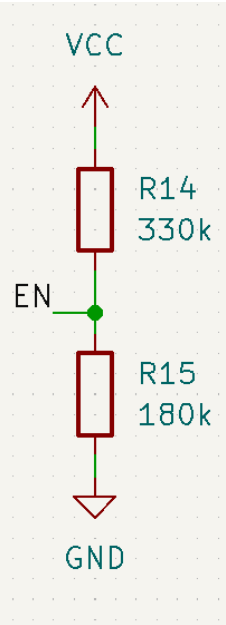


Obrázek D.2: Opravené schéma gateway s chybějícím pull up rezistorem

D.2 Špatně zvolená velikost rezistoru, nastavujícího zapínací práh regulátoru

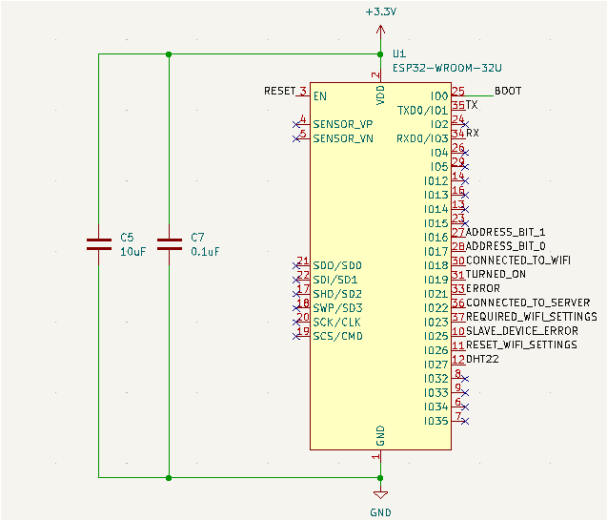


Obrázek D.3: Špatně zvolený rezistor R14, který umožňoval desky napájet pouze napětím vyšším jak 7V

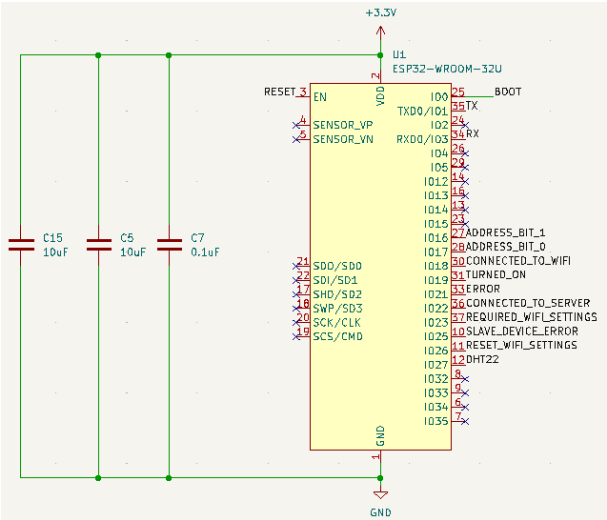


Obrázek D.4: Nižší hodnota rezistoru R14, který nyní umožňuje napájení desky napětím rovným nebo vyšším než 4V

D.3 Doplnění filtrovacího kondenzátoru k modulu ESP-32

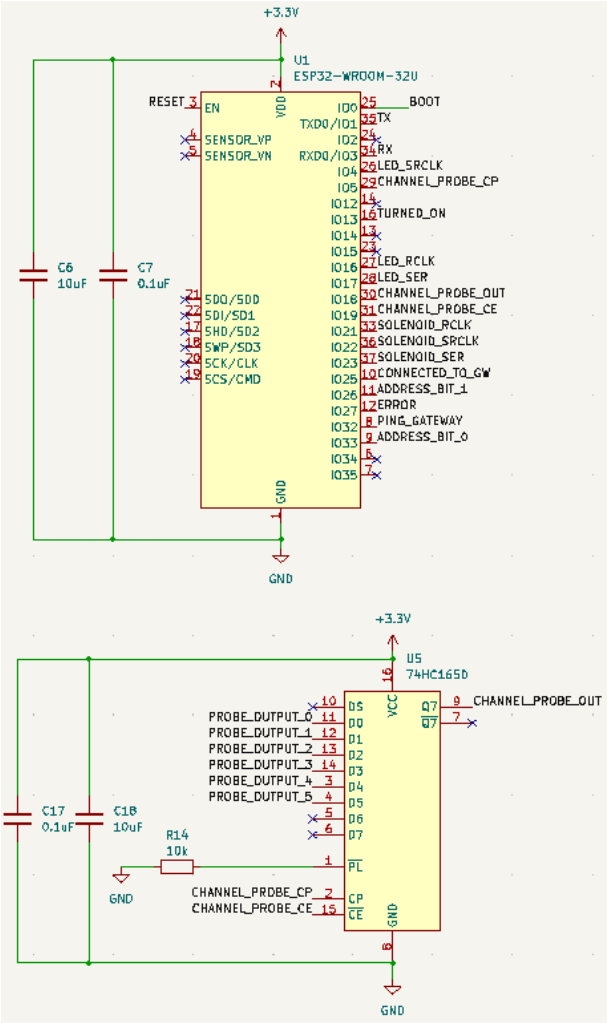


Obrázek D.5: Modul ESP-32 na desce Gateway pouze se 2 kondenzátory doporučenými od výrobce

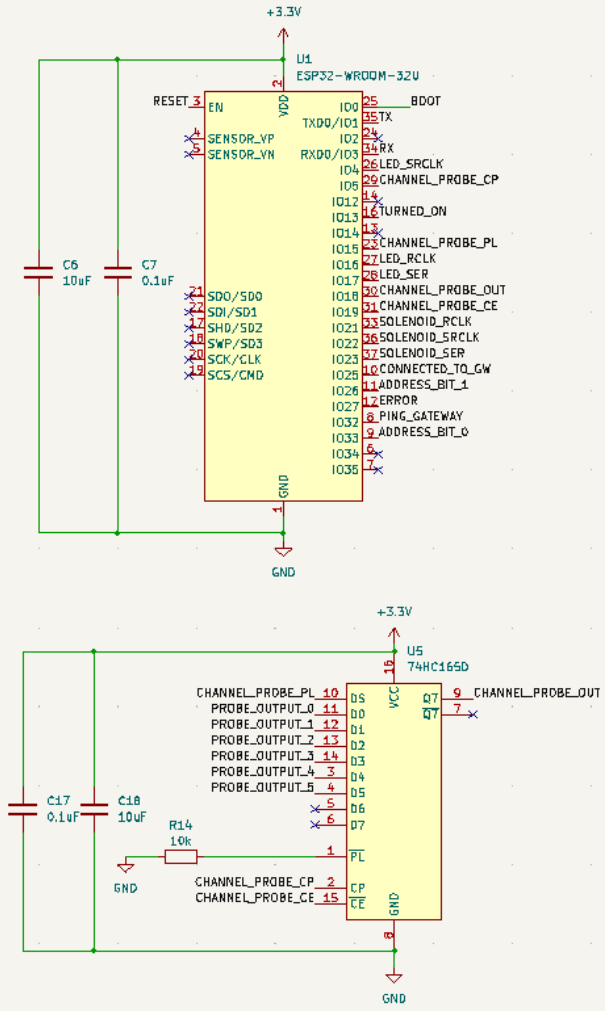


Obrázek D.6: Modul ESP-32 na desce Gateway doplněný o 3. filtrovací kondenzátor

D.4 Doplnění vodiče mezi ESP32 a pinem PL na čipu SN74HC165



Obrázek D.7: Deska řadiče ventilů bez propojení mezi GPIO 15 a pinem DS na čipu SN74HC165



Obrázek D.8: Deska řadiče ventilů s přidaným propojením vyřešeným pomocí jmenného štítku

Příloha E

Zdrojové kódy, návrhy desek a 3D modely

Všechny zdrojové kódy lze najít v repozitáři na fakultním gitlabu.

E.1 Struktura skupiny zahrada na gitlabu

Tato sekce uvadí strukturu skupiny na gitlabu obsahující všechny zdrojové kódy, návrhy desek, 3D modely a dokumentaci celého systému v podobě této práce. Struktura popisuje strukturu skupiny na fakultním gitlabu.

- 3D models - podskupina obsahující 3D modely uložené ve formátu stl
  - ValveDriverBoxModels - repozitář obsahující 3D modely krabičky řadiče ventilů uložené ve formátu stl
  - GatewayBoxModels - repozitář obsahující 3D modely krabičky gateway uložené ve formátu stl
- Firmware - podskupina obsahující firmware pro gateway a řadič ventilů
  - Gateway - repozitář obsahující firmware pro gateway implementovaný v MicroPythonu
  - ValveDriver - repozitář obsahující firmware pro řadič ventilů implementovaný v MicroPythonu
- PlantAndPlayBackend - repozitář s backendem webové služby implementovaný v Javě 17 ve frameworku SpringBoot 2.7.5. Repozitář obsahuje funkční pipeline pro testování a stavění projektu
- FrontEnd - repozitář obsahující implementaci frontend části webové služby ve frameworku React v jazyce Typescript
- Actuator - repozitář obsahující navržené schéma a desku plošného spoje pro řadič ventilů v programu KiCad 6.0
- Gateway - repozitář obsahující navržené schéma a desku plošného spoje pro gateway v programu KiCad 6.0
- MoistureSensor - repozitář obsahující navržené schéma a desku plošného spoje pro čidlo vlhkosti půdy v programu KiCad 6.0

- TemperatureSensor - repozitář obsahující navržené schéma a desku plošného spoje pro čidlo vlhkosti a teploty vzduchu v programu KiCad 6.0
- BachelorThesisDocumentation - repozitář obsahující dokumentaci projektu ve formě této práce

Příloha F

Literatura

[1] Gardena micro drip systém. https://www.gardena.com/cz/produkty/zavlaha/micro-drip/. Accessed: 29.4.2023.

[2] Gardena počítače pro řízení závlahy. https://www.gardena.com/cz/produkty/zavlaha/rizeni-zavlazovanie/. Accessed: 29.4.2023.

[3] Immax neo lite smart zavlažovací systém. https://www.immax.cz/immax-neo-lite-smart-zavlazovaci-system-dvojity-wifi-p13980/?utm_source=Google+n%C3%A1kupy&utm_medium=ppc&utm_campaign=Immax+NEO+LITE+Smart+zavla%C5%BEovac%C3%AD+syst%C3%A9m,+dvojit%C3%BD,+WiFi&gclid=Cj0KCQjwgLOiBhC7ARIsAIeetVBrmrvw7utc8GPZHkZCE7OW6YZNIZaScX1rNJtZNe1C5xCjHH8KlbsaAlTEwcB. Accessed: 29.4.2023.

[4] Micropython. https://micropython.org/. Accessed: 7.5.2023.

[5] Orbit 57946. https://www.amazon.com/Orbit-57946-6-Station-Sprinkler-Controller/dp/B01D15HOIQ/ref=sr_1_2_sspa?keywords=smart%2Birrigation%2Bsystem&qid=1683391056&sprefix=smart%2Birigation%2Caps%2C198&sr=8-2-spons&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUFUQ1VST1BGRkZRTk4mZW5jcnlwdGVkSWQ9QTAxNzU3OTgxRzNGWjBCV0sC&th=1. Accessed: 6.5.2023.

[6] Rachio 3. https://www.amazon.com/Rachio-Smart-Sprinkler-Controller-8-Zone/dp/B07CZ864Y9. Accessed: 6.5.2023.

[7] Rostlinné komodity. https://eagri.cz/public/web/mze/zemedelstvi/rostlinna-vyroba/rostlinne-komodity.

[8] Zemědělská výroba. https://eagri.cz/public/web/mze/zemedelstvi/zemedelstvi.html.

[9] *Introduction to Software Architecture*, pages 1–33. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[10] J. Canete, Cipriano Galindo, and Inmaculada Moral. *Introduction to Control Systems*, pages 137–165. 01 2011.

[11] G. Carl, G. Kesidis, R.R. Brooks, and Suresh Rai. Denial-of-service attack-detection techniques. *IEEE Internet Computing*, 10(1):82–89, 2006.

[12] George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems*. International Computer Science Series. Addison-Wesley Educational, Boston, MA, 4 edition, June 2005.

[13] Bohumír Dovalil. *Chov ryb v akvaponickém systému*. PhD thesis, 2014.

[14] Michal Jakob. Úvod do distribuovaných systémů, Jan 2021.

[15] Michal Jakob. Čas a kauzalita v ds, Jan 2021.

[16] Merle H. Jensen. *Hydroponics*, volume 32, page 1018–1021. Department of Plant Sciences, University of Arizona, 1997.

[17] Michal Jákob. Vzájemné vyloučení procesů, Jan 2020.

[18] Avijit Mallik. Man-in-the-middle-attack: Understanding in simple words. *Cyberspace: Jurnal Pendidikan Teknologi Informasi*, 2(2):109–134, 2019.

[19] Soumyalatha Naveen. Study of iot: Understanding iot architecture, applications, issues and challenges. 05 2016.

[20] Ameya Nayak, Anil Poriya, and Dikshay Poojary. Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems*, 5(4):16–19, 2013.

[21] Michael Predko. *Digitální elektronika bez předchozích znalostí*. Computer Press, 2008.

[22] Sharvari Rautmare and D. M. Bhalerao. Mysql and nosql database comparison for iot application. In *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, pages 235–238, 2016.

[23] Ian J.H Reynolds. Iot architecture: 3 layers, 4 stages explained, May 2021.

[24] Karen Rose, Scott Eldridge, and Lyman Chapin. The internet of things: An overview. *The internet society (ISOC)*, 80:1–50, 2015.

[25] Durgesh Samadhiya, Su-Hua Wang, and Dengjie Chen. Quality models: Role and value in software engineering. In *2010 2nd International Conference on Software Technology and Engineering*, volume 1, pages V1–320–V1–324, 2010.

[26] Martinus Richardus van Steen and Andrew S Tanenbaum. *Distributed Systems*. 2017.

[27] M. Vošta. Společná zemědělská politika eu a její aplikace v České republice. pages 127–142, 2 2010.

[28] Chang-Le Zhong, Zhen Zhu, and Ren-Gen Huang. Study on the iot architecture and gateway technology. In *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 196–199, 2015.