ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra telekomunikační techniky

Bakalářská práce

Užitný a technický rozbor inteligentního zařízení domácí lékovky

Jakub Jíra
Studijní program: Otevřená Informatika
Obor: Internet věcí

Květen 2021
Vedoucí práce: doc. Ing. Leoš Boháč,Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jíra** Jméno: **Jakub** Osobní číslo: **483779**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Otevřená informatika**

Specializace: **Internet věcí**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh inteligentního zařízení domácí lékovky

Název bakalářské práce anglicky:

Intelligent Pills Dispenser Design

Pokyny pro vypracování:

Pokyny CZ Proveďte návrh inteligentního domácího dávkovače léků, který by usnadnil chronicky nemocným snazší dávkování. Systém by měl pracovat v modální interakci s uživatelem tak, aby jej mohli využít i zrakově, či sluchově postižení lidé. Systém by měl upozornit uživatele na potřebu braní láků v daný čas, nutnost zásobení léků a měl by podporovat další funkce, které identifikuje v rámci první části bakalářské práce student. Systém by měl integrovat nezbytnou elektroniku a mechaniku dávkování, včetně případného napojení bezdrátovými komunikačním prostředky do centrálního systému pro správu léků a případně i jejich objednání přes lékaře apod. Tyto funkce student zmapuje a dle možností v práci implementuje.

Seznam doporučené literatury:

[1] Sinclair I, Dunton J.: Practical Electronics Handbook. Jordan Hill: Elsevier Science & Technology; 2006.
[2] Schwartz M.: Arduino Networking. Olton: Packt Publishing, Limited; 2014.
[3] Olsson: Arduino Wearable Projects. Birmingham: Packt Publishing, Limited; 2015.
[4] Bayle J.: C Programming for Arduino. Olton: Packt Publishing, Limited; 2013.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Leoš Boháč, Ph.D., katedra telekomunikační techniky FEL

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

Datum zadání bakalářské práce: **22.01.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce:
do konce letního semestru 2021/2022

doc. Ing. Leoš Boháč, 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

Poděkování / Prohlášení

Děkuji

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 21. května 2021

.....

Abstrakt / Abstract

Cílem této práce bylo provést analýzu a návrh inteligentního dávkovače léků, který usnadní chronicky nemocným snazší dávkování. Navržené zařízení je připraveno na uzpůsobení specifickým požadavkům uživatelů. Systém je schopen upozornit uživatele na braní léků a monitorovat zacházení se zařízením. Tyto data je schopno pomocí Azure IoT Hub odeslat do aplikace, ze které je schopné příkazy i přijímat a následně se jimi řídit. V práci se nepovedlo vytvořit minimální životaschopný produkt, avšak se povedlo vytvořit konceptuální zařízení na základě kterého se dá následující produkt vystavět.

Klíčová slova: Internet věcí, ESP32, Microsoft Azure, návrh vestavných systémů.

The goal of this project was to analyze and design an inteligent drug dispenser to facilitate easier dosing for chronically ill patients. The designed device is ready to be adapted to the specific requirements of the users. The system is capable of alerting the user to take medication as well as monitoring the usage of the device. The device is able to send this data to the application via Azure IoT Hub, from which it is also able to receive commands and then execute them. The thesis has not succeeded in creating a minimum viable product, but has been able to create a conceptual device upon which the following product may be built.

Keywords: Internet of things, ESP32, Microsoft Azure, embeded system design.

Title translation: Intelligent Pills Dispenser Design

Obsah /

/ Obrázky

Kapitola 1
Úvod

1.1 Hlavní myšlenka produktu

Motivací ke vzniku této práce je využití moderních technologií a trendu IoT k usnadnění života lidem, kteří potřebují asistenci při denních úlohách. IoT zařízení jsou na trhu již delší dobu, avšak jejich momentální problém je ten, že v odvětvích, ve kterých by se dala využívat, a byla přínosem pro danou práci, je jejich implementace často obtížná.

Lidé v těchto odvětvích se nechtějí starat o daná zařízení. Nechtějí řešit problémy s nimi spojená a jejich správu. Tímto odvětvím je i prostor sociálních služeb. Sociální pracovníci, jakožto humanitně vzdělaní lidé, mají častokrát minimální zkušenosti s moderními technologiemi a, z jejich pohledu, minimální motivaci se s nimi naučit zacházet.

Dalším cílem jsou senioři, kteří jsou také spíše skeptičtí vůči dané tématice. Avšak této skupině by právě chytrá zařízení mohla pomoci nejvíce. Ať s monitorováním či připomínáním jednotlivých úkolů, tak s případným přivoláním pomoci.

Tato projektová studie se však nechce zaměřit jen na seniory, ale mít na paměti i na pacienty se zrakovou, sluchovou, či jinou indispozicí. Cílem produktu je tedy co největší modularita, která by umožnila zařízení co nejjednodušeji přizpůsobit danému pacientovi.

Výsledkem implementace této studie by měl být produkt schopen pomoci výše uvedeným lidem (seniorům, chronicky postiženým) s užíváním léků. Většina seniorů je již na obyčejnou lékovku zvyklá a bylo by vhodné z obyčejné lékovky co nejvíce vycházet. Mít zařízení, které má přihrádky, do kterých se dají vložit různá léčiva, ze kterých si je subjekt bude brát na základě předpisu. Cílem je rozhodně umožnit, aby zařízení mohlo zůstat navenek co *nejhloupější*. Aby senior, pokud nechce, měl zařízení, na které byl doteď zvyklý a se kterým je spokojený.

Zařízení by však nemělo zůstat hloupé. Mělo by být schopné, pokud se to po něm vyžaduje, vykonat následující funkce:

- Uživateli připomene (zvukově i vizuálně), že nastal čas některý z medikamentů užít.
- Uživatel je sám schopen nastavit upozornění a přiřadit danému upozornění vlastní přihrádku, případně přihrádky.
- Zařízení je schopné takto zadaná data zobrazit uživateli.
- Zařízení je schopné v případě výpadku energie přejít na vnitřní baterii.
- Zařízení je schopné operovat jak v offline, tak v online režimu. V obou režimech je schopno logovat stav. Tyto logy v případě připojení na internet může poskytnout pověřené osobě.
- Zařízení je schopné zamknout a odemknout určité přihrádky.
- Zařízení je schopné detekovat otevření/zavření určité přihrádky.
- Zařízení je schopné přijímat data ze sítě. Na základě jich se dají určit časy upozornění a jednotlivá nastavení vzdáleně.
- Zařízení je schopné přehrát zvukové zprávy.

Toto jsou funkcionality, které by dané zařízení mělo zvládat jakožto základní. Zařízení by také mělo umožňovat implementaci dalších modulů, které umožní celému systému být ohebnější. Díky tomuto přístupu bude jednodušší se více přizpůsobit unikátním požadavkům jednotlivých potencionálních uživatelů.

1.2 Motivace pro vytvoření takovéhoto produktu

Motivací pro tento projekt je samozřejmě pomoci lidem. To samotné je sice hezký záměr, avšak tento produkt nezávisí na samotném uživateli, neboť se počítá s tím, že uživatel je nějak znevýhodněn. Zařízení by musela pořídit osoba starající se o daného člověka. Systém může být sebelepší, ale pokud se těmto lidem nebude zamlouvat, bude velice těžké daný produkt dostat ke koncovému zákazníkovi.

Na základě této otázky jsem kontaktoval sociální pracovnici pracující se seniory. Produkt popsat a otázal se, zda má daný koncept pro ni smysl, či zda ji napadají některé funkce, které by zařízení mělo rozhodně mít.

Reakcí mi bylo, co jsem zhruba čekal. Zařízení by smysl mělo, za správnou cenu, samozřejmě. A důležitým pilířem je co největší jednoduchost pro finálního uživatele. Jak pro případného seniora, tak pro člověka, který bude zařízení spravovat. Dalším problémem je jednoduché a rychlé dávkování, které bude pravděpodobně provádět sociální pracovník starající se o případného pacienta.

Můj vzorek je velice limitovaný a sériové výrobě by jistě muselo předcházet mnohem rozsáhlejší šetření. Tento můj krok měl však za úkol otestovat zda člověku z praxe přijde takovýto produkt přijatelný, jestli je po něj prostor a nějaký vhled do celé věci, který je při vývoji jakéhokoliv produktu velice důležitý.

1.3 Průzkum trhu

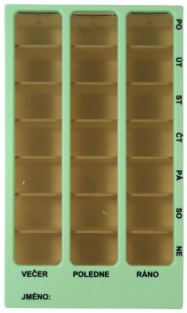
Na začátku každého produktu musí nastat neodmyslitelně fáze průzkumu trhu. Je-li pro nový produkt místo, poptávka, či jestli se daný produkt již na scéně nenachází. Momentálně se na trhu nachází následující produkty:

1.3.1 Obyčejná lékovka

Lékovka jednoduchá, ale funkční. Používaná širokou veřejností pro její cenu a jednoduché užívání. Takovéto lékovky primárně zjednodušují organizaci léků.

Lékovka Benu [1]

Dávkovač na léky kitos [2]



Dávkovač léků OBZOR [3]

Obrázek 1.1. obyčejné lékovky běžně dostupné na trhu.

1.3.2 Obyčejná lékovka s vylepšením

Do této kategorie se primárně řadí dávkovače léků, které mají v sobě zakomponovaný časovač s alarmem. Umožní uživateli nastavit časy, ve které chce býti upozorněn. Mimo toto vylepšení někdy přidávají další funkcionalitu v podobě například měření tepu.

Lékovka KNL09 [4]

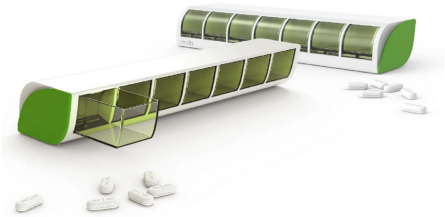
Lékovka ISO [5]

Inteligentní krabička Pilly [6]

Obrázek 1.2. obyčejné lékovky s vylepšením běžně dostupné na trhu.

1.3.3 Lékovky s bezdrátovým připojením

Tyto dávkovače jsou již blíže mému konceptu. Mají obvykle ve výbavě bluetooth, kterým se připojí k telefonu, se kterým za pomoci proprietární aplikace komunikují. Uživateli pomocí notifikací připomínají, zda nastal čas na konzumaci léčiva. Tato závislost na dalším uživatelově zařízení je něco, čemu by bylo vhodné se vyvarovat, neboť eliminace proprietárních zařízení usnadní netechnickým tipům snazší zacházení. Problém těchto lékovek je ten, že obvykle pochází z `fundraiser` stránek (kickstarter, indiegogo...) a jsou častokráte v evropě nedostupné a jejich dlouhodobá podpora není jistá.



Pillbox by Tricella [7]

Memo Box Vibrant [8]

Obrázek 1.3. Chytré lékovky na trhu.

1.3.4 Bezpečnostní lékovka

Tento tip dávkovače je obvykle kruhový s posuvným vnitřkem a motorem, který přihrádkami otáčí a umožní uživateli přístup vždy jen k jedné. Umožňuje funkce jako předešlé lékovky - zvuková, světelná upozornění a jako bonus znepřístupňuje přihrádky, které nemají být v daný čas přístupné. Výrobce uvádí i notifikaci autorizovaného uživatele, zprostředkované pomocí proprietárního Home8 cloud serveru.

home8alarm Medicaion dispenser [9]

Obrázek 1.4. Bezpečnostní lékovka.

1.4 Definice produktu

Z výše uvedených příkladů si můžeme zhruba udělat obraz stavu trhu. Bezpečnostních lékovek je na trhu málo a jsou zaměřované primárně na seniory, přičemž lidé jinak znevýhodněni, například zrakově budou mít s plněním těchto lékovek problém. Momentální produkty jsou pro jejich potřeby neoptimálně optimalizovány. Pokud by se měl mnou navrhovaný produkt něčím zásadním lišit bude to právě podmínění potřebám daného pacienta.

Potenciál tohoto produktu může být využitý i z druhé strany spektra. Pokud bychom si představili idealistický zmodernizovaný svět, mohla by tato nová lékovka být využita přímo firmou prodávající léky. Od doktorů by dostala informace, co jaký pacient potřebuje. Na základě toho by naplnila lékovky a dopravila uživatelům. Pomocí vzdáleného přístupu k lékovce by se mohly monitorovat návyky jednotlivých uživatelů, případně lékovky obměňovat v rámci nějaké `as a service` služby. Ušetřilo by se i na obalových materiálech.

Toto je samozřejmě velmi vizionářské a bylo by potřeba zvážit veškerá rizika spojená se sdílením osobních údajů, ale myslím si, že zde potenciál na podobné využití je.

Výsledkem by tedy měl být produkt který pomáhá uživateli s užíváním léků, dohlíží na správné užívání a je modulovatelný pro individuální přizpůsobení uživateli.

Kapitola 2
Možné přístupy řešení zadání

Jak již bylo předneseno, zařízení by mělo být schopné dávkovat a monitorovat uživatelovu spotřebu léků. Způsob řešení tohoto úskalí markantně ovlivní celou funkčnost zařízení, uživatelovo zacházení a celkový vzhled.

Různé přístupy se hodí pro různé uživatele a scénáře. Ve vybírání finálního přístupu jsem se rozhodl brát ohled na následující skutečnosti:

- Vyžadovaná angažovanost uživatele v celém procesu. Je nutné počítat s tím, že zařízení mohou používat i uživatelé s limitovanými motorickými schopnostmi, kterým se musí případně zařízení patřičně uzpůsobit.
- Styl dávkování léků. Má zařízení pouze určit, které léky se mají užít a to uživateli sdělit, či je má i sám patřičně nadávkovat. Případně pokud se jedná o přidělení času patřičné dávce, či přidělení léků určenému času.
- Nutnost zpřístupnění více přihrádek v jeden okamžik.
- Desinfekce produktu. Lékovka jakožto zdravotnický nástroj musí být dostatečně často dezinfikována, což musí umožnit i návrh zařízení.

Následující návrhy přistupují unikátně k problému a snaží se na dané podněty odpovědět.

2.1 Lékovka s přihrádkami

Tato lékovka vypadá jako krabička, ve které budou léky uloženy v malých přihrádkách podobné šuplíkům. Každý tento šuplík je plně vyndávací, což umožňuje snazší manipulaci a dávkování. Počet šuplíků závisí na stylu používání.

Pokud je zařízení používáno jakožto dávkovač, kde každý šuplík znázorňuje jeden den. Bylo by vhodné použít přihrádek sedm, a každému dnu v týdnu jednu přiřadit. Takovýto šuplík může být dále rozdělen na více segmentů, kde každý segment může znázorňovat jeden čas - ráno, poledne, večer - případně jiné rozdělení, podle potřeby uživatele.

Návrh šuplíku závisí na použití uživatelem. Pokud je preferováno, aby byl jeden šuplík jedna dávka - je možné nechat jejich vnitřní rozdělení nesegmentované. Ve chvíli, kdy by se jednalo o šuplík rozdělený na části, je důležité zajistit, aby byla vždy užita patřičná dávka. Zde tento přístup selhává neboť část užívání je dána do rukou uživatele a záleží, jak moc mu důvěřujeme.

Když jeden šuplík je vyžívaný jakožto jedna dávka léku, usnadní to uživateli manipulaci. Nebude nutné pro užití léků dostávat jednotlivé prášky z krabičky, stačí, když si je uživatel vyklepne. Tento přístup však s sebou nese problém s častým a komplexním dávkováním.

Tím, že šuplíky jsou odjímatelné, tak je velice snadná jejich údržba a případná desinfekce.

Zařízení může být ale použito i jakožto produkt, který uživatele upozorní a specifikuje dávku, avšak samotné dávkování nechá na uživateli. V tomto přístupu by měl každý

šuplík přiřazen jeden lék. Zařízení by pak ve chvíli, kdy by nastal čas užití medikamentů, upozornilo uživatele ze kterých přihrádek si má vzít kolik léků. Tyto přihrádky by i světelně, či jinak označilo a uživatele na nesprávnou manipulaci upozornilo. Jednotlivá dávkování by byla uložena v zařízení a na uživateli by bylo akorát léky, které jsou v dávce obsaženy, vytáhnout z přihrádky a užít. Tento přístup vyžaduje větší interakci uživatele, ale zařízení je mnohem flexibilnější a umožňuje více se přizpůsobit uživateli. Lékovka by mohla být využívána i na uložení léků, které se neberou pravidelně - například léky na srážení teploty. Zároveň by poté mohla lékovka hlídat, zda byla mezi dávkami těchto nepravidelných léků dodržena patřičná prodleva.

K omezení přístupu uživatele k lékům, které momentálně nemají být užity by mohly být použity solenoidy. Každý šuplík by měl na zadní straně ouško, za které by bylo možné solenoid zachytit a tím limitovat uživateli přístup k dané dávce. V případě potřeby je pak možné solenoid uvolnit a tím i přístup k lékům. Typ solenoidu záleží na užití. Pokud by se mělo jednat o zařízení, které má být po většinu času zavřené, volba by padla na solenoidy, jejichž výchozí stav je zavřený. V případě, kdy by uzamykání měla být pouze vedlejší funkce, je lepší využít solenoidy, které jsou ve výchozím stavu zatažené.

Pro detekci otevřeného a zavřeného šuplíku by se využily magnetické senzory. Na konci šuplíku by byl umístěný magnet, a v krabičce senzor. V momentě, kdy by byla detekována přítomnost magnetu - šuplík by se považoval za zasunutý a solenoid by mohl daný šuplík zamknout. Touto cestou se dá efektivně monitorovat s jakými šuplíky a potažmo léky je manipulováno.

V obou dvou přístupech se však mění pouze vnitřní logika zařízení, jeho velikost v závislosti počtu přihrádek, případně vnitřní rozložení šuplíků. Podstata zařízení však zůstává stejná.

Tímto přístupem získáme velkou modulovatelnost a možnost přizpůsobení se uživateli. Šuplíky mohou býti použity na více než jeden účel. Podle přístupu také snadné doplňování, případně dávkování - podle toho, co preferujeme.

Nevýhodou však je nutná interakce koncového uživatele se zařízením. Šuplík je třeba vysunout a zase zasunout. V případě šuplíků použitých jakožto separátních léků je i na uživateli správné nadávkování.

2.2 Lékovka s krokovým motorem

Tento návrh vychází z bezpečnostní lékovky uvedené výše 1.4 v tomto dokumentu. To jest princip krabičky, která má zabudovaný krokový motor, jehož pohybem se zpřístupňují dané přihrádky.

Na krokový motor je připevněno víko, které svým tvarem umožňuje přístup pouze k jedné přihrádce a tím limituje uživatele od užití jiných léků. Obdobně jako na víko může být motor připojen přímo na přihrádky. Pokud nastane čas nové dávky, posunutím krycího víka se stane nová přihrádka aktuální a tím pádem dostupná pro uživatele.

Tímto přístupem se zjednodušuje celý proces dodání léku uživateli. Není možné, aby se uživatel v dávce spletl - vždy má k dispozici jen to, co k dispozici mít má. Tento striktní přístup může být však také limitující. Pokud by uživatel vynechal jednu dávku léků, není možné se k ní zpětně vrátit, bez komplexní detekce léků.

Limitace tohoto přístupu jsou také jeho největší silou. Zařízení jedná jednoduše, intuitivně a pro člověka přirozeně. Pokud je krycí kryt průhledný, má uživatel dobrou představu o následujících dávkách a nemusí nijak ze zařízení zjišťovat, co ho čeká a zda je potřeba nějaké léky doplnit.

2.3 Lékovka jakožto dávkovače/automatu

Tato lékovka funguje na principu toho, že uvnitř zařízení jsou uložené patřičné léky, které může na vyžádání, pokud je k tomu pravý čas, uživateli naservírovat. Zařízení funguje na principu dávkovače - uživatel nastaví ruku, či nádobu, do které chce léky, a zařízení pak již pomocí gravitace tyto léky do daného místa dopraví.

Hlavní výhodou je nulová angažovanost koncového uživatele. Po upozornění, že je čas si vzít léky, se pouze dostaví k zařízení, potvrdí že je přítomen a připraven vzít léky a následně dostane do ruky přesnou dávku, kterou může požít. Tento přístup by byl vhodný pro uživatele se zhoršenou motorikou, kterým by manipulace se zařízením mohla činit problémy.

K interní logice zařízení se může přistoupit dvěma způsoby.

Zařízení může mít v sobě přihrádky, v nichž budou předpřipravené dávky a tyto dávky bude v pravý čas dávat uživateli. Každá přihrádka bude odpovídat jednomu času braní léků. Tento přístup je velice jednoduchý, ale efektivní. Složitější je zde doplňování léků, které by musel provádět personál, neboť můžeme předpokládat, že člověk, který potřebuje léky přesně podat nebude schopen ani zařízení doplnit.

Druhý přístup je obdobný, avšak neoperuje s již připravenými dávkami, ale jednotlivými léky. Ty je schopen detekovat a uživateli nadávkovat. Takovéto zařízení je ideální jak z pohledu správce, který bude zařízení doplňovat, tak z pohledu uživatele. Proces detekce a vytvoření dávky je však velice komplexní.

V zařízení by se nacházely přihrádky, které obsahují unikátní typ léku. K nim by zde byla ještě jedna, která by byla připravená na podání uživateli. Zařízení by muselo být schopné detekovat, které léky už připravilo, případně jaký počet. Detekce, zda se jedná o správnou či špatnou dávku, se dá zařídit pomocí vah, avšak v případě špatné dávky je její náprava defacto nemožná.

Výhody tohoto přístupu činí bezpochyby minimální uživatelova interakce. Problémem je komplexní implementace.

2.4 Výběr implementace

Pro splnění požadavků tohoto projektu jsem se rozhodl pro první variantu - tedy krabičku se šuplíky.

Jedná se o nejlépe modulovatelné zařízení, které se dá snadno přizpůsobit škále uživatelů a jejich potřebám. Zároveň snadná detekce a případné dávkování, na kterém se bude případně podílet sám uživatel umožní snazší implementaci.

Kapitola 3
Technické řešení

3.1 Výpočetní Jádro

Na trhu se nachází velké množství čipů, které se dají v tomto projektu využít. Následuje přesný popis, jak jsem při výběru postupoval. V rámci tohoto projektu jsem uvažoval nad následujícími alternativami.

- STM32
- ESP8266
- ESP32

Nejprve jsem chtěl začít s platformou STM32. S tou jsem se setkal v několika předmětech a jsem s ní tedy dobře obeznámen. Požadavkem v našem projektu je však připojení k internetu. To by nebyl takový problém - STM nabízí produkty, které mají vestavěnou WiFi, či se dají připojit různé moduly, které tuto funkcionalitu umožní.

Vlastním STM32 čip, na kterém jsem chtěl implementaci uskutečnit. Problém je, že můj čip nemá WiFi funkcionalitu. Musel bych tedy dodat tuto schopnost externím zařízením, jako například ESP8266 čipem. Komunikace mezi těmito zařízeními se dá jednoduše zprostředkovat skrze UART. STM čip však v takovéto chvíli bude do velké míry redundantní, jelikož výpočty, které bude STM provádět, by se daly provádět již přímo na ESP čipu.

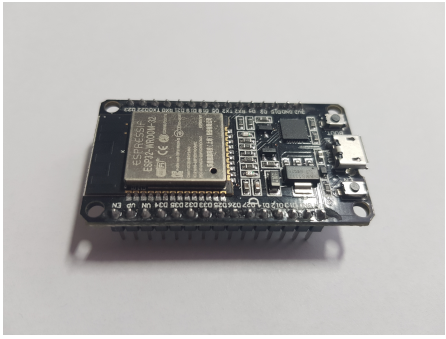
Tato cesta mě navedla k použití již zmíněného čipu ESP8266. Jedná se o jedno jádrový čip firmy ESPRESSIF, který by námi stanovené požadavky měl splňovat. Má vestavěnou WiFi, schopné výpočetní jádro a umožňuje široké spektrum programovacích jazyků, o tom více sekci o software v 4.1.

Rozhodl jsem se však proti ESP8266 na úkor jeho novějšího nástupce ESP32. Hlavní d;vody pro upgrade zde byly následující. ESP32 má 34 GPIO pinů, což je 2x více než ESP8266. Vzhledem k modulovatelnému přistupu je vhodné, aby zařízení bylo připravené na další potencionální periferie, což nám vyšší počet GPIO pinů umožní.

Další výhodou je vestavěné Bluetooth. Mnou navržená aplikace zatím nepočítá s implementací služby, která by ho využívala. Avšak jeho přítomnost umožňuje implementace uživatelských aplikací, které by byly více provázané s mobilním telefonem.

V neposlední řadě je také dobré zmínit, že ESP32 má ve stavu deep-sleep menší spotřebu (10 μA[10]) oproti ESP8266 (20 μA[11])

ESP32 má již také své novější nástupce v podobě ESP32-S2, či ESP32-C3. Pro naše účely si však myslím, že ESP32 bude naprosto uspokojivé a není důvod utrácet více peněz za jádro. Nemluvě o tom, že následující nástupci ESP32 mají také své nevýhody, například ESP32-S2 přišlo o bluetooth.



Obrázek 3.1. DOIT ESP32 devkit v1

K implementaci jsem tedy nakonec vybral ESP32. Konkrétně jsem vybral desku DOIT ESP32 devkit v1, která je osazena čipem ESP32-WROOM-32. Tato deska se použije pouze pro vývoj. Pro finální produkt by bylo více než vhodné vytvořit desku vlastní.

3.2 Periferie

Na základě splnění zadání je třeba připojit k zařízení periferie, které budou zprostředkovávat nejen interakci s uživatelem, ale také potřebné úkony s okolím.

3.2.1 Měření reálného času

Jeden z problému přenosného zařízení je udržení si časové informace i v případě, kdy nebude dostupné napájení.

ESP32 má vestavěné hodiny, které dobře udržují přesný čas - problém nastává v případě ztráty napájení, případně při prvním zapnutí. Pokud máme k dispozici WiFi - máme k dispozici NTP protokol, díky kterému můžeme získat přesný čas. V případě, že by se zařízení nacházelo v místech, kde je omezený WiFi signál, či by bylo vyžadováno plné offline zázemí. Je možné vybavit zařízení Real-time modulem, který si udržuje přesný čas i po výpadku napájení.

Možný modul, který by se na tento úkon dal použít je DS3231 RTC [12]. Ke komunikaci s tímto modulem se dá použít knihovna RTClib [13].

Obrázek 3.2. DS3231 Real-time modul [14]

Vzhledem k tomu, že zařízení je cíleno na užití ve WiFi síti, rozhodl jsem se tento modul nevyužít. Z důvodu snížení ceny produktu a zjednodušení základní implementace.

3.2.2 Uchování dat během ztráty napětí

Je třeba počítat se skutečností, že zařízení může ztratit napájení i WiFi signál. V tomto případě je naprosto nezbytné, aby zařízení bylo schopně znovu obnovit předchozí relaci. ESP32 je na takovéto situace připraveno a máme k dispozici dvě metody, kterými jsem schopen tížený výsledek dosáhnout.

První možností je využít EEPROM paměť. Jedná se o stabilní, mazatelnout paměť, na kterou můžeme zapisovat s využitím ESP32 knihovny EEPROM.h [15].

Paměť EEPROM má dvě nevýhody. Je relativně pomalá, což by naším požadavkům nijak nevadilo, ale hlavně je velice malá. Její objem v našem případě činí pouze 512B. Pro uložení holé informace - uložení upozornění na daný čas - by tato paměť stačila. Pokud bychom ale chtěli ukládat náročnější informace, jako jméno léku, paměť nám rychle dojde.

Druhou možností je využít SPIFFS. Jedná se o plný souborový systém, ve kterém můžeme založit soubor a využít tento soubor k uložení našich dat. Velikost SPIFFS oddělení na čipu ESP32 s 4MB flash pamětí se pohybuje okolo 2MB, což pro naše potřeby bude dostačující. Na tuto implementaci se dá využít knihovna SPIFFS.h [16].

Vzhledem k tomu, že se jedná o soubor, můžeme ukládat data v konvenční formě do například CSV souboru, dle našeho přání. V mé implementaci jsem pro uložení dat zvolil právě CSV soubor.

3.2.3 Technika zamykání přihrádek

Jako jedna z funkcí, kterou má lékovka zpřístupnit, je zabránit uživateli v konzumaci léků, které nejsou momentálně aktuální.

K tomuto účelu lze využít solenoidy. Připojené způsobem, který je vyobrazen na mém schématu 3.9.

Dioda na obrázku je využita pro filtraci napěťových spiků, které mohou nastat při vypnutí solenoidu. Tato dioda by mohla v praxi být nahrazena mosfetem, pokud bychom chtěli zamezit úbytku napětí, které s sebou dioda přináší.

Pro mé účely testování jsem zvolil obyčejný 5V solenoid. Je malý a levný - ideální pro implementaci do obdobného zařízení, jelikož nevyžaduje vysoké napětí a zároveň limituje váhu zařízení, která by v případě přenositelného zařízení měla zůstat co nejmenší.

Obrázek 3.3. Dvoustavový solenoid

V čem je však solenoid problematický, je jeho potřeba proudu. Pokud bychom napájeli solenoid z baterií, budou potřeba výkonné články, které budou schopné dodat požadovaný proud. Pokud bychom měli solenoid držet sepnutý po dlouhou dobu, velice rychle bychom přišli o veškerou kapacitu baterií.

Možným řešením tohoto problému by byla přesná detekce pohybu šuplíčku a solenoid otevírat jen na nezbytně dlouhou dobu.

Celý tento solenoidový přístup by mohl být možná nahrazen pomocí sofistikované mechaniky a servomotoru. Tuto skutečnost jsem si však uvědomil až příliš pozdě ve vývoji a zůstal jsem u implementace pomocí solenoidů s tím, že pokud by tato varianta selhala, uvažoval bych o této alternativě.

Je zde ještě alternativní přístup, který stojí za uvážení a to použitím elektropermanentních magnetů [17]. Jejich velká výhoda je, že po sepnutí nevyžadují žádnou dodatečnou energii. S touto technologií nemám však žádné zkušenosti a nevím, zda by tato implementace byla možná u našeho zařízení.

3.2.4 Technika detekce otevření/uzavření přihrádky

Pro detekci se dají použít magnetické spínače, které nám umožňují detekovat, co všechno je otevřené/uzavřené.

Fungují na principu statického magnetu na jedné straně a senzoru na druhé. Senzor funguje na jednoduchém principu dvou vodičů, vzájemně oddělených. Přiblížení magnetu způsobí kontakt těchto dvou vodičů a my můžeme získat informace o stavu přihrádky.

Pokud bychom chtěli přesnější monitorování stavu přihrádky, můžeme přidat jeden dodatečný senzor například jeden na přední a druhý na zadní stranu přihrádky. Díky tomu budeme schopni detekovat i menší pohyby a na jejich základě šetřit energii tím, že budeme schopni napájet solenoid jen v nutné chvíle. Když bude šuplík plně vyndaný, je zbytečné, aby po celou dobu byl solenoid sepnutý. Pomocí detekce předního magnetu poznáme, že se šuplík vrací a aktivujeme solenoid. Ve chvíli kdy bude aktivován i zadní senzor, víme, že šuplík je plně zastrčený a můžeme solenoid vrátit opět do uzamčené pozice.

3.2.5 Display

Velkou otázkou, kterou jsem i konzultoval s mým vedoucím bylo, zda dodat k zařízení displej. Jedná se o periferii, která by výrazně zvedla cenu celého zařízení, ale zároveň i jeho schopnost být použito bez připojení k WiFi.

Displej by umožnil uživatelovi zařízení lépe spravovat. A umožnilo by to zařízení přidat další funkcionality.

Displej je také unikátní v tom, že pokud by bylo potřeba zařízení v rámci možností změnit, dá se tak provést pouhou změnou v softwaru bez nutnosti měnit celý design zařízení. V případě, že by bylo celé zařízení vybaveno i dotykovou vrstvou na displeji, bylo by možné měnit celé uživatelské rozhraní.

Výhodou je, že s všeobecným postupem v technologiích, jsou již senioři na dotykové displeje zvyklí a umí je ovládat. S postupujícím časem se tato gramotnost bude rozhodně jen zvyšovat a je důležité s tímto vývojem do budoucna počítat ve výběru finální implementace.

Obrázek 3.4. 4 palcový display [18]

Je důležité si však uvědomit, že takový displej by mohl být dosti redundantní. V dnešní době většina domácností již má displej, který by se k takovému účelu dal využít - mobilní telefon, počítač. Zařízení, která se k lékovce mohou připojit skrze webové rozhraní a lékovku takto ovládat. V tomto případě by se ušetřilo na displeji, ale zařízení by se stalo závislé na okolních zařízeních.

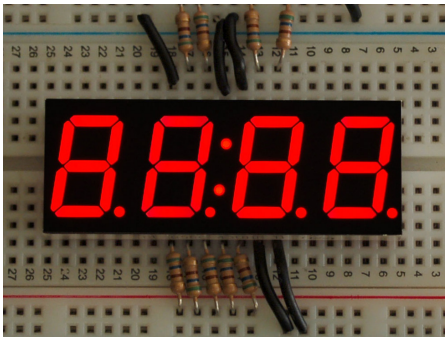
Cena displeje není malá. Musíme si také uvědomit, že většina chronicky nemocných jsou senioři, kteří mají častokrát i zrakové problémy a vyžadují větší displej. Dalo by se ušetřit kdyby se využil monochromatický displej, místo plně barevného. Cena však bude stále vysoká.

Pokud by se jednalo o uživatele, který bude primárně užívat zařízení offline a bude ho i offline spravovat, displej by byl velice dobrým dodatkem. V momentě, kdy zařízení bude připojeno po většinu času k internetu a bude spravováno někým jiným, začne být displej redundantní.

Obrázek 3.5. 7 palcový display [19]

Je však dobré do zařízení dodat něco, co uživateli umožní alespoň nějakou zpětnou vazbu, či případně možnost provést základní nastavení, jako přidat upozornění na lék.

Pro tuto funkcionalitu se hodí nějaká obdoba sedmi segmentového displeje. Takovýto displej je levný, avšak dokáže zprostředkovat základní informace, jako čas, případně jsme z něj schopni zadat čas nový, který můžeme využít na seřízení, či na zadání nového upozornění.



Obrázek 3.6. Sedmi segmentový displej [20]

Je cenově velice dostupný a v kombinaci s vhodným posuvným registrem umožní jednoduchou obsluhu. Výhodou je také, že lidé jsou již s obdobnými displeji dobře seznámeni a nebyl by to z hlediska uživatelské zkušenosti problém s využíváním obdobného zařízení.

Pro usnadnění obsluhy a přidávání/odebírání upozornění by mohl být displej více modifikován, například dodatečnými statickými ikonami. Jejich rozsvícením by zařízení naznačilo, v jakém módu se nachází. Zda bude alarm přidávat, či odebírat a podobně.

Možnost je také samozřejmě vynechat displej úplně. Tento přístup by ušetřil nejvíce peněz, ale zase limitoval zařízení výhradně na online obsluhu. Pokud by se jednalo o uživatele, který je technicky zdatný, dal by se tento kompromis ospravedlnit. Avšak v našem případě nemůžeme na něco podobného spoléhat.

V mém konceptuálním produktu jsem se rozhodl implementovat přístup využívající sedmi segmentový displej. Použil čtyř číselný displej napojený na posuvný registr TM1637 a knihovnu TM1637Display.h [20]. Sedmi segmentový displej je vhodný na prověření konceptu a nevyžaduje rozsáhlou práci na uživatelském rozhraní.

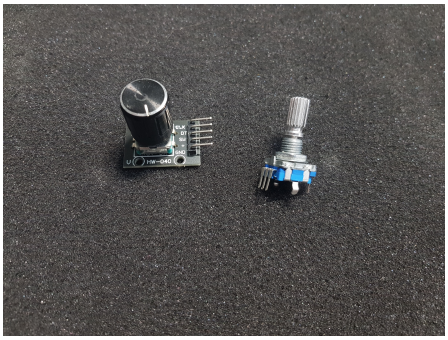
Finální produkt by však mohl mít různé verze, které budou každý mířený pro jiného uživatele s jinou cenovkou.

3.2.6 Vstupní periferie uživatelského rozhraní

Pokud bychom chtěli uživateli umožnit offline nastavení zařízení, jsou vstupní periferie prakticky nezbytné. Naše periferie se budou lišit s ohledem na použitý displej.

Pakliže by se jednalo o dotykový displej, bylo by vhodné veškeré interakce provádět skrze něj. Omezíme tím výdaje za periferie a prodloužíme životnost zařízení limitováním mechanických součástek.

V případě užití sedmi segmentového displeje, nebo displeje bez dotykové vrstvy je využití mechanického vstupního zařízení nutné. V takovém případě jsem se rozhodl pro implementaci pomocí tlačítek a rotačního enkodéru.



Obrázek 3.7. Rotační enkodér

Rotační enkodér využívám pro jeho intuitivní ovládání. Rotační pohyb je ideální v nastavování času a případnou orientaci v navigačním menu, pokud by zařízení bylo vybaveno i displejem.

Pro vhodné ovládání jsem se rozhodl pro dvě tlačítka a rotační enkodér, který je klikatelný. Díky kombinaci těchto vstupních periferií jsem schopen dodat obstojnou uživatelskou zkušenost. A následně pomocí těchto periferií zařízení nastavit pomocí offline logiky. Tato logika není preferovaná, avšak je dobré mít možnost zařízení tímto offline způsobem spravovat.

Pro ovládání rotačního enkodéru jsem zvolil knihovnu RotaryEncoder.h [21].

3.2.7 Zvuková signalizace

Aby mohlo zařízení býti používo pro větší škálu uživatelů. Rozhodl jsem se pro přidání volitelného modulu - reproduktoru. Tento reproduktor umožní podat zpětnou vazbu uživateli, Jako například, říci nahlas čas, upozornit, který šuplíček je nedovřený, že dochází baterie atp.

Věřím že tato signalizace pomůže primárně lidem, kteří mají problémy se zrakem. Zvuková signalizace by mohla být v tomto ohledu velice přínosná.

ESP32 je připravené na přehrávání zvuku. Má dva Digital-to-Analog piny, které se pro daný úkon dají velice dobře využít. ESP32 má i knihovny umožňující přehrání zvuku uskutečnit. Pokud bychom však chtěli nejen statické předem nahrané fráze, bylo by potřeba připojit zařízení ke službě, které by umožnila přeložení z textu do zvuku. Více v sekci o softwaru.

3.3 Napájení

Napájení pro finální zařízení by mělo být kombinované. Tedy schopné být napájeno jak z přímo ze sítě, tak z baterií. ESP čip sám pracuje na 3V3 logice, avšak moje deska obsahuje regulátor napájení, který je schopen přijmout i vyšší napětí. Z důvodu napájení periferií, jako jsou solenoidy je nezbytné využít napájení o minimálním napětí 5V.

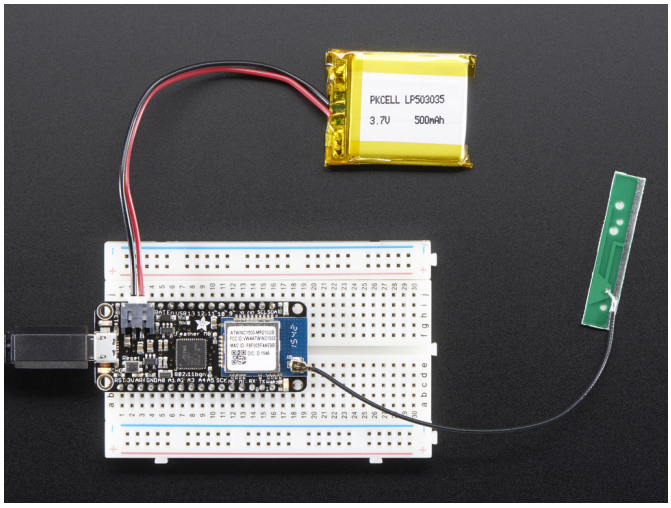
Ve chvíli, kdy však vytváříme zařízení na baterie, je nutné limitovat veškeré napájecí ztráty, abychom zvýšili dobu, během které je zařízení schopno operovat. Z tohoto pohledu je neefektivní napájet zařízení vyšším napětím, které je následně redukováno. V takovýto okamžik by stálo za zvážení oddělit napájení čipu a solenoidů.

Vezmeme-li k tomu v úvahu vysoký proud potřebný pro aktivaci solenoidu, idea oddělených napájení není zdaleka tak nereálná.

Napájení samotného čipu by mohla dobře obstarat baterie $LiFePO_4$ [22].

Hlavní výhoda tohoto typu baterie je, že napětí, které dodává dost dobře odpovídá napětí, které ESP čip vyžaduje a tím pádem není nutné napětí nijak regulovat. Dále je tento typ baterie stabilní, dá se dobíjet a vykazuje dobrou spolehlivost.

Další dobrou volbou by mohla být LiPo baterie [23].



Obrázek 3.8. Deska Adafruid Feather M0 s připojenou LiPo baterií [24]

Tyto baterie mají výhodu, že jsou již v praxi hojně užívány, implementovány a mají dobrou podporu. Můžeme jednoduše nalézt i desky, které mají pro tento typ baterie nativní podporu, jako například FireBeetle ESP32 IoT Microcontroller [25].

Napájení solenoidů z baterie představuje kvůli požadavku na vysoký proud výzvu. Pro tento úděl bych vybral baterii typu NiMH [26].

Tyto baterie jsou využívány například v modelářství, díky jejich schopnosti dodat proud ve výšce jednotek ampér. Nevýhodou je, že jednotlivé baterie mají malé napětí a bylo by nutné jich využít několik zapojených do série, pokud bychom chtěli napájet 5V solenoid.

Téma baterií je nutné ještě do budoucna prozkoumat a dobře zmapovat, neboť volba baterie je pro celé zařízení klíčová. V mé implementaci jsem operoval zatím primárně se statickým napájením a pro budoucí vývoj bude ještě třeba provést patřičné testy.

3.4 Návrh vnějšího provedení zařízení

Hrubý návrh zařízení jsem již popsal v sekci o možných přístupů řešeních. Zde si dovolím přiblížit některé faktory, které ovlivní budoucí produkt a jeho vzhled.

Hrubou představu o zařízení již máme. Jedná se o krabičku, ve které jsou umístěny šuplíčky na skladování léků. Zařízení dále obsahuje potencionální displej a případné vstupní periferie.

Začnu s podrobnějším popisem šuplíčků. Ten musí být dostatečně malý na to, aby zařízení zbytečně nenarůstalo na velikosti, ale zároveň dostatečně velký, aby umožnil uživateli pohodlnou manipulaci. S mým vedoucím práce jsme toto téma dlouho diskutovali a výsledkem byl závěr, že je třeba šuplík navrhnout tak, aby uživatel měl pohodlný přístup k lékům, nemusel je složitě vytahovat, či být schopný je vyklepnout.

Pokud uvažujeme zařízení, které má co šuplík, to lék, uživatel má možnost si odsypat potřebnou dávku, pokud by měl problémy s vyndáváním léků. V případě, kdy se jedná o situaci - jeden šuplík jeden den, je třeba nastavit přístup jiný. Zde již nejde jednoduše obsah vysypat, neboť bychom riskovali promíchání denních dávek.

Pokud bychom chtěli i v tomto případě zachovat možnost vyklepnutí léků, dal by se tento problém potencionálně vyřešit formou krytí, které by bylo schopné se přemísťovat a na základě toho určit, co se má vyklepnout. Šuplík by mohl mít drážky, do kterých by byly vsazeny dvě ploché destičky, kterými by se dalo pohybovat. Každá destička může zakrývat jeden segment. V tomto případě je možné posunem těchto destiček vybrat, který segment má být připraven k vyklepnutí a které dávky mají zůstat uvnitř.

Další otázkou je umístění displeje. Zdali by se mělo jednat o umístění na plochu zařízení, kolmo, pod úhlem, či polohovatelně. Vzhledem k tomu, že by zařízení mělo být schopné fungovat, minimálně v případě sedmi segmentového přístupu, i jako zobrazovač hodin, je ploché připojení nevhodné.

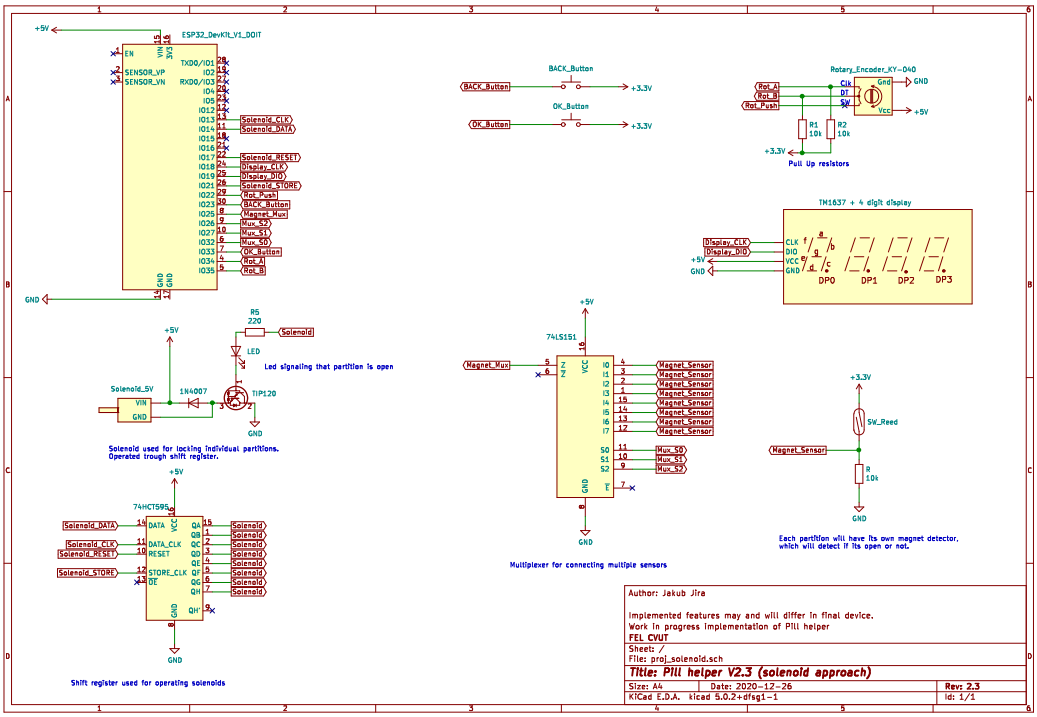
Polohovatelný displej přinese nejvíce flexibility, avšak nutnost vyvinou, nebo minimálně pořídit kloub pro polohování displeje by zbytečně zvedla cenu a ovlivnila i případnou poruchovost zařízení. Z tohoto důvodu bych volil staticky připojený displej.

Vstupní periferie musí být umístěny tak, aby umožňovali pohodlnou obsluhu displeje a jejich obsluha by neměla člověku nijak bránit ve výhledu na displej.

V rámci této práce jsem nestihl vytvořit prototyp, do kterého bych mohl zařízení uložit. Vzhledem k mé minimální apriorní znalosti s 3D tiskem jsem se rozhodl vyhnout riskantní časové investici, která by znamenala nastudování si celého tématu 3D modelování a 3D tisku. Tato fáze je však v budoucí cestě produktu nezbytná.

3.5 Elektronický návrh

Většinu výše zmíněných komponent jsem vložil do návrhu 3.9. Ve schématu se nachází uživatelská interakce ve formě dvou tlačítek a rotačního enkodéru, návrh na zapojení jednoho solenoidu a jejich případné zapojení skrze posuvný registr. Schéma magnetického senzoru a jejich napojení na multiplexor. A displej napojený na posuvný registr TM1637. Toto jsou moduly, které mám reálně ozkoušené a vím, jak se budou chovat.



Obrázek 3.9. Schéma zapojení základních periferií v KiCadu

Mezi další periferie, které se v momentálním schématu zatím ještě nenachází je reproduktor s případným zesilovačem a baterie. Oboje z důvodu, že jsem implementaci těchto modulů odložil na pozdější čas, jelikož nebyly nezbytně nutné pro vývoj konceptuálního zařízení. Je s nimi však do budoucna počítáno.

Pro reproduktor je uvolněn jeden z DAC pinů - jmenovitě pin 15. Implementace baterií změní schéma v tom, že přibude dělič napětí a jeden, či více pinů (záleží na počtu baterií), bude použit na čtení stavu baterie.

3.6 Volitelné rozšíření

V rámci studie tohoto zařízení je dobré prozkoumat i cesty, kterými by se produkt mohl vydat do budoucna a případně k nim uzpůsobit již momentální architekturu, aby budoucí implementace byla co nejsnazší.

Jeden z potencionálních modulů by mohl být mikrofon, jakožto dobrý doplněk k reproduktoru. Díky mikrofonu by uživatel dostal další cestu, jak interagovat se zařízením, a díky cloudové službě by mohlo zařízení fungovat jako domácí zařízení obsahující i virtuálního asistenta v podobě například Alexy, Google asistenta či Cortany.

Jakožto zdravotní produkt, který tato lékovka bez pochyby je, by se dal zkombinovat s dalšími potencionálními zdravotními pomůckami, jako například tlakoměrem. Tímto způsobem, pokud by byla zaručena komunikace mezi naším ESP čipem a tlakoměrem, by mohlo zařízení reportovat i naměřený tlak. Další funkci, kterou by mohl produkt zastat jakožto zdravotní pomůcka, by mohlo být zařízení, které je schopné přivolat pomoc.

Dalším možným modulem by mohla být čtečka čárových kódů. Pokud by systém byl dobře implementovaný, uživatel by nemusel řešit zadávání podrobností do zařízení. Pouze by naskenoval čárový kód z krabičky léků a o zbytek by se postaralo zařízení. Tento modul je velice optimistický, ale nepřijde mi nereálný.

Kapitola 4
Programové vybavení

4.1 Volba programovacího jazyka

Jak jsem již několikráte zmínil, jádrem celého systému je čip ESP32. Tento čip umožňuje několik přístupů, jak by se dal programovat.

Nejzákladnější přístup je pomocí jazyka C/C++ a frameworků využívajících tyto jazyky jako svůj základ. C/C++ má velkou řadu výhod. Jakožto kompilovaný jazyk je jeho exekuce rychlejší, než u interpretovaných jazyků, což se ve vestavných systémech s limitovaným výpočetním výkonem využívá. Zároveň mi ale přijde vhodné zmínit i alternativní přístupy programování ESP čipu.

Hojně využívaný v poslední době je jazyk, který nese název MicroPython [27]. Výhoda tohoto jazyka je bezpochyby jeho programátorská přívětivost, která si bere, jak již název napovídá, sytax jazyka Python a dodává k základním knihovnám další, které umožňují zařízení komunikovat s periferiemi plus další funkce, které se u vestavných systémů vyžadují. MicroPython umožňuje i zahrnout C kód, takže se jedná o všestranné zařízení. S MicroPythonem jsem se již setkal, a věřím že má ve světě své místo.

Další zajímavým zajímavým jazykem by mohlo být Espruino [28]. Jedná se o interpreter pro JavaScript uzpůsobený pro ovládání vestavných zařízení. Jakožto interpretovaný jazyk je pomalejší, avšak Espruino s tímto počítá a umožňuje i přidání předkompilovaného kódu, pro časově náchylné operace. Interpreter však má i své výhody. Jakožto interpretovaný jazyk je schopen například modifikovat funkce za běhu programu. To znamená, že se dají provést změny v kódu, bez toho, aniž by zařízení muselo být vypnuto.

Když už budeme mluvit o jazyku C, je nutné zmínit Assembler. ESP toolchain umožňuje využít psaní v čistém assembleru. V rámci studia jsem se díky předmětu Návrh vestavných systému s assemblerem pro vestavné systémy dobře seznámil, a pochopil, že pokud navrhujeme vysoce optimalizovanou aplikaci, tak pouze assembler nám umožní přesně korigovat vykonávané instrukce. Avšak vzhledem ke komplexnosti aplikace a z důvodu nutnosti využití knihoven pro komunikaci s cloudem je přístup assembleru velice náročný.

Mou volbou pro tento projekt bude jazyk C/C++. Díky široké základně kódů je pro danou aplikaci naprosto ideální. V rámci těchto jazyků také hojně využívám nativní framework pro rodinu ESP čipů - ESP-IDF. Čisté C je pro mě nativní jazyk a také je hojně využíváno v kódu. Většina mnou využívaných knihoven využívá primárně jazyk C++, a tak může můj kód v některých případech vypadat nekonzistentně. Na moji obranu uvedu to, že za dobu studia jsem se bohužel nesetkal s předmětem, který by vyučoval C++, a tak z důvodu ušetření času jsem obětoval konzistenci za čas, který bych strávil nad studiem specifikací C++.

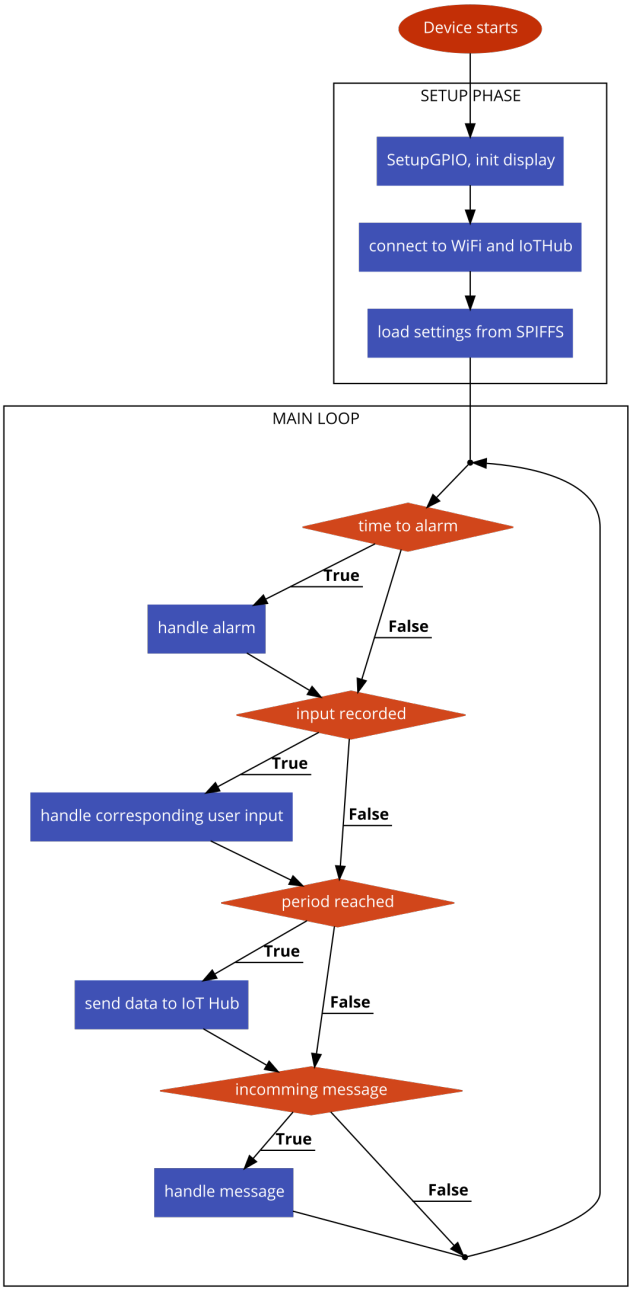
V rámci mého kódu také používám knihovnu arduino.h. Můj vývoj původně začal bez použití této knihovny, avšak pří následném vývoji a snaze připojit zařízení k Azure IoT hubu jsem se k arduinu přemístil z důvodu použití Azure IoT hub knihoven, které knihovnu arduino.h využívají.

V rámci vývoje jsem uvažoval i nad implementací minimálního operačního systému, jako FreeRTOS [29]. Za dobu vývoje jsem však nepocítit nutnost přejít ze sekvenčního vykonávání kódu, na něco sofistikovanějšího a tak tato možnost zůstala nevyužitá. Věřím však, že v budoucnosti, až bude komplexnost vykonávaných funkcí narůstat, bude implementace nějaké formy operačního systému velice vhodná.

4.2 Popis funkce software

V této sekci provedu popis fungování software. V první části ukáži koncept, jak je míněno celé finální zařízení. Ve druhé popíši, co se mi podařilo implementovat a jak jsem k dané problematice přistupoval, případně, jak ji řešil.

Naznačení hrubé funkce software je v následujícím vývojovém diagramu. Jedná se o obrysový vývojový diagram, který znázorňuje funkci softwaru zařízení 4.1.



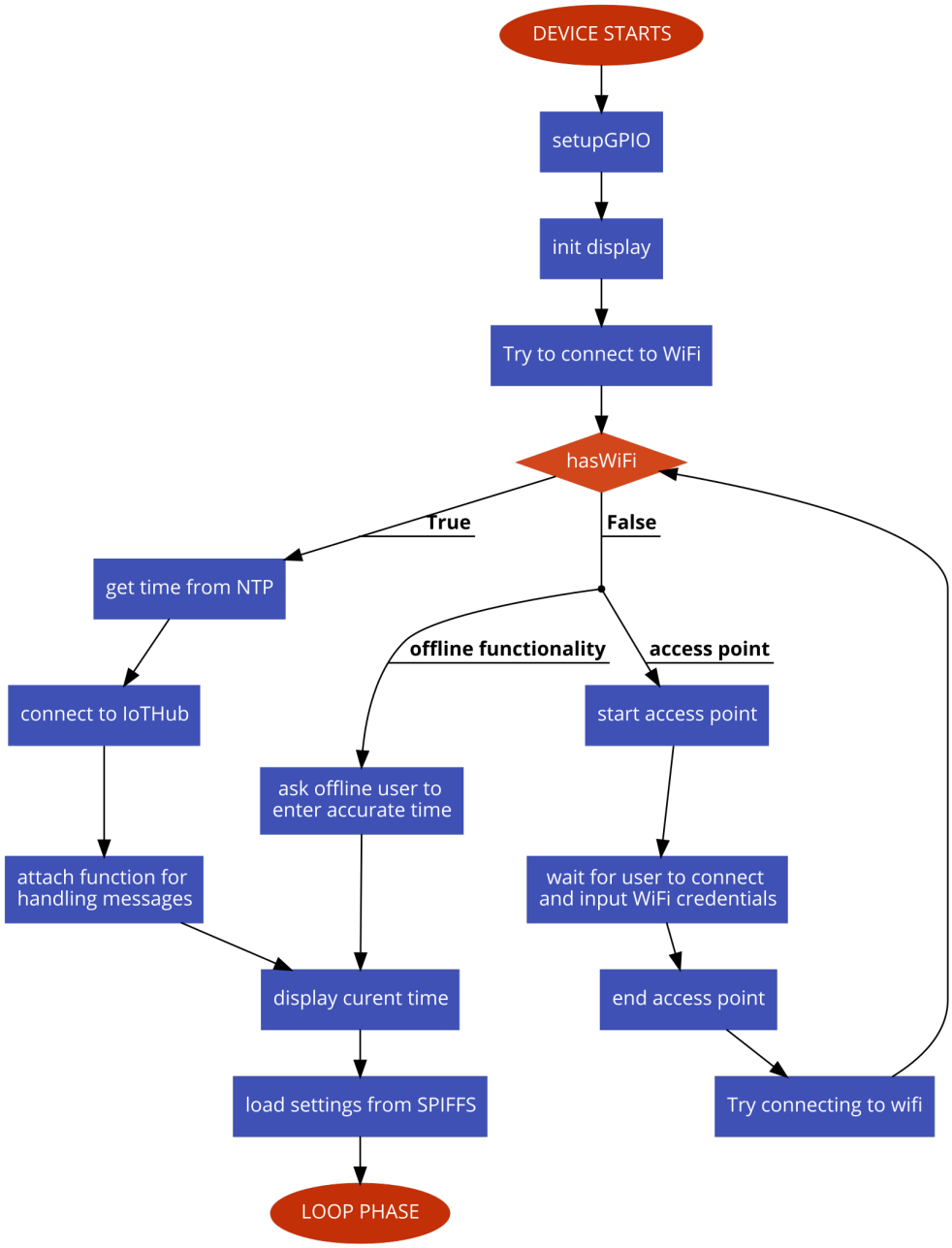
Obrázek 4.1. Kostra funkcí programu.

4.2.1 Fáze setup

Funkce setup, je funkce, která je volána právě jednou při startu zařízení. V rámci této funkce inicializuji periferie a navazuji spojení a připravuji zařízení na jeho očekávaný chod. Funkce setup je blíže popsána ve vývojovém diagramu 4.2.

Neprodleně po startu začíná inicializace periferií, které budeme využívat. Jedná se o inicializaci všech uživatelských vstupů, displeje, zapnutí WiFi a v mém konceptu i inicializací seriového rozhraní, které je během vývoje využito k logování a pomocným výpisům. Finální zařízení seriové rozhraní mít aktivní nebude.

Následující funkcí je inicializace displeje. Rozsvícením dáme uživateli najevo, že se něco děje a případně ho můžeme informovat o momentálně prováděných úkonech.



Obrázek 4.2. Naznačení funkce setupu pomocí vývojového diagramu.

Následuje pokus o připojení k WiFi.

Pokud se připojení zdaří, zařízení vyšle žádost NTP serveru a následně aktualizuje vnitřní hodiny. Pro komunikaci se využívá SNTP protokol, který je určený právě pro zařízení vestavných systémů.

Poté proběhne pokus o připojení k Azure IoT hubu. Pro komunikaci s IoT hubem jsem použil nástavbu Visual Studio Code - Azure IoT Device workbench [30], který využívá odnož oficiální knihovny Microsoftu pro programování Arduino zařízení [31].

Díky této knihovně jsem schopen přijímat stringy z Azrue IoT Hubu a následně je předat mnou zvolené funkci. Tento proces jsem ve vývojovém diagramu znázornil uzlem *attach function for handling messages*. Tato funkce bude zavolána pokaždé, kdy bude přijata zpráva.

Pokud se připojení k WiFi nepodaří, zařízení aktivuje dva procesy. Jeden je plně offline, během kterého bude uživatel požádán o zadání přesného času, aby zařízení bylo schopné fungovat alespoň v minimální podobě. V případě, že by zařízení obsahovalo modul, který drží přesný čas 3.2 a bylo by nastaveno, tento krok se může přeskočit.

Druhý proces přepne zařízení do režimu access point. Zařízení umožní uživateli se připojit na WiFi síť vytvořenou naším zařízením. Po připojení bude uživateli umožněno zadat přihlašovací údaje pro WiFi, ke které se má zařízení připojit. Pro celý tento proces je dobře uzpůsobená knihovna WiFiManager [32].

Během vývoje jsem chtěl tento proces co nejvíce automatizovat a co nejméně vyžadovat od uživatele interakci. Pro tento přístup jsem se přiklonil z důvodu toho, že jsem nenalezl lepší řešení tohoto problému. Dalšími kandidáty, nad kterými jsem uvažoval byla implementace nějaké formy lokálního serveru a využít protokol UPnP [33], či nějak využít protokolu WPS [34]. Ve finále jsem se však rozhodl odstoupit od těchto variant. Ať z důvodu bezpečnosti, či složitější implementace.

Po zadání přístupových údajů uživatelem a úspěšném přihlášení bude zařízení pokračovat ve svém očekávaném módu.

Setup fáze končí načtením medikamentů ze souboru uloženém v ESP zařízení. Tento soubor je typu CSV [35]. Hlavičkový záznam vypadá následovně:

```
pillName,assignedPartition,alarms,activeDays,dose,numberOfPills
```

V tomto formátu jsou ukládány všechny medikamenty poté, co jsou založeny. Program je uzpůsobený na to, aby tyto zápisy četl a vytvářel z nich struktury, se kterými pracuje v živém běhu.

PillName je `String` o maximální velikosti 127 charů a určuje název přiřazený medikamentu. Tento String se také v momentální fázi používá jako primární klíč. Porovnávání Stringů není nejrychlejší možná metoda, ale pokud by rychlost, případně výpočetní náročnost v budoucnu byla nutná optimalizovat, dala by se jednotlivým zápisům přiřazovat jednoznačná číselná hodnota, či pomocí hashovací funkce hledání urychlit.

AssignedPartition je typu `uint8_t` a značí přiřazenou přihrádku danému léku. Tato informace se nedá použít jakožto primární klíč, protože chci, aby byl uživatel schopen lék, který nebude nějakou dobu užívat, deaktivovat a později ho případně znovu aktivovat, bez nutnosti provádět celý zakládací proces.

Položka alarms obsahuje všechny časy, během kterých má být uživatel na tento medikament upozorněn. Vnitřní logika umožňuje uložit a načíst několik upozornění pro jeden medikament. Pro strukturu alarmů jsem použil následující vzor: `<dd-dd>`, kde d znační jeden dekadický charakter. Tento vzor se může libovolně opakovat. Takže pokud bychom měli medikament, který se má užívat v 8:30 a 17:05, zápis ve sloupci alarms by vypadal následovně: `<08-30><17-05>`.

ActiveDays v sobě drží informaci o tom, v jaké dny je tento medikament aktivní. Tato položka je typu `uint8_t`, kde každý bit značí jeden den, počínaje od neděle na intexu 0. Bit na intexu 7 je zatím nevyužitý, ale dal by se použít pro určení, zda je medikament aktivní každý den, bez nutnosti číst celé slovo. Pokud by tato modifikace byla užitečná by záleželo na užívání.

Dose je označení pro dávku, kolik léků se má užít. Hodnota je typu `uint8_t`.

NumberOfPills je označení pro momentální počet léků uložených v zařízení.

4.2.2 Fáze loop

Tato fáze je hlavní tělo programu a obsahuje veškerou logiku rozhodování zařízení. Momentálně funguje na principu kontrolování podmínek v sérii, případném vykonání dané funkce. Do budoucna by však bylo vhodné implementovat formu operačního systému, který by umožnil vykonávání těchto funkcí v paralelní formě. Mohli bychom tím lépe kontrolovat například problém výpadku WiFi a její případné obnovení.

Na počátku smyčky projde zařízení všechny uložené medikamenty a zkontroluje, zda některý z nich není v časovém oknu, kdy by se měl užít. Pokud se takový medikament najde, je spuštěna funkce alarmu, která upozorní uživatele a čeká na užití léku.

Forma upozornění bude vázaná na specifického uživatele s tím, že základní je světelná a zvuková notifikace.

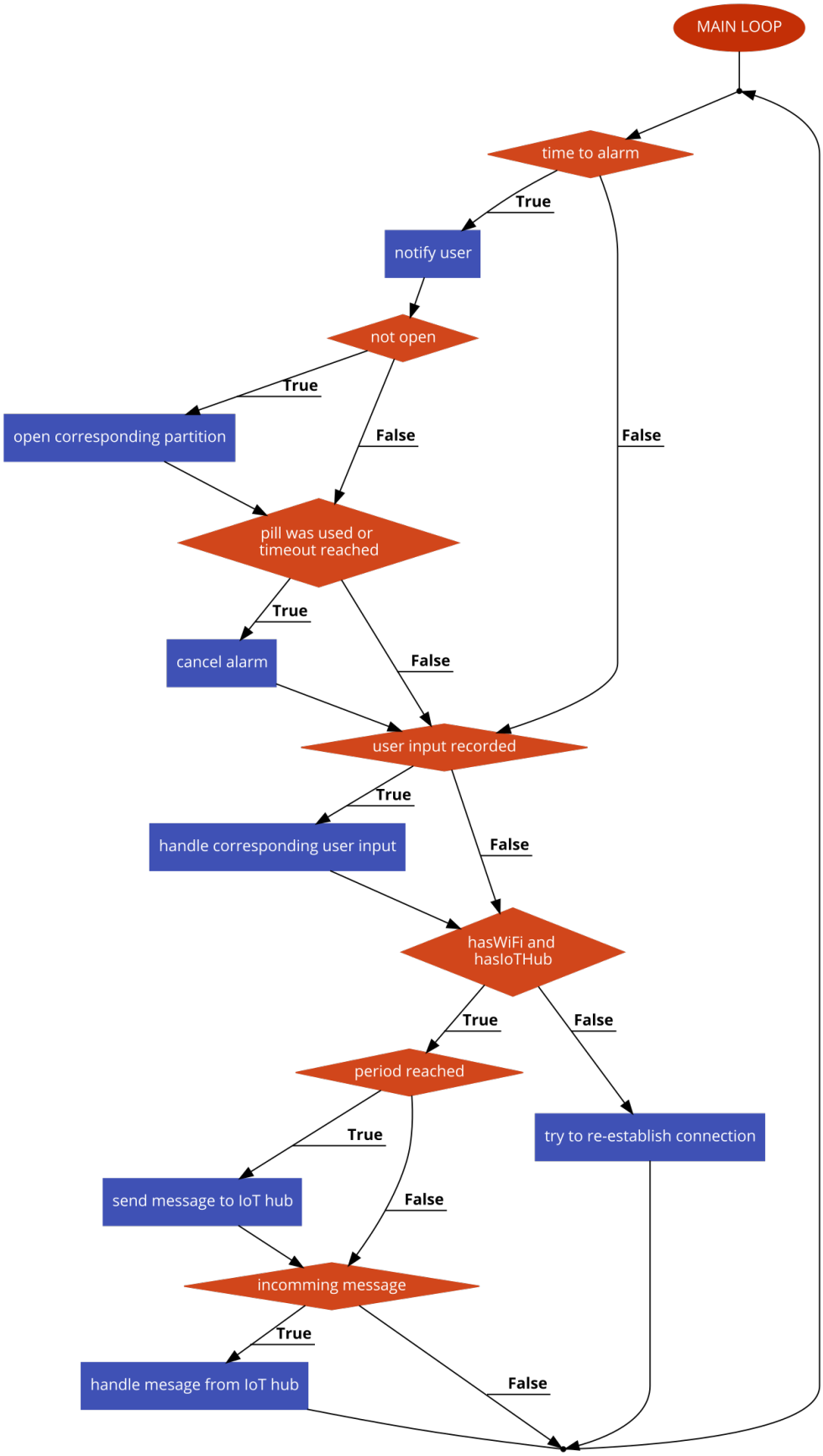
Po splnění některé z podmínek na ukončení alarmu se upozornění ukončí. Těmito podmínkami může být buďto užití léku, což bude vyhodnoceno monitorováním pohybu přihrádky za pomocí magnetického senzoru. Druhou podmínkou je vypršení časového okna během kterého má být medikament užit. Tyto informace se logují a následně reportují do serverové aplikace.

Momentálně je tato funkce implementována pouze jakožto vyslání signálu do statické periferie. Avšak vnitřní architektura je připravena na vykonání celé vyžadované sekvence.

Následuje kontrola uživatelské interakce. Tento bod je velice široký, jelikož záleží na tom, pro jaký displej ovládání implementujeme. V sekci o uživatelském rozhraní mám návrh, jak by celé rozhraní mohlo vypadat a fungovat. Plus ukázku velice základního naprogramovaného konceptu, jak by mohla vypadat interakce se sedmi segmentovým displejem.

Následuje kontrola připojení a případná interakce s IoT hubem. Pokud je Připojení v pořádku, zařízení pošle zprávu o tom, které medikamenty se uvnitř zařízení nacházejí a v jakém stavu jsou.

Zároveň se také zkontroluje, jestli nejsou ve frontě nějaké zprávy z IoT hubu pro zařízení. Momentálně implementované funkce jsou Přidání, Odebrání, Modifikace a Report. Přidáním, Odebráním a Modifikací můžeme měnit obsah medikamentů uvnitř zařízení. Možnost report je implementována za účelem, pokud nechceme čekat na periodu během které zařízení reportuje stav.



Obrázek 4.3. Naznačení funkce hlavní smyčky pomocí vývojového diagramu.

Formát komunikace je inspirován formátem ukládaných dat - používá se stejný CSV princip s tou výjimkou, že zpráva je uvedená příkazem, který ze má vykonat a samotný popis medikamentu CSV je uložen ve složených závorkách. Příkazy vypadají následovně:

```
<ADD>{pillName,assignedPartition,alarms,activeDays,dose,numberOfPills}
<REMOVE>{pillName}
<MODIFY>{pillName,setting,modification}
<REPORT>
```

`<ADD>` je identický s uloženým záznamem v CSV souboru. `<REMOVE>` odebere medikament určený proměnnou pillName. `<REPORT>` je určen pro vyžádání si statusu zařízení.

Jediný zajímavější příkaz je `<MODIFY>`. PillName značí, který medikament má být modifikován. Setting je typu `uint8_t` a značí, který atribut má být modifikován. Modification je pak nová hodnota daného atributu.

4.3 Následná rozšíření

Tento vývojový diagram, který jsem zde popsal však není plná funkcionalita, kterou od minimálního produktu očekávám a měla by mít ještě následující funkcionality.

První a hlavní funkcionalita, která je nutná, zda-li má zařízení být napájeno z baterií, je funkce, která bude monitorovat stav baterií. Pro tuto funkci bych využil ADC pin a skrze něj bych četl napětí na baterii. Případně pokud by bylo nutné bych použil dělič napětí, abych se přizpůsobil 3V3 logice, kterou ESP32 očekává.

Další zajímavý problém je nutnost být schopen vzdáleně aktualizovat software. Pro tento případ je zařízení ESP připraveno takzvaným OTA [36]. Tento proces lze napojit i na Azure a provést ho plně automatizovaně.

Kapitola 5
Uživatelské rozhraní

Hlavní vlastností, která bude schopná zařízení prodat, však nebude výkonný čip, nebo geniální architektura. To jsou jen věci, které mají dopomoci něčemu finálnímu. A tím finálním je co nejlepší uživatelská zkušenost. A ať chceme nebo ne, uživatelské rozhraní je něco, co může zařízení naprosto vysvobodit z některých ne úplně šikovných rozhodnutí, nebo ho naopak úplně potopit.

Co trochu komplikuje celý návrh je rozmanitost produktu. Tím, že nemáme jednotný návrh displeje, je třeba počítat se všemi možnostmi a tyto možnosti co nejlépe přizpůsobit finálnímu zařízení.

5.1 Přizpůsobení zařízení uživateli

Při navrhování uživatelského rozhraní jsem počítal s tím, že se bude jednat o člověka, který nemá s ovládáním elektronických zařízení zkušenosti a bude požadovat co nejjednodušší ovládání. Proto jsem se rozhodl vstupní uživatelské periferie udržet na minimálním počtu. Věřím, že to pomůže celkovému zjednodušení ovládání.

Zároveň je také třeba počítat s tím, že koncový uživatel, s dobrou pravděpodobností senior, bude mít problémy se zrakem a bude potřebovat rozhraní co největší, aby bylo pro něj příjemně ovladatelné. A to nejen displej, případně text, ale zároveň i velikosti tlačítek, případně rotačního enkodéru.

Je také vhodné uvažovat nad tím, zda offline uživateli, který bude potencionálně právě senior, nelimitovat možnosti nastavení zařízení. Není to z důvodu, že bychom uživateli nedůvěřovali, ale z důvodu, že čím méně volby uživatel dostane, tím je větší šance, že si nebude připadat zahlcen a větší pravděpodobnost, že si osvojí ovládání zařízení. Kompletní nastavení můžeme nechat na správci zařízení, který se k němu může připojit vzdáleně skrze aplikaci, která už bude mít plnou funkcionalitu.

Tím, že se jedná o software, je možné připravit program, který bude uzpůsobený k tomu, že se u něj mohou, i třeba během chodu měnit nastavení a práva offline uživatele.

5.2 Offline ovládání

Na začátku je nutné si uvědomit, které funkce offline uživatel musí nutně mít. Když se nad touto otázkou zamyslíme, zjistíme, že neexistuje triviální odpověď, neboť to plně záleží na daném uživateli, jakou asistenci potřebuje.

Zvládne sám zařízení dávkovat? Je pro uživatele hrozba, pokud bude zařízení permanentně odemčené? Má si sám spravovat časy léků? To vše ovlivňuje, jaké pravomoci a tedy jaké ovládání uživateli umožníme.

Pokud bychom chtěli, aby offline uživatel měl plnou kontrolu, jsem schopný zařízení vybavit displejem, který umožní zařízení nastavovat. Pro tento případ jsem vytvořil hrubé naznačení, jak by daná aplikace mohla vypadat[1].

[1] `https://www.figma.com/file/hZDGBlHj4rijzdqD5qoDS3/pillHelper?node-id=0%3A1`

Tento návrh pracuje s monochromatickou grafickou paletou, jelikož je možnost, že by finální zařízení mohlo mít pouze monochromatický displej.

Design je navrhnut tak, aby písmena byla co největší a co nejčitelnější. Zde záleží na velikosti displeje. Při navrhování této aplikace jsem počítal s displejem o velikosti kolem 4". Pro tuto velikost je podle mého úsudku velice důležité, aby velikost písma nebyla kompromitována a umožnila co nejlepší čitelnost.

Pro lepší orientaci v programu jsem se rozhodl navrhnout spodní lištu, která monitoruje, kolik se v dané úrovni nachází položek, případně na jaké se nyní nacházíme. Věřím že podobný prvek může výrazně pomoci v orientaci v komplexnějším programu. Obdobně by bylo užitečné implementovat prvek, který by mohl znázorňovat horizontální zanoření. Jediné, na co si je třeba dávat pozor je, aby tyto pomocné prvky nezhoršily čitelnost textu, a aby displej nevypadal zahlceně.

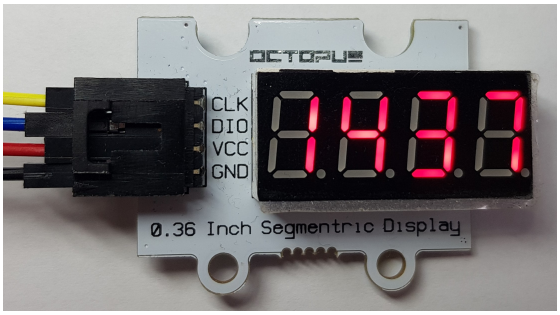
Limitace tohoto přístupu jsou právě problémy s tím, že je zařízení navrhováno co nejjednodušší. Například při vytváření léku je problém se zadáním jména jen za pomoci rotačního enkodéru. Implementace tohoto problému je možná, jak je naznačeno v mockupu. Není to však uživatelsky nejpřívětivější.

Tento problém by mohla vyřešit implementace dotykového displeje, který umožňuje širší spektrum funkcionalit. Rozhodl jsem se však tento přístup nebrat jako primární, vzhledem k ceně displejů. Myšlenka plného odkázání na dotykovou plochu mi nebyla příjemná, neboť mám nějaké zkušenosti s levnými displeji a jejich responzivita na dotyk není nejlepší. To by znamenalo, že investice by musela být vyšší, aby se na tento vstup dalo stoprocentně spolehnout. To by s sebou však také přineslo vyšší cenu.

Zbytek ovládání si myslím, že je v rámci možností přímočarý. Na zadávání čísel se rotační enkodér výborně hodí a navigace skrze okna by neměla být problém.

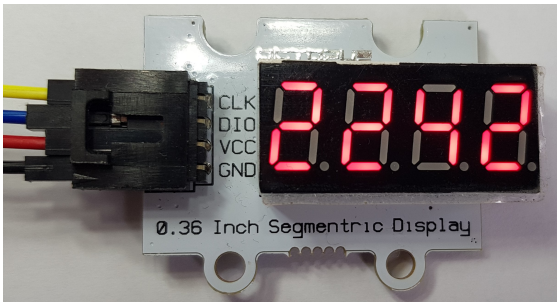
Toto vše je však jenom koncept a finálnímu produktu musí neodmyslitelně předcházet testování v reálných situacích s reálnými uživateli a až tehdy jsme schopni zjistit, na kolik procent je tento přístup vhodný.

Verze zařízení se sedmi segmentovým displejem, má pochopitelně funkcionalit méně, avšak i tak jsem implementoval do svého kódu způsob, jak do zařízení přidat nový medikament.



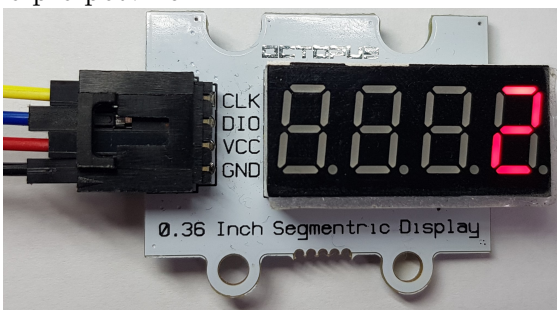
Obrázek 5.1. Displej zobrazuje aktuální čas.

Zařízení je zapnuté a tak displej zobrazuje aktuální čas. Pokud stiskneme a budeme držet tlačítko, které nám umožňuje přidávat nová upozornění, začne hodinová část displeje blikat. To nám dává najevo, že pokud nyní budeme točit enkodérem, budeme modifikovat hodinovou část alarmu.



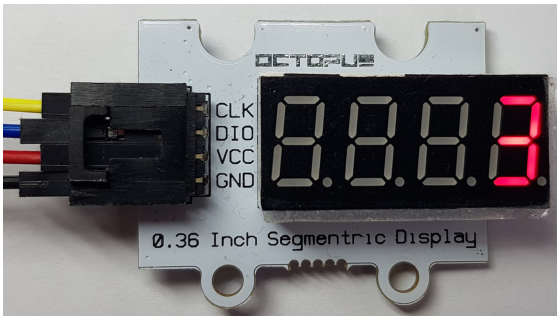
Obrázek 5.2. Displej zobrazuje modifikaci hodin.

Po nastavení správného hodinového času můžeme zmáčknout enkodér. Toto způsobí, že přestane blikat hodinová část, ale začne blikat minutová. Obdobně jako u hodin můžeme nastavit. Pokud bychom se chtěli vrátit k editaci hodin, můžeme tak učinit opětovným stiskem rotačního enkodéru. Pokud jsme s vybraným časem spokojeni, můžeme stisknout tlačítko pro potvrzení.



Obrázek 5.3. Displej zobrazuje výběr přihrádky.

Nyní vybíráme, která přihrádka má být danému medikamentu přiřazena. V nynějším stavu tato volba není ničím doprovázena, avšak ve finálním produktu by tento výběr byl doprovázen světelným označením momentálně vybrané přihrádky. Pokračujeme tlačítkem pro potvrzení.



Obrázek 5.4. Displej zobrazuje určení dávky.

Následuje další volba a tato určuje kolik prášků se má v tento čas užít. Obdobně jako v předchozím kroku vybereme tíženou dávku a potvrdíme tlačítkem.

V tuto chvíli je již vše nastaveno, medikament uložen a můžeme pustit přidávací tlačítko.

Kdybychom se podívali do paměti zařízení, nalezneme nový záznam, který bude vypadat následovně:

```
New Alarm 1,2,<22-42>,255,3
```

Jméno takto zadaného prášku je New Alarm x, kde x se inkrementuje v závislosti na počtu léků zapsaných touto cestou. Následuje číslo 2 značící přihrádku. Čas uložený v očekávaném formátu. Číslo 255 značí, že prášek se má brát každý den. Číslo 3 značí námi zvolenou dávku.

Toto je jenom základní ukázka, jak by dané zařízení mohlo fungovat. U určování přihrádky a dávky se dá určitě využít minimálně prvních dvou segmentů na zobrazení, o jaké nastavení se jedná, abychom uživateli daný proces zpříjemnili. Pro tyto účely by mohli být připraveny i specifické ikony na displeji, který by byl vytvořen přímo pro potřeby tohoto zařízení.

5.3 Online ovládání

Pro nastavení zařízení by měla však být primární a preferovaná cesta nastavení skrze aplikaci, která ideálně běží ve webovém prohlížeči. Momentálně bohužel není tato funkcionalita implementována, více o tom proč v 6.

Tato webová aplikace by však měla mít velice jednoduchý a přímočarý ráz, bez grafických zbytečností. Nejprve by mělo být uživateli umožněno zvolit zařízení, případně zvolit určitou podmnožinu zařízení. A dále by měl být schopen dané zařízení nastavit.

Jednou otázkou, která mě při vývoji napadla je, zda tuto možnost nastavení zařízení není možné dát již prvnímu připojení k zařízení, kdy se zařízení poprvé připojuje na WiFi. Je to určitě potencionální řešení, které může celkovou obsluhu zařízení ulehčit.

Kapitola 6
Serverové řešení

K zprostředkování požadavků, které byly stanoveny je potřeba sofistikované serverové logiky. Moje implementace se momentálně skládá z kombinace cloudového a On-premise řešení. Cloud jsem si vybral, jelikož je to momentální trend, kterým se průmysl řídí a protože jsem se chtěl naučit něco nového.

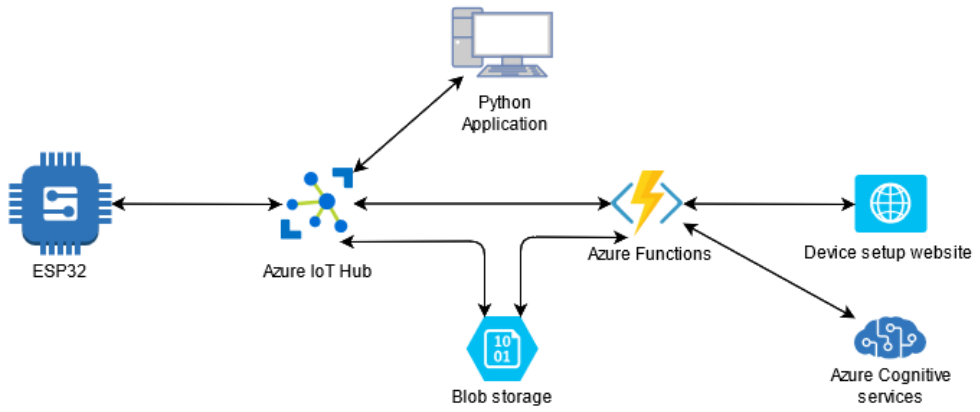
To, že Azure není jen platforma pro IoT zařízení, nám otevírá možnosti v širším spektru než jen správu a obsluhu našeho zařízení. Umožňuje nám to skrze Azure Functions připojit různé další funkce k našemu zařízení. A to nejen pro splnění zadání, ale i přidávání dalších budoucích modifikací bez nutnosti markantně zasahovat do celé architektury.

Microsoft také dodává velkou řadu návodů, ze kterých se dá při vývoji aplikací napojených na Azure IoT Hub čerpat.

Další výhodou je, že jakožto student ČVUT mám možnost si založit studentský účet na Azure a využívat benefity, které Microsoft pro studenty nabízí.

6.1 Cloudové řešení

Z repertoáru Microfost Azure jsem se rozhodl využít funkce popsané v diagramu 6.1.



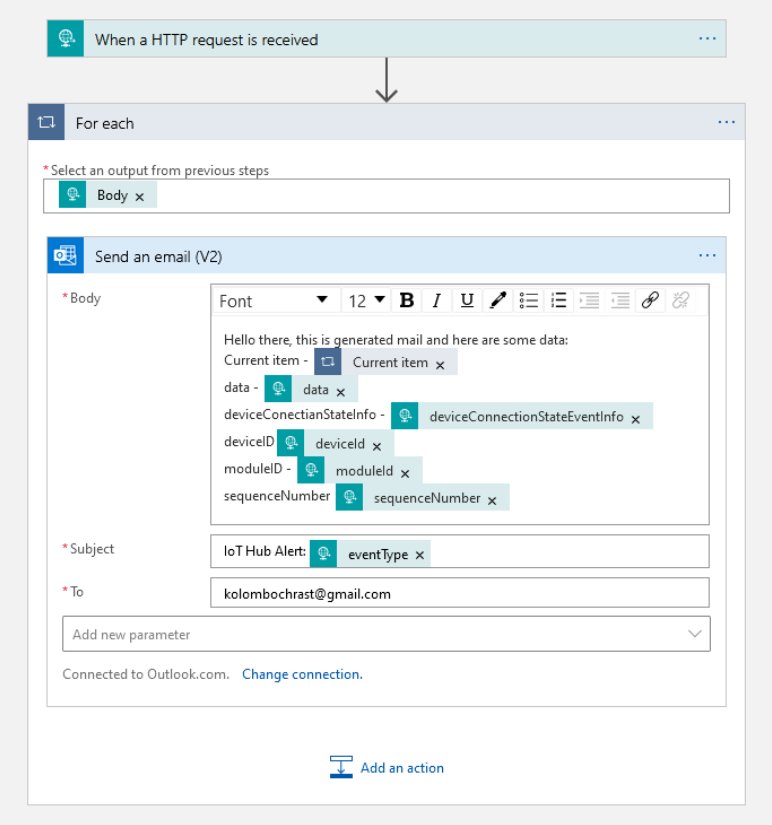
Obrázek 6.1. Schéma architektury azure.

Základním stavebním kamenem celé architektury je Azure IoT Hub. Tato služba umožňuje připojit zařízení do Cloudu a vyměňovat zprávy se zařízením. Co je ale mnohem zajímavější je, že na Azure IoT Hub můžeme navázat další služby a pomocí triggerů tyto služby spouštět. Zároveň je také možné na IoT Hub přímo navázat úložiště, které automaticky spravuje přijaté zprávy a ukládá je v požadovaném formátu.

Na IoT Hub můžeme napojit buďto takzvanou Azure Functions, případně Logic Apps, které budou reagovat na přednastavené triggery. Obě dvě funkcionality jsem vyzkoušel. Logic apps umožňují vytvářet komplexní propojení napříč Azure funkcemi, které se dají nastavovat buď skrze Logic App Designer, případně přímo skrze .json kód.

Pro vyzkoušení Azure Logic App funkcionality jsem se rozhodl vytvořit jednoduchý proces, který bude zrcadlit stav zařízení do mailového klienta. Logic App je připojena

na Azure IoT Hub, který generuje upozornění podle chování zařízení. Logic App tyto zprávy bere a následně je posílá díky propojení s Outlook API do určené destinace. Tato Logic App není jen obyčejné zrcadlení zpráv, které zařízení vysílá, ale může i informovat uživatele to tom, zda bylo zařízení připojeno, či odpojeno od IoT Hubu.



Obrázek 6.2. Náhled návrhu logic app.

Funkce Logic Apps jsou vhodné, pokud chceme automatizovat nějaký proces v příjemném uživatelském rozhraní. Pokud však chceme něco sofistikovanějšího, případně vykonávat specifický kód, je třeba využít druhé zmiňované funkce a tou je Azure Functions.

Azure Functions jsou více techničtější z toho pohledu, že většina funkcionality musí být naprogramovaná uživatelem. Azure Functions podporují několik jazyků od nativního C#, po Javu či Python. Tím, že se jedná o spustitelný kód je možné tyto funkce testovat i lokálně na zařízení bez nutnosti napojení na Azure. Co Azure Functions dále nabízí je takzvaná *Server-less* exekuce kódu. To znamená, že o daný server, na kterém se kód vykonává se já, jakožto správce starat nemusím.

Azure Functions jsem chtěl použít pro připojení internetové stránky určené k tomu, aby se skrze ní dalo zařízení nastavovat. Bohužel jsem však v tomto úkonu nedostal k tíženému výsledku. Zkoušel jsem primárně uspět za pomocí skriptovacího jazyka C# script, který, jak již název napovídá, je založen na C#. Bohužel jsem při vývoji narazil na problémy v linkování knihoven, které se mi nepovedlo obejít. Snažil jsem se poté tento problém vyřešit přesunutím se do jazyka Python, avšak po prvních nezdarech jsem se rozhodl tuto funkcionalitu dočasně opustit, jelikož jsem nad ní strávil velké množství času a byl jsem nucen pokračovat ve vývoji ostatních funkcí.

Tento nedostatek jsem vyřešil tím, že jsem tuto funkcionalitu přenesl na Python aplikaci umístěnou lokálně, bez využití webové stránky, jakožto zprostředkovatele interakce

s uživatele. O této aplikaci více v sekci 6.2. Toto je však jistě část, ke které se chci v co nejbližší době vrátit a plně ji dokonat.

Poslední popsanou využitou funkcionalitou je Azure Cognitive Services. Z této služby primárně využiji *Text to speech* službu. Skrze napojení na Azure Functions, případně Logic App, budu schopen ze zařízení poslat požadavek do této služby na přeložení textu do mluveného slova. Aby se ušetřil zaplacený výpočetní výkon, tak následně poté, co bude slovo přeloženo, se daná stopa uloží do blob úložiště, kde bude k dispozici ke stažení do daného zařízení.

Blob úložiště využívám z důvodu dobrého poměru rychlosti vůči ceně.

6.2 On-premise řešení

Tato aplikace umožňuje momentálně interagovat se zařízením skrze příkazový řádek a předávat zařízení informace v souladu s komunikací popsanou v 4.2.2. Tento program je založen na příkladu od Microsoftu [37].

V příkazovém řádku si můžeme vybrat z popsaných možností a odeslat danou zprávu do zařízení. Následuje ukázka výpisu příkazového řádku v příkladu přidání léku.

```
PS C:\Users\kolom\Codes\pillhelper-server> py .\C2D.py
PillHelper concept messaging app based on microsoft example
Please select command. Type --help for more info.
--help
This is concept program used for communication with ESP device connected
wia Azure IoT hub. Allowed mesages are "add", "remove", "modify",
"report"
Please select command. Type --help for more info.
add
Enter pill name: New Pill
Enter which partition you want to assign: 5
Input time (format hh-mm) : 11-45
Wanna input aditional time? Y/n Y
Input time (format hh-mm) : 13-05
Wanna input aditional time? Y/n n
Should the pill be taken every day? Y/n Y
Enter number of pills for taking: 2
Sending message number: 0
Please select command. Type --help for more info.
```

Aplikace zatím nemá žádnou větší vnitřní logiku, než vytváření a odesílání zpráv. Nijak nekontroluje uživatelské vstupy, zda-li jsou validní. Pokud by toto měl být finální způsob komunikace se zařízením, bylo by nutné tuto funkcionalitu doprogramovat. To však není princip této aplikace. Tato aplikace měla umožnit testování přijímání zpráv na mém zařízení. Stále platí, že v budoucí fázi vývoje budu upřednostňovat webový prohlížeč, jakožto hlavní zdroj komunikace se zařízením. K této alternativě jsem se uchýlil kvůli mým neúspěchům, které v oblasti implementace skrze webový prohlížeč nastaly.

Budoucnost této aplikace je tedy velice nepotřebná z hlediska celého produktu, jelikož uživatel interagující skrze webový prohlížeč nemusí instalovat dodatečné aplikace a může vše vyřítit z libovolného zařízení schopného spustit webový prohlížeč.

Dále jsem použil ještě jednu Python aplikaci, která je podobně jako první založena na příkladu od Microsoftu [38]. Tato aplikace je připojena na Azure IoT Hub a v reálném čase vypisuje zprávy, které zařízení odesílá. Tato aplikace je primárně využívaná teď při

vývoji jakožto logovací aplikace. Dala by se však modifikovat na aplikaci ukládající dané zprávy a případné serverové úložiště, pokud by o to byla poptávka. Zároveň si ale myslím, že lepší výsledky by byly dosaženy, kdyby tato funkcionalita byla implementována plně cloudově. Záleží zde však na požadavcích jednotlivého uživatele.

6.3 Volba v implementaci

Jak již bylo zmíněno, finální implementace by měla být plně cloudová s případnou možností on-premise zařízení. Veškerá funkcionalita by měla být zprostředkována skrze Azure služby, které umožní dobrou podporu budoucích modulů.

Zároveň velice příjemný bonus je ekonomický. Azure umožňuje *Pay as you go* možnost, která umožňuje platit skutečně jen za výpočetní výkon, který se využívá a případně tento výkon škálovat dle potřeby. Více o tomto tématu v ekonomické sekci.

Ačkoliv momentální stav není plně cloudový z důvodu problémů při vývoji, tak věřím že v budoucnu se zkušenostmi, které jsem díky tomuto projektu nabyl, se k problematice vrátím a nesnáze, se kterými jsem se potýkal, překonám.

Kapitola 7
Ekonomický pohled

Při vývoji jakéhokoliv produktu je důležitá výrobní cena a zdali je tento produkt udržitelný. Bohužel nemám dostatečnou kvalifikaci na to, abych zhodnotil všechny risky a hrozby pro toto zařízení a navrhl cenu, která bude odpovídat všem výdajům. Pokusím se však ukázat co nejvíce cenu ovlivňujících atributů, kterým alespoň trochu rozumím.

7.1 Výrobní cena

Jak už jsem zmínil dříve v tomto dokumentu, s návrhem vlastních desek mám pramalé zkušenosti a tak zde můj odhad rozhodně není úplný. Pokusím se však alespoň zhruba cenu odhadnout. Budu vše počítat pro zařízení s pěti přihrádkami.

Vycházím z předpokladu, že celá, potencionálně navržená deska bude zhruba stejně drahá, jako deska, kterou momentálně využívám pro fázi vývoje. Tento předpoklad jsem si dovolil učinit, protože věřím, že naše deska nebude obsahovat všechny komponenty, jako například micro USB port, tlačítka... Zároveň jsem si ale jist, že celkový objem těchto vývojových desek mnohonásobně předčí objem desek, ze kterých bych byl připraven vytvářet daná zařízení. Tím pádem zaplatím více na jednotlivých komponentech, avšak jich budu mít méně. Navím, zda je tato logika legitimní, avšak předpokládám, že se tímto dostaneme na cenu zhruba $5

Po sečtení výdajů za tlačítka, rotační enkodér, 7 degmentový displej a jeho řadič, solenoidy a jejich posuvný registr, megnetické senzory a jejich multiplexor, reproduktor a zesilovač. Dostanu se na částku okolo $25. Jedná se o velice hrubý odhad na základě cen, které jsem nalezl na čínských obchodech. Zároveň je toto cena, kterou jsem zhruba zaplatil, když jsem vše kupoval po jednotlivých kusech za účelem vývoje zařízení.

Celková částka za komponenty by šla mnohem níže poté, co bychom nakupovali ve velkém množství.

Ale cena samotných komponent není jediný faktor, který rozhoduje o ceně zařízení. Produkty se musí někde skladovat, musí být k dispozici náhradní materiál, vše se musí dopravit. Zároveň je třeba počítat s případnou výrobní linkou a poskládáním celého produktu. O plastové krabičce nemluvě.

Věřím, že o tom, jak funguje tento svět výroby mám největší nedostatky a pokud by se uskutečnil finální vývoj, budu zde potřebovat nutně pomoc dalšího člověka, který se v této oblasti vyzná.

7.2 Cena Azure služeb

Hlavním argumentem, který v dnešní době hraje proti cloudovým službám je jejich cena. V mém případě jsem využil Azure pricing calculator [39], díky kterému jsem mohl dobře odhadnout cenu.

Azure nabízí většinou dvě možnosti, jak za své služby platit. Buďto pomocí klasické subskripce, kdy se měsíčně platí daná částka, či možnost platby *pay as you go*. Tato

možnost se podobá obyčejné subskripci v tom smyslu, že platíme za poskytované služby periodicky. Avšak cena není pevně stanovena a odvíjí se od množství využitých služeb. Tento flexibilní systém se dobře hodí pro produkty, jako je ten náš, neboť komunikace, kterou vyžadujeme není nijak frekventovaná. Každá ze služeb je financována specificky.

Pro hrubý výpočet ceny služeb si vezmu fiktivní sociální zařízení, které má 50 pacientů.

Azure IoT Hub má v nabídce *Free tier*, který umožňuje připojit 500 zařízení a doručit 8000 zpráv za den. V počtu zařízení jsme se zde vešli naprosto v pořádku. U zpráv záleží, jak agresivně bychom chtěli stav zařízení reportovat. Zde se naskytuje možnost i nastavit koncovým uživatelům zařízení na míru. Pokud nám o užívání léků stačí jedna zpráva denně, či pokud bychom chtěli mít každý neužitý lék okamžitě nahlášený. Průměrně nám vychází 160 zpráv na zařízení při rovnoměrném rozdělení. Musíme si však uvědomit, že do těchto čísel se musí vejít i případná správa zařízení a nastavování. Tím, ale že těchto 160 zpráv není vázáno pevně na zařízení, je možné s touto množinou zpráv libovolně manipulovat.

Pokud by přeci jen bylo potřeba více zpráv, tak další možností je nabídka *Standard 1*. Zde dostáváme neomezené množství připojených zařízení a 400 000 zpráv za den. Azure IoT hub bohužel nenabízí možnost *Pay as you go*. Verze *Standard 1* by nás vyšla na $25.00 měsíčně.

Další věc, na kterou se dá ještě IoT Hub Využít je takzvaný *IoT Hub Device Provisioning Service* [40]. Tato služba umožňuje spravovat zařízení bez nutnosti lidské interakce, jako je například automatické přidání zařízení do příslušného IoT Hubu. Pokud bychom chtěli tuto službu využívat vyjde nás na $0.10 za každý započatý tisíc operací za měsíc.

Další je služba Azure Functions. Zde je již nabízeno placení v možnosti *Pay as you go*. Tato služba nabízí prvních 400 000 GB/s, či 1 000 000 exekucí za měsíc zdarma. Tato služba by měla zprostředkovávat komunikaci ze strany webového prohlížeče směrem k zařízení. Vzhledem k mé neúspěšné implementaci nejsem bohužel schopen odhadnou přesnou dobu exekuce a velikost vyžadované paměti. Jsem však přesvědčen, že bych se do této stanovené hranice vešel. Samozřejmě by záleželo na vytížení zařízení a jak často by se k nim přistupovalo. Kdybychom však tento limit překročili, bude nás tato služba stát $0.000016 za 1GB/s případně $0.20 za každý další započatý milion exekucí.

K ovládání skrze webovou stránku se případně váže potenciální hosting. I ten jsme schopni skrze Azure zařídit, avšak pro mou imaginární firmu bych spíše doporučil nějakou formu lokálního hostingu, už jen z bezpečnostních důvodů.

Další částí služeb je cloudové úložiště. Zde bych opět využil *Pay as you go* plán, kde budu platit jednak za požadovanou velikost úložiště a zároveň za jednotlivé operace, které budu nad daty vykonávat.

Pro blob úložiště je cena jednoho GB dat $0.02. Musíme k tomu však připočítat i operace s daty. Zápis 10 000 operací stojí $0.050, přičemž čtení stojí $0.004 za stejný počet operací.

Po rychlém výpočtu jsem zjistil, že pro uložení všech zpráv na měsíc bude potřeba maximálně 50MB. Výpočet jsem provedl *velikost jedné zprávy * maximální počet zpráv za den * počet dní v měsíci*, kde uvažuji, že všechny zprávy z IoT hubu byly využity jakožto report stavu zařízení. Pokud se do toho započítá cena za zápis a případné čtení vychází mi cena zhruba na $1.95 měsíčně. Tato cena může ještě vzrůst na základě počtu uložených zvukových stop přeložených kognitivními službami. Pokud by se na tento účel využilo obyčejné úložiště tak zaplatíme méně peněz. V mých hrubých výpočtech $1.65 měsíčně. Přijdeme však o rychlost blob úložiště, které bych chtěl využít pro ukládání *text*

to speech funkcionality. Pokud však zákazník nebude chtít toto využít, bude obyčejné úložiště více než dostačující.

To nás přináší k poslední funkcionalitě a tou je *Text to speech*. Zde je ještě otázkou, jak se bude tato funkce do celého systému integrovat. Pokud by integrace proběhla skrze *Azure Functions* cena by odpovídala této službě, avšak byla by vyžadována větší prvotní investice na vývoj. Druhou volbou je využít *Logic Apps*, ty jsou na propojení jednotlivých Azure komponent přímo uzpůsobeny, avšak jejich cena je vyšší.

Samotná *Text to speech* se odvíjí od počtu charakterů, které si zažádáme zpracovat. *Free tier* umožňuje zpracovat 5 milionů charakterů za jeden měsíc a to pouze jednomu požadavku v jednu chvíli. Pokud by to byla limitace, je zde *Standard tier*, který umožňuje zpracovat 100 současných požadavků za cenu $4.00 za milion charakterů. Pouze jeden požadavek v jednu chvíli by pro naši aplikaci neměl být problém, neboť přeložené texty budou uchovány v blob úložišti a nemělo by být tedy nutné tuto službu využívat pro každý překlad.

Díky tomu, že naše čísla jsou dostatečně malá, naše aplikace využije z velké části *Free tier*y a umožní to celou cenu správy držet nízko.

7.3 Po vývojová fáze produktu a jeho budoucnost

Samozřejmě pouze služby Azure nezajistí veškerou podporu produktu. Je třeba počítat i s náklady, které přijdou po celém vývoji a implementaci. Na případné aktualizace a dodatečné moduly. Vzhledem k tomu, že zařízení má být modulovatelné, tak i následná podpora pro koncové uživatele a pomoc s nasazením.

Dokáži si představit, že tento produkt by mohl fungovat na dvou úrovních. Jedna je ta, že se prodá produkt uživateli a druhá by byla forma subskripce, která by zajišťovala, že by uživatel dostal plnou podporu a případné starání o administraci zařízení a uživatelů.

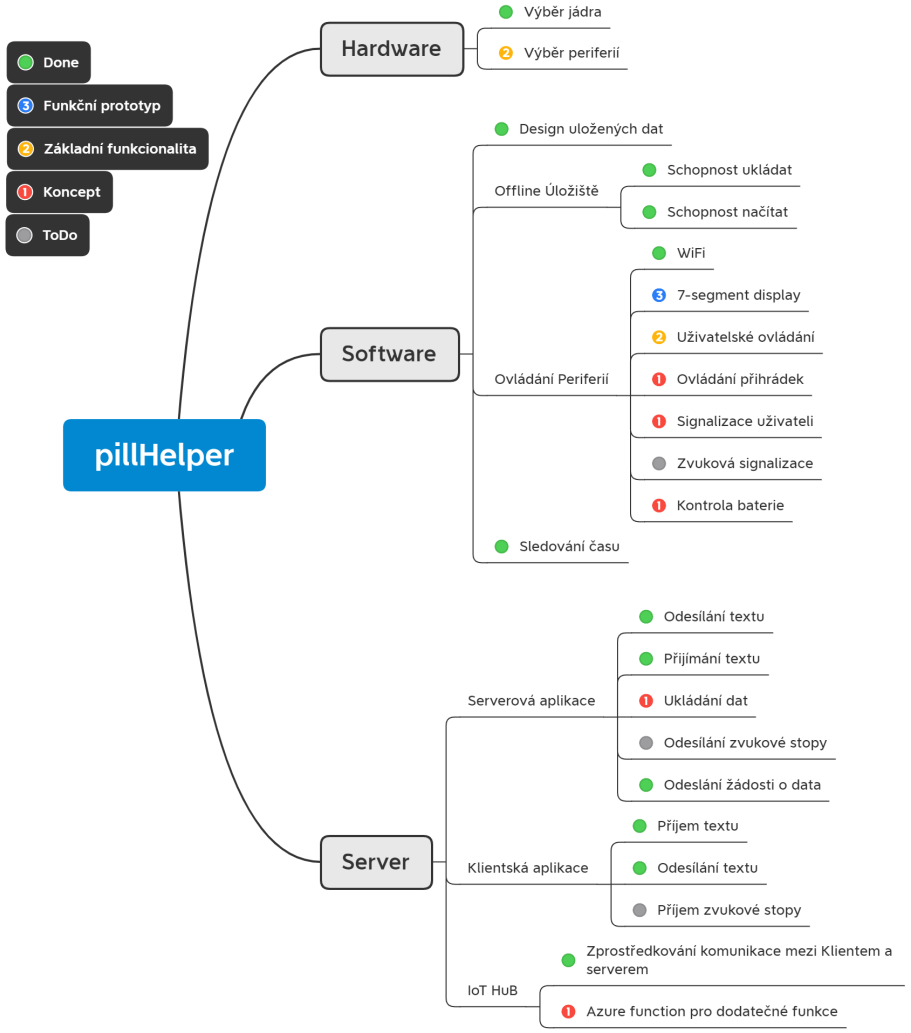
Nevím, jak moc by tato případná subskripce musela být vysoká, aby to pokrylo všechny náklady a zdali nebude vysoká příliš. Pokud by se tak stalo. Bylo by možné s mým produktem oslovit větší firmu a té produkt nabídnout.

Kapitola 8
Závěr

8.1 Vývoj produktu

Celý dosavadní vývoj by se dal rozdělit volně do dvou částí. První část byla celkové seznámení se všemi moduly, které budu potřebovat. Na ukázkových příkladech jsem modeloval periferie a služby, které jsem měl v plánu ve finálním zařízení implementovat.

Po obeznámení se s jednotlivými komponenty jsem pokračoval již samotnou implementací mého konceptu. Verze, která je momentálně dostupná je zatím jen pouze základ, který se bude následně rozrůstat. Momentální stav můžeme vidět na obrázku 8.1



Obrázek 8.1. Vizualizace stavu jednotlivých komponent.

Tuto myšlenkovou mapu jsem využil, abych mohl vizualizovat můj dosavadní postup a kam bude mé úsilí směřovat v následujícím vývoji. Nejvíce práce stále zůstává v soft-

waru samotného zařízení, kde je třeba doprogramovat interakce s periferiemi a rozšířit možnost ovládání.

Jakmile bude hotová základní funkcionalita a interakce s periferiemi, přišel by čas na vytvoření základního prototypu krabičky. Pro tento krok bych využil možnosti 3D tisku, abych mohl vytvářet prototypy, na kterých budu moci testovat celé rozložení produktu a jeho celkovou funkčnost. Ověřit, zda mé koncepty skutečně umožňují pohodlné ovládání a případně se poté vrátit k návrhu a některé detaily přehodnotit.

Další cíl je znovu navštívit problém Azure Functions a vyřešit komunikaci skrze webový portál. Následně bych zůstal u Azure Functions a zaměřil se na implementaci kognitivních služeb.

Následovat bude testování celého produktu, co se správné funkcionality týče. Vzhledem ke komplexnosti celé aplikace bude tato část dle mého názoru dosti časově náročná.

Po vhodném otestování bych byl schopen tuto fázi nazvat jako minimální produkt. Následovat tomu bude vytvoření návodů, a testování na skutečných uživatelích, případně vývoj dalších modulů a různých variant zařízení.

8.2 Zhodnocení práce

Tento projekt mi skutečně pomohla pochopit, jaké úsilí stojí za produkty, které každý den využíváme. Jaká je asi jejich cesta. Jak je nelehká, komplexní a co vše se musí na cestě za finálním produktem uskutečnit, abychom my, koncoví uživatelé, mohli tyto produkty využívat.

V rámci práce se mi povedlo analyzovat celý problém dávkovače léků. Provedl jsem rozbor trhu a zaměřil se na specifický problém bezpečnostní lékovky. Tento produkt jsem zanalyzoval a nabídl několik možností přístupu k řešení. Vybraná řešení jsem popsal a pokusil se navrhnout optimální přístup a zmapovat potencionální risky, které je třeba nutné vyřešit.

Následně jsem začal pracovat na konceptuálním zařízení. Otestoval jsem všechny nutné funkcionality, které jsou pro zařízení třeba z hlediska návrhu softwaru a hardwaru. A zmapoval řešení pro všechny úkony, které zařízení musí vykonávat.

Tyto úkony jsem promítl do kontextu koncového uživatele a pokusil se zanalyzovat problém uživatelské zkušenosti a navrhnout potencionální cestu, jak komunikaci zařízení-uživatel udělat co nejpříjemnější.

V komunikaci zařízení s ostatními službami jsem implementoval cloudové služby Microsoft Azure. Zde se bohužel nepovedlo implementovat veškeré plánované funkce, avšak v rámci konceptuálního zařízení jsem navrhl dočasné řešení, které je pro vývoj zařízení dostačující. Další práce v tomto odvětví je však nutná.

Celkově bych zhodnotil práci jako částečně úspěšnou, povedlo se mi vytvořit konceptuální zařízení, které má dobrý základ na to, aby nad ním mohl vyrůst finální produkt. Než se tak však stane, bude třeba dodatečný vývoj této aplikace.

Literatura

[1] Lékárna BENU. *DEPAN Dávkovač léků týdenní*.
https://www.benu.cz/depan-davkovac-leku-tydenni?aw=1&gclid=Cj0KCQiA9P__BRC0ARIsAEZ6irgLFg1K-5Q02tZeeoXUmwtnLlhmLtS8XgF625Kr6aB2cqUjCPzrFPAaAvm1EALw_wcB.

[2] kitos. *Dávkovač na léky - 7 dní, ráno/večer*.
https://www.kitos.cz/davkovac-na-leky---7-dni-ranovecer/159386p/.

[3] Lékárna - lékárna.cz. *Dávkovač léků OBZOR typ 02 týdenní zelený*.
https://www.pilulka.cz/davkovac-na-leky-anabox-1x7?gclid=Cj0KCQiA9P__BRC0ARIsAEZ6irjhjI67IrbO0wZ3tfxeGqBgxI3_aziVnpqDGiuf_aKGB9hHP3lSIaAv21EALw_wcB.

[4] poštovnézdarma.cz. *Krabička na léky KNL09*.
https://postovnezdarma.cz/321753-krabickanaleky-knl09?gclid=Cj0KCQiA9P__BRC0ARIsAEZ6irhndiYfQBriDBu9lqyWjX9KTZb5mW8yzsI9G8VVyg8RmThu4UpN_7YaAqZREALw_wcB.

[5] dárekv akci. *ISO 10972 Dávkovač léků s alarmem*.
https://www.darekvakci.cz/iso-10972-davkovac-leku-s-alarmem?gclid=Cj0KCQiA9P__BRC0ARIsAEZ6irgjGlr6UKxx-64QoPIPzL6SerB1vj4KwoHkbBNWDxDOpWDK9mhdB6waAkAqEALw_wcB.

[6] velký košík. *Elektronická inteligentní krabička na léky Pilly*.
https://www.velkykosik.cz/telo-a-zdravi/chytry-davkovac-leku/.

[7] tricella. *Pillbox by Tricella*.
https://www.tricella.com/.

[8] Tinylogics. *Memo Box Vibrant*.
https://pillbox.tinylogics.com/collections/all-product.

[9] home8alarm. *28-slot Medication Dispenser with Lock*.
https://www.home8alarm.com/store/.

[10] Espressif Systems. *ESP32 datasheet*.
https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[11] Espressif Systems. *ESP8266 datasheet*.
https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.

[12] Maxim Integrated. *DS3231 Real-Time module. Extremely Accurate I2C-Integrated RTC/TCXO/Crystal*.
https://datasheets.maximintegrated.com/en/ds/DS3231.pdf.

[13] Adafruit Industries. *RTClib.h - A fork of Jeelab's fantastic RTC library*.
https://github.com/adafruit/RTClib.

[14] GM ELECTRONIC. *Modul RTC, I2C a 32kb flash ZS-042 / DS3231*.
https://www.gme.cz/modul-rtc-ds323-i2c-a-32kb-flash.

[15] Espressif Systems - Paolo Becchi. *EEPROM.h - library for ESP32 based systems.* https://github.com/espressif/arduino-esp32/tree/master/libraries/EEPROM.

[16] Ivan Grokhtkov Espressif Systems - Hristo Gochkov. *SPIFFS.h - ESP32 SPIFFS File System.* https://github.com/espressif/arduino-esp32/tree/master/libraries/SPIFFS.

[17] Veronika Mártonová. *Electropermanent Magnet Study*. 2019. https://dspace.cvut.cz/bitstream/handle/10467/79592/F3-BP-2019-Martonova-Veronika-Electropermanent_Magnet_Study.pdf?sequence=-1&isAllowed=y.

[18] Laskarduino. *4.0" 480x320 TFT displej, ST7796, SPI, dotykový.* https://www.laskarduino.cz/4-0--480x320-tft-displej--st7796--spi--dotykovy.

[19] UNISYSTEMS. *WF70ATIAGDNC0 - LCD-TFT display from Winstar Co. (7.0 inches, 800x480, no controller, capacitive touch screen).* https://www.unisystem-displays.com/en/products/lcd-tft/standard/wf70atiagdnc0.html.

[20] Adafruit Industries. *Red 7-segment clock display - 0.56" digit height.* https://www.adafruit.com/product/865.

[21] Matthias Hertel. *Use a rotary encoder with quadrature pulses as an input device.* https://github.com/mathertel/RotaryEncoder.

[22] Wikipedia. *Lithium iron phosphate battery.* https://en.wikipedia.org/wiki/Lithium_iron_phosphate_battery.

[23] Wikipedia. *Lithium polymer battery.* https://en.wikipedia.org/wiki/Lithium_polymer_battery.

[24] Wikipedia. *Adafruit Feather M0 WiFi.* https://www.adafruit.com/product/3061.

[25] DFRobot. *FireBeetle ESP32 IoT Microcontroller.* https://www.dfrobot.com/product-1590.html.

[26] Wikipedia. *Nickel–metal hydride battery.* https://en.wikipedia.org/wiki/Nickel-metal_hydride_battery.

[27] Damien P. George. *MicroPython.* https://micropython.org/.

[28] Gordon Williams. *Espruino.* https://www.espruino.com/.

[29] originally developed by Richard Barry Open source with multiple contributors. *freeRTOS.* https://www.freertos.org/.

[30] Microsoft. *Azure IoT Device Workbench.* https://marketplace.visualstudio.com/items?itemName=vsciot-vscode.vscode-iot-workbench.

[31] Microsoft. $ESP32_AzureIoT - AzureIoTHublibraryforesp32devicesinArduino.$ https : //github.com/VSChina/ESP32_AzureIoT_Arduino.

[32] tablatronix tzapu. WiFi Connection manager with fallback web configuration portal. https://github.com/tzapu/WiFiManager.

[33] Wikipedia. Universal Plug and Play. https://en.wikipedia.org/wiki/Universal_Plug_and_Play.

[34] Wikipedia. Wi-Fi Protected Setup.
https://en.wikipedia.org/wiki/Wi-Fi_Protected_Setup.

[35] Yakov Shafranovich. Common Format and MIME Type for Comma-Separated Values (CSV) Files. *RFC 4180. Request for Comments. 2005.*
https://rfc-editor.org/rfc/rfc4180.txt.

[36] Espressif Systems. Over The Air Updates (OTA).
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/ota.html.

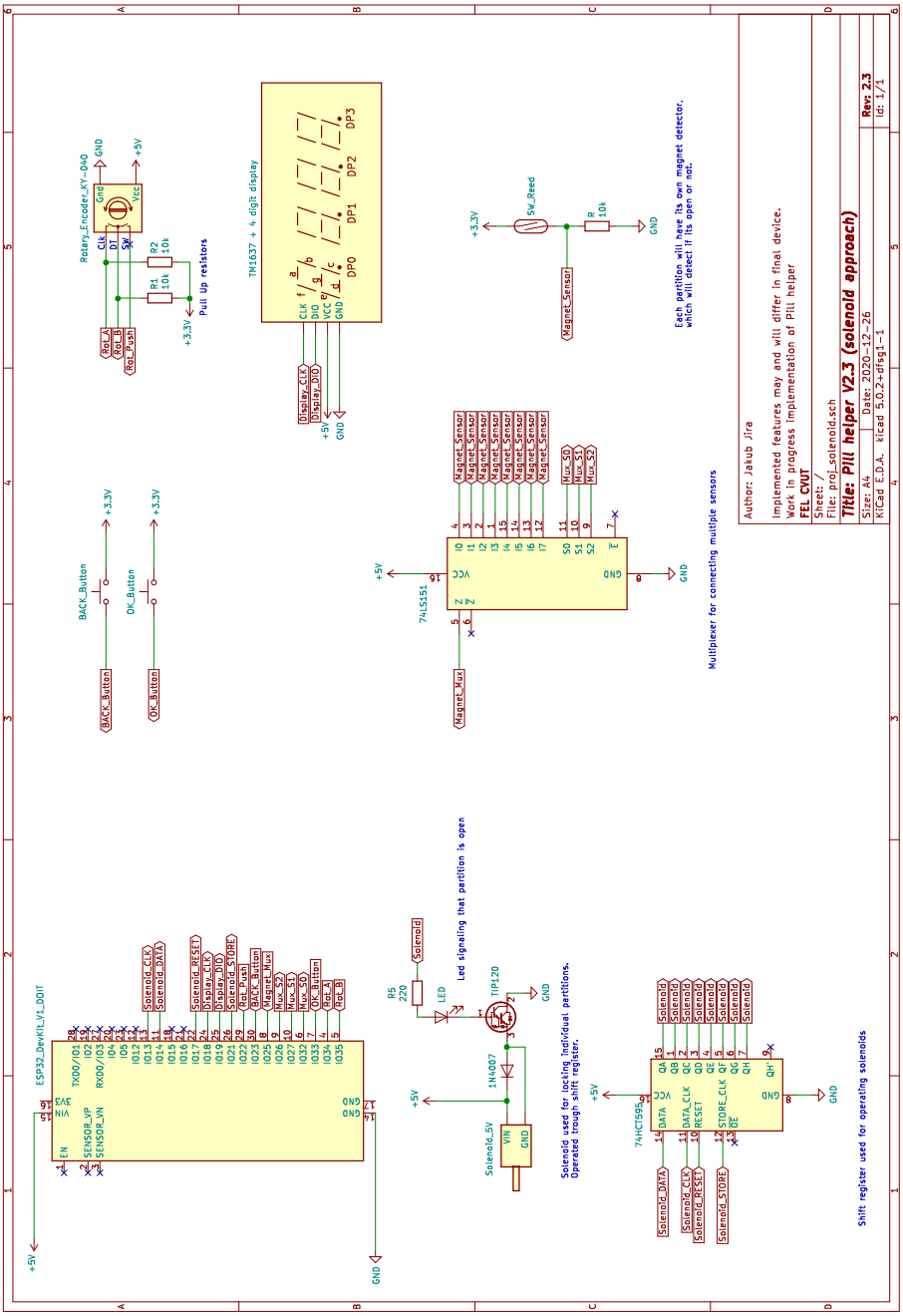
[37] Microsoft. Send cloud-to-device messages with IoT Hub (Python).
https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-python-python-c2d.

[38] Microsoft. Read device to cloud messages (Python).
https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-messages-read-builtin#read-from-the-built-in-endpoint.

[39] Microsoft. Azure Pricing Calculator.
https://azure.microsoft.com/en-us/pricing/calculator/.

[40] Microsoft. What is Azure IoT Hub Device Provisioning Service?
https://docs.microsoft.com/en-us/azure/iot-dps/about-iot-dps.

Příloha A
Zvětšené KiCad schéma



Obrázek A.1. Schéma zapojení základních periferií v KiCadu