VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

UNIVERZÁLNÍ PROGRAMOVATELNÁ SENZORICKÁ PLATFORMA

UNIVERSAL SENSORIC PLATFORM WITH PROGRAMMABLE FEATURES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. DAVID GÁL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÁCLAV ŠIMEK**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/20213/2016/xgalda01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Gál David, Bc.**
Obor: Počítačové a vestavěné systémy
Téma: **Univerzální programovatelná senzorická platforma**
Universal Sensoric Platform with Programmable Features
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s požadavky kladenými zadavatelem na programovatelnou senzorickou platformu a proveďte jejich podrobnou analýzu.
2. Nastudujte technické parametry a způsob práce s WiFi modulem ESP-WROOM02. Dále zvažte možnosti integrace interpretu jazyka Lua ke stávajícím vývojovým nástrojům.
3. Návrhněte a následně proveďte realizaci testovacího přípravku pro sběr senzorických dat (teplota, tlak, atd.) a jejich přenos s využitím standardu IEEE 802.11.
4. Připravte implementaci firmware umožňujícího konfiguraci připojení senzorického modulu, stažení a spuštění uživatelského kódu a logování naměřených dat.
5. Vytvořte aplikaci pro serverovou část systému programovatelné senzorické platformy, která umožní skrze grafické rozhraní intuitivní tvorbu uživatelského kódu.
6. Proveďte podrobné ověření funkčnosti vytvořeného řešení. Zhodnoťte dosažené výsledky, pokuste se navrhnout případná rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Šimek Václav, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Práce se zabývá tvorbou zařízení určeného pro sběr senzorických dat ve výrobním prostředí. Zařízení je schopné sbírat data z různých typů senzorů, ale také řídit exerní systémy prostřednictvím vstupně výstupních portů. Sběr dat a řízení jsou ovládány skriptovacím jazykem Lua. Skripty jsou vytvářeny vzdáleně v uživatelsky přívětivém vizuálním programovacím prostředí Blockly. Jednotka je založana na WiFi modulu ESP-WROOM-02.

Abstract

The aim of this thesis is to design universal sensoric platform for data collection in manufacturing environment. Device is capable of collecting data from variety of sensors, but it is also able to control external systems. Platform is controled by Lua scripts. The scripts are generated in user friendly visual programming environment Blockly. Platform is based on WiFi module ESP-WROOM-02.

Klíčová slova

WiFi, senzorická jednotka, esp8266, ESP-WROOM-02, Lua, sběr dat

Keywords

WiFi, sensoric platform, esp8266, ESP-WROOM-02, Lua, data collection

Citace

GÁL, David. *Univerzální programovatelná senzorická platforma.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Šimek Václav.

Univerzální programovatelná senzorická platforma

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Václava Šimka. Další informace mi poskytli zaměstnanci firmy ALPS Electric Czech, s.r.o. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Gál
24. května 2017

Poděkování

Chtěl bych poděkovat panu Ing. Josefovi Nevrlému a Ing. Marku Gálovi za odborné vedení, za pomoc a rady při zpracování této práce a dále pak panu Ing. Václavu Šimkovi za odborné rady.

Obsah

Seznam tabulek

Seznam obrázků

Kapitola 1

Úvod

Většina výrobních prostředí je v dnešní době alespoň částečně automatizována. Mimo zrychlení, zefektivnění a zkvalitnění výroby s sebou automatizace přinesla i ulehčení sběru dat týkající se toku materiálů. Díky tomu je možné zredukovat náklady spojené se skladováním a nákupy. V určitém případě jsou přístroje schopné sledovat i jednotlivé vyráběné kusy a umožňují tím zpětnou dohledatelnost např. v případě výroby vadného kusu. I když se jedná o relativně obsáhlé množství dat, je pokryta jenom malá část výrobního procesu. Z hlediska dalšího zefektivnění výroby je žádoucí sledovat také okolí výrobních linek i linky samotné. Jde například o sledování odběru energie strojů. Monitorování může také pomoci předcházet selhaním strojů, případně usnadnit diagnostiku pokud už k selhání došlo.

Hlavní překážkou plošného sledování výroby jsou pořizovací náklady monitorovacích jednotek. V poslední době se s rostoucí popularitou tzv. Internetu věcí (IoT) začalo objevovat velké množství levných a výkonných komponent. Vzhledem k tomuto se otevřela cesta k realizaci vyhovující nízkonákladové senzorické jednotky. Náplní této práce není vytvoření finální verze měřící jednotky, ale spíše ověření realizovatelnosti celého konceptu. Určité části by však měly být tvořeny s ohledem na využití v možném finálním výrobku. Práce je zadána společností ALPS Electric Czech, s.r.o.

Důvodem, proč nelze využít některé existující řešení, jsou specifické požadavky kladené na platformu. Jednotka by měla být schopná zaznamenávat data z různých druhů čidel (el. proud, teplota, aj.) a tato data bezdrátově ukládat prostřednictvím sítě WiFi do centrální databáze. Různé senzory vyžadují různé předzpracování dat v jednotce a odlišný bude také požadavek na četnost datových záznamů podle typu čidla. Sběr může být prováděn buďto v určitých časových intervalech, ale také na základě externího impulsu, ať už z externího zařízení nebo od operátora u linky. Dále by kromě prostého sběru dat mělo zařízení, na základě naměřených hodnot, být schopno ovládat externí zařízení. Příkladem může být zapnutí chlazení pokud zjistí, že došlo k nárůstu teploty v měřeném systému.

Požadavky na pokročilé funkce jednotky vyžadují netriviální konfigurační rozhraní. Konfigurace prostřednictvím menu by byla příliš limitující. Změna firmware jednotek je naopak značně nepraktická a vyžaduje hluboké znalosti o jednotce. Firma se rozhodla využít poznatků z jiné vlastní aplikace, která k obdobným účelům s úspěchem využívá zabudovaného interpretu skriptovacího jazyka uvnitř konfigurované jednotky a kód skriptu je generován v grafickém webovém rozhraní. Do senzorické platformy tak bude potřeba zabudovat interpret skriptovacío jazyka a k tomu provést odpovídající modifikace konfiguračního rozhraní. Skript by měl být do jednotky nahrán bez nutnosti fyzického přístupu, tedy prostředníctvím bezdrátové sítě.

V první části práce jsou blíže shrnuty a rozebrány veškeré požadavky zadávající společností ALPS na výslednou platformu. Druhá část teoreticky popisuje součásti nezbytné k návrhu platformy. Následuje samotný návrh a realizace hardware testovacího přípravku.

Ve zbývajících kapitolách je navržen a jsou realizovány hlavní části firmware senzorické jednotky a také programové vybavení serverové částí zahrnující konfigurační prostředí a uložiště naměřených dat. Serverové vybavení vychází z již existujícího řešení, které je ve firmě nasazeno.

Kapitola 2

Specifikace požadavků zadavatele

Kapitola blíže popisuje požadavky zadavatele na výslednou platformu a poukazuje na problematické části návrhu. Na konci jsou požadavky shrnuty do několika bodů.

2.1 Podrobný popis

Sekce zahrnuje podrobný popis požadavků kladených na platformu.

Sběr dat

Základním požadavkem na senzorickou platformu je schopnost sbírat data. Jak již bylo řečeno v úvodu, jednotka by měla umožňovat připojení různých druhů senzorů. Podporovány by měly být senzory s digitálním i analogovým výstupem.

K testování platformy byly vybrány 3 typy senzorů. Jsou to senzor:

- **Teploty** okolního prostředí
- **Proudu** odebíraného výrobními linkami – Linky jsou napájeny ze standardní 230V rozvodové sítě. Nejedná se o elektricky výkonná zařízení charakteru těžkého průmyslu. Pro pokusná měření se nepředpokládá překročení hodnoty 30A. Senzor by mělo být možné použít bez nutnosti přerušení vedení.
- **Tlaku** externě přiváděných plynů a kapalin do výrobních zařízeních nebo atmosférického tlaku ve výrobních halách

Kromě hardwarové podpory bude zapotřebí realizovat také dostatečně flexibilní programové vybavení jednotky. Může být např. zapotřebí odebrání velkého množství vzorků v krátkém čase s tím, že relevantní je pouze průměrná hodnota z nich. Přenos celého datového záznamu by mohl mít negativní dopad na propojovací síť.

Důležítá je také možnost volby frekvence odebírání dat podle druhu senzoru/pozici jednotky ve výrobním prostředí (např. jednou za minutu/hodinu).

Propojení externích systémů

Platforma nemá sloužit pouze jako pasivní prvek, ale má být schopna ovlivňovat svoje okolí. To vyžaduje schopnost zpracovávat výstupní signály, nebo spínánat externí zařízení.

Vstupy mohou být použity jako iniciátory odběru datových vzorků, nebo mohou sami sloužit jako zdroj dat. Příkladem může být detekce korektního uzavření dvířek přístroje.

Výstupy mohou reagovat na určitý naměřený stav (nárust teploty nad určitou mez = zapnutí chlazení), případně mohou z jednotky udělat samostatný testovací přípravek, který bude spouštět zařízení a zároveň monitorovat jeho průběh.

Většina nainstalovaných systémů ve výrobním prostředí ALPS má dvoustavové digitální rozhraní a je uzpůsobeno pro napětí v rosahu 5-24 V. Měřící platforma má v sobě zahrnovat 3 vstupní a 3 výstupní linky akceptující uvedený rozsah napětí.

Mobilita

Komunikace s jednotkami má probíhat prostřednictvím WiFi. Důvodem proč bylo zvoleno právě WiFi je dostupnost levných a výkonných modulů a dobré pokrytí výrobního prostředí sítí WiFi. Volba v této práci padla na populární modul ESP-WROOM-02 [5]. Cena modulu se pohybuje okolo $4 [35] za kus. Obrovská výhoda modulu je jeho soběstačnost - dokáže fungovat jako hlavní řídicí jednotka.

Pro úplnou volnost při umístnění jednotky je třeba eliminovat ještě závislost na zdroji elektrické energie. Zařízení by tak mělo být schopno běžet ze zabudovaného akumulátoru.

Spojení bateriového napájení a WiFi není právě typycký požadavek. Úspornost rozhodně nepatří k přednostem WiFi. S problémem se bude muset vypořádat firmware jednotky vhodným využitím úsporných režimů wifi modulu.

Uživatelské rozhraní

Zařízení by mělo zobrazovat základní stavové informace na zabudovaný OLED displej a k minimální možnosti interakce má sloužit jedno kapacitní (dotykové) tlačítko. Tlačítko by mělo být umístněno pod zobrazovací část displeje. Pokud by se řešení ukázalo jako funkční, mohlo by mít značný vliv na úsporu místa ve finálním produktu.

Rozměry a cena

Z hardwarového hlediska se jedná především o ověření celého konceptu nízkonákladové, baterií poháněné WiFi platformy. Rozměry ani cena zde nejsou prioritou. Měly by však posloužit jako ukazatel možného konečného řešení.

Firmware

S velkou flexibilitou hardwarového provedení jednotky jsou spojeny i vyšší nároky na programové vybavení jednotky. Velkou překážkou obecného použití platformy by mohlo být konfigurační rozhraní. Konfigurace přes vestavěné menu by znamenalo výrazné omezení možné funkčnosti. Úprava firmware jednotlivých zařízení nabízí největší volnost konfigurace. Je však značně nepraktické - obsluhu zvládne jenom člověk s pokročilými znalostmi programování mikrokontrolérů. Obsluhu by přitom měl zvládnou člověk se základní znalostí programování.

Ve firmě se již setkali z obdobným požadavkem. Jednalo se o nahrazení PLC jednotek vlastní platformou. Problémem se zabýval Pavol Vargovčík ve své bakalářské práci [37]. Výsledkem bylo řešení postavené na integraci interpretu skriptovacího jazyka (python3) do koncového zařízení (Raspberry Pi). Pro maximální uživatelskou přívětivost byl kód generován z grafického webového programovacího prostředí Blockly [15]. Vygenerovaný kód byl

následně automatizovaně nahrán do patřičného zařízení. Stejný princip má být aplikován v této práci. Namísto výpočetně náročného Pythonu má být použit jazyk Lua (obohacený o vlastní funkce podporující snímání dat). Stejně tak bude potřeba modifikovat prosředí Blockly (podpora záznamu dat, generace Lua kódu). A také bude potřeba umožnit automatizované stažení patřičného skriptu měřící platformou. Serverová část této práce má být, pokud možno, integrována do existujícího prostředí.

Aby se při každé změně nakonfigurované sítě nemusel pokaždé přepisovat firmware, musí být možné provést základní nastavení i jinak. K tomu se vybízí použití modulu WROOM-02 jako přístupového bodu (AP) s jednoduchým webovým (HTTP) rozhraním.

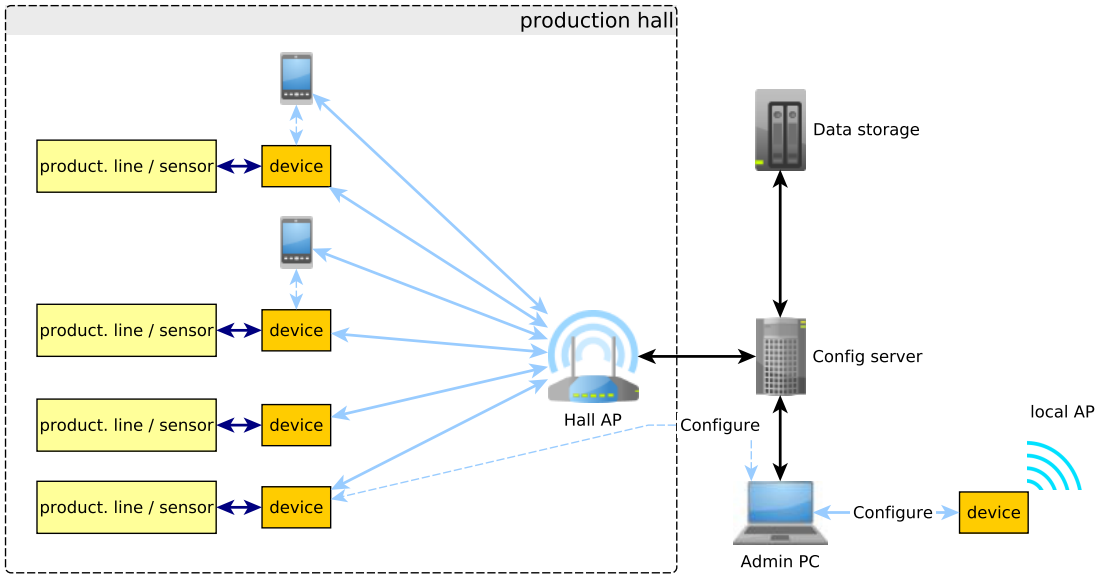
Pokud by to hardware jednotky dovoloval, měla by být ověřena i možnost interaktivního zadávání dat z webového rozhraní operátorem výroby (např. ruční zadání číselné hodnoty). Operátor by se na jednotku připojil prostřednictvím tabletu.

V návrhu je vhodné počítat i s budoucí potřebou modifikovat firmware stávajících jednotek a to jak kvůli odstranění nedostatků tak kvůli rozšíření funkcionality. Protože by mohlo být nasazeno velké množství jednotek, není manuální změna příliš praktická. Firmware by mělo jít měnit bez fyzického přístupu - prostřednictvím sítě ve které budou jednotky připojeny. Obecně je metoda označována jako "over the air"(OTA) aktualizace.

Serverová část

Součástí projektu je také zavedení databázového systému pro ukládání dat a jeho propojení s měřícími jednotkami. Na serveru se bude nacházet výše zmíněné grafické programovací prostředí Blockly.

Možnou hardwarovou konfiguraci systému zobrazuje schéma na obrázku 2.1.



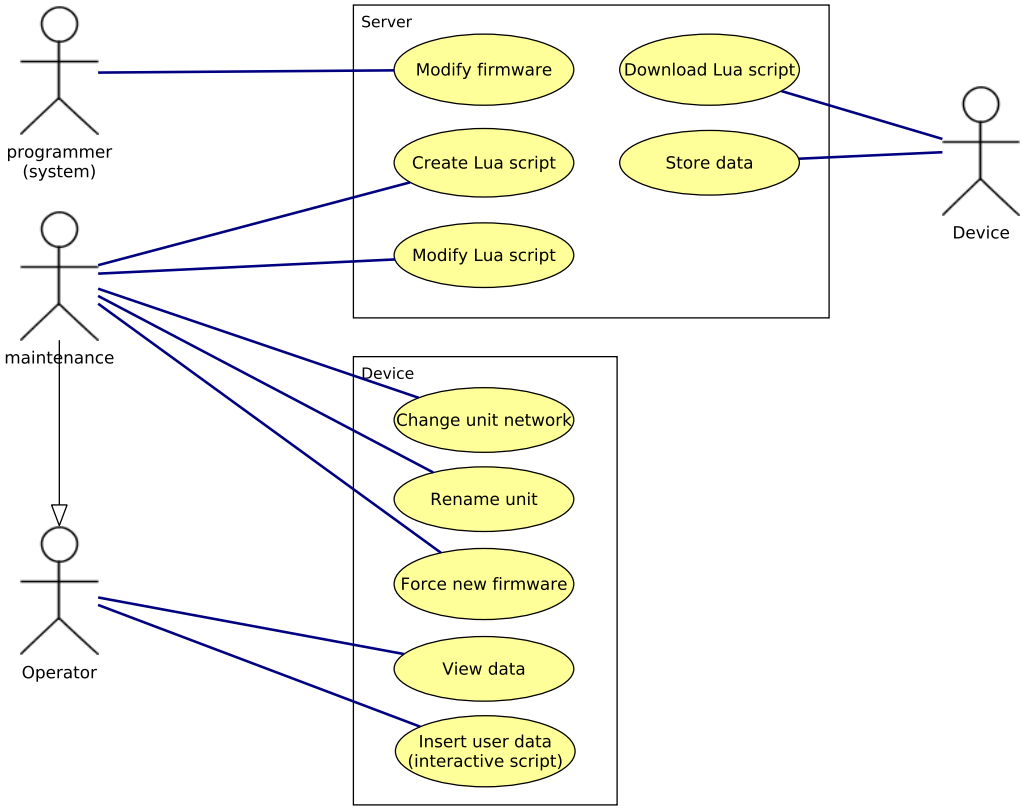
Obrázek 2.1: HW konfigurace systému

2.2 Shrnutí

V kostce lze požadavky shrnout do několika oblastí.

- Sběr dat – Různé typy dat z různých senzorů. Ověření na tlakovém, teplotním a proudovém snímači.
- Propojení externích systémů – 3 vstupní a 3 výstupní digitální rozhraní. Operační napětí 5 - 24 V.
- Mobilita - Komunikace výhradně prostřednictvím WiFi (použití bezdrátového modulu ESP-WROOM-02). Možnost napájení ze zabudovaného akumulátoru.
- Konfigurovatelnost – Zabudování interpretu jazyka Lua a spouštění uživatelských skriptů. Skripty generovány v grafickém programovacím prosředí Blockly.
- Uživatelské rozhraní – 1.3"OLED displej, kapacitní tlačítko zabudované pod displejem a případně uživatelské HTTP na jednotce.
- Vzdálená modifikace firmware (OTA update).
- Rozměry a cena - Ta by měla být co nejmenší, jedná se však o vývojový prostředek v prvotní fázi vývoje. Tento požadavek tedy není natolik kritický.

  Kompletní požadavky užití celého systému zachycuje schéma na obrázku 2.2.



Obrázek 2.2: Diagram užití navrhované platformy.

Kapitola 3

Dostupné prostředky

Kapitola popisuje dostupné hardwarové i softwarové prostředky, včetně dostupných technologií sběru požadovaných dat.

3.1 Hlavní řídicí jednotka

Srdcem celého zařízení se má stát bezdrátový modul ESP-WROOM-02. Jedná se o nízkonákladový bezdrátový Wi-Fi modul založený na ESP8266 SoC (System on Chip) [8] vyráběný firmou Espressif Systems. ESP8266 poporuje Wi-Fi standardy 802.11 b/g/n.

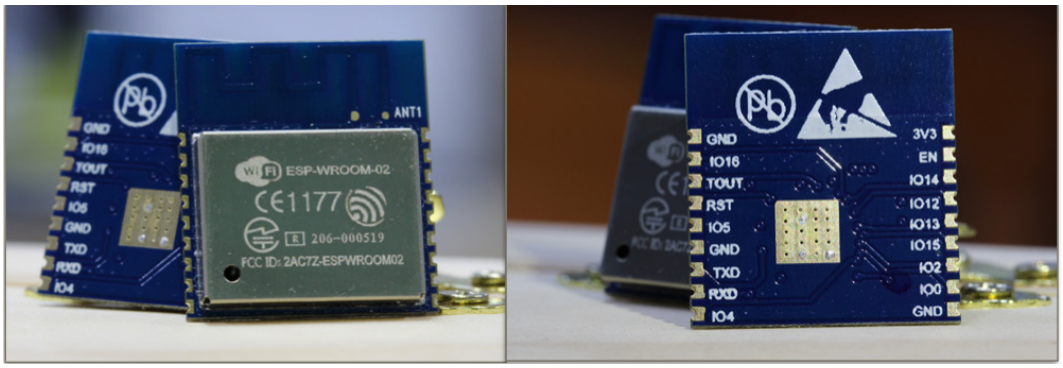
Původně měl ESP8266 sloužit pouze jako serial to WiFi převodník (tedy jako rozšíření konektivity stávajících platforem, aniž by platforma musela řešit WiFi specifikaci). Nedlouho po uvedení výrobku uvolnil výrobce i programové nástroje potřebné ke tvorbě vlastního firmware. ESP8266 je tedy sám schopný kompletně nahradit externí zařízení a fungovat jako hlavní řídicí prvek. V podstatě se tedy jedná o relativně levné MCU schopné nativně komunikovat prostřednictvím WiFi rozhraní. ESP8266 má však i své negativní stránky, tou nejzávažnější je nedostatečná oficiální podpora ať už jde o dokumentaci nebo o vývojové prostředky.

Díky svým možnostem však ESP8266 vzbudil velký zájem veřejnosti, která dokázala tyto nedostatky do určité míry odstranit. Dobrým zdrojem informací se tak stalo komunitní web o esp [14] (fórum, wiki), ale také volně dostupná kniha Neila Kolbana [17].

K výše zmíněnému je třeba dodat, že čip samotný nemůže fungovat zcela samostatně, ale potřebuje ke svému běhu několik externích periferií. Kvůli svým rozměrům (5 * 5 mm) by byl pro většinu nadšenců jenom těžko použitelný. Za velkou část úspěchu tak čip vděčí ESP8266 modulům, které z něj dělají plně funkční samostatnou jednotku. Na trhu je jich dostupná celá řada. Mezi nejrošířenější patří moduly od společnosti Aithinker - dodávané pod označením ESP8266-1—12. Sám Espressif také dodává jeden modul a tím je právě ESP-WROOM-02. Oproti ostatním modulům má dostupnou alespoň základní dokumentaci a navíc je možné jej zakoupit od oficiálních distributorů jako například TME [35]. Většinu ostatních modulů je možné sehnat pouze neoficiální cestou na různých internetových obchodech. Další jeho výhodou oproti konkurenci je prokazatelná certifikace FCC (Federal Comunications Commision) zaručující určité elektromagnetické vlastnosti zařízení.

3.1.1 HW parametry modulu

Ačkoliv je modul ESP-WROOM-02 pouchých 20*18 mm velký, zahrnuje v sobě všechno potřebné k vytvoření soběstačné WiFi jednotky. K esp8266 čipu je v modulu přidán zdroj

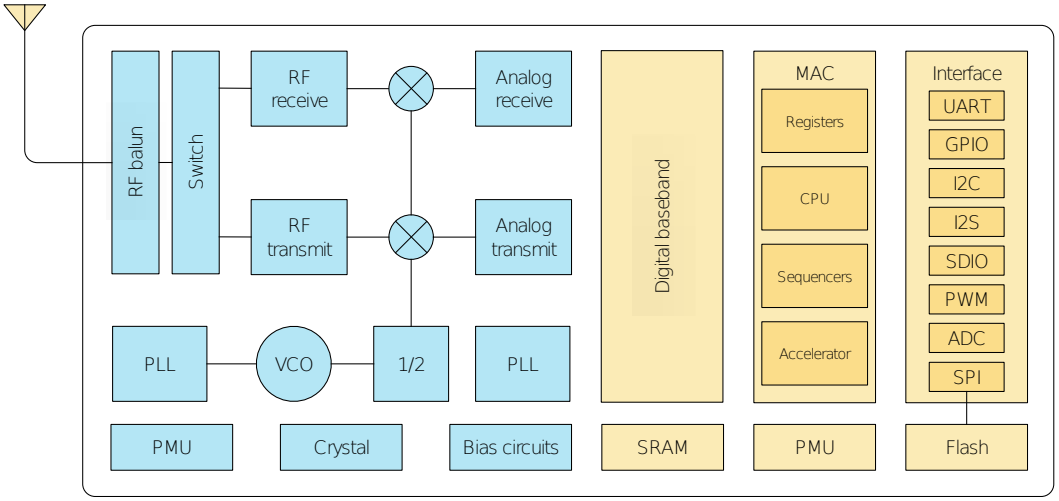


Obrázek 3.1: ESP-WROOM-02 [5]

hodinového signálu, taktovaný na frekvenci 26 MHz, externí paměť typu flash a anténa vyhotovená v podobě obrazce na DPS (komunikace ve frekvenčním pásmu 2.4 GHz).

Externě je potřeba dodat pouze zdroj elektrické energie. Zařízení vyžaduje ke svému běhu napájecí napětí 3.3 V. Udávaná průměrná spotřeba je 80 mA. Ve špičkách (odesílání dat) však může narůst až na 170 mA. Spotřebu je možné snížit využitím některého z úsporných režimů. Ten nejúspornější odebírá pouze 10 µA.

Blokové schéma 3.2 zachycuje veškeré prvky obsažené v tomto modulu. Modře je zvýrazněna analogová (WiFi) část zařízení. Oranžově jsou obarveny digitální prvky.

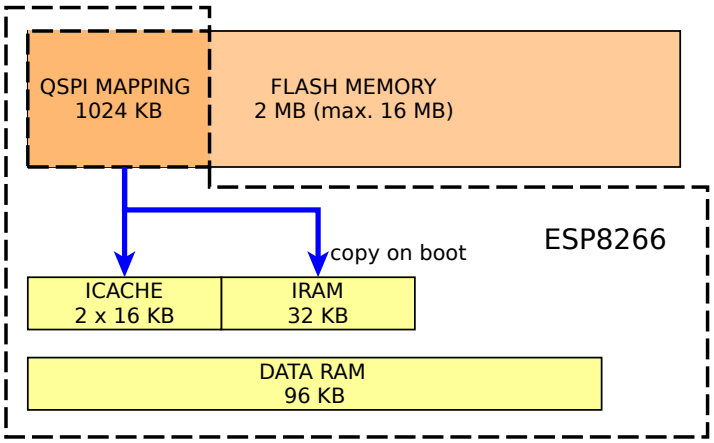


Obrázek 3.2: Blokové schéma modulu WROOM-02 [13]

3.1.2 MCU

O řízení ESP8266 se stará 32 bitový RISC mikroprocesor Xtensa LX106 od společnosti Tensilica. Procesor je v základní konfiguraci taktován na frekvenci 80 MHz, v software je možné jej přepnout až na 160 MHz. MCU disponuje datovou RAM pamětí s kapacitou 96 KB. Kvůli přítomnosti WiFi stacku je však reálně použitelných přibližně 45 KB. Uživatelský kód je ukládán na flash paměť o velikosti 2 MB. Paměť je jedna z periferií dodávaná WROOM-02 modulem, proto je k procesoru připojena přes quad-SPI rozhraní. Flash paměť je možné dále rozšířit opět přes rozhraní SPI až do maximální velikosti 16MB. Kvůli hard-

warovému omezení lze však použít pouze 1 MB paměti pro kód (libovolný 1 MB, nemusí začínat na adrese 0). Nevyužitá část flash paměti je přístupná pro ukládání uživatelských dat. Protože by čtení instukcí přímo z QSPI flash paměti zdržovalo procesor, obsahuje esp8266 rychlou instrukční RAM paměť o velikosti 32 KB a také 2 x 16 KB instrukční cache paměť. RAM paměť funguje jako uživatelem kontrolovaná cache - při startu jsou do ní nakopírovány programátorem zvolené instrukce (značnou část však zabírá WiFi stack).



Obrázek 3.3: Pameť modulu ESP-WROOM-02

3.1.3 Připojení periferií

Pro připojení externich zařízení je uživateli k dispozici 11 vstupně-výstupních pinů. Na vybraných pinech lze aktivovat některý z periferních modulů UART, I2S (2x), PWM (3x), SPI, ADC a IR. Výrobce udává také sběrníci I2C, její hardwarová realizace na esp8266 chybí, takže je nutné ji případně emulovat v software.

3.2 Vývojové nástroje

Kapitola popisuje dostupné vývojové nástroje pro modul ESP-WROOM-02.

V základu je modul dodáván s firmware dovolujícím ovládat WiFi zařízení prostřednictvím AT příkazů [7], které jsou do modulu odesílány prostřednictvím rozhraní UART. Ten však není pro tuto práci příliš podstatný, protože umožňuje využití ESP modulu pouze pouze coby externího WiFi modulu bez možnosti spouštění vlastního kódu. Dá se však využít k snadnému otestování WiFi funkcionality na čipu. Zajímavější jsou softwarove prostředky pro tvorbu vlastního programového vybavení. Existuje také řada projektů snažící se o usnadnění vývoje WiFi aplikací vestavěním interpretu některého vyššího programovacího jazyka. Nejvýznamější z nich je interpret jazyku Lua, označovaný jako NodeMCU [25].

3.2.1 Oficiální vývojové nástroje

Firma Espressif na svých oficiálních stránkách poskytuje dva vývojové balíky. Jsou to: NON OS SDK [10] (proprietální řešení vystavěné na callback funkcích) a RTOS SDK [12] (využívá ke svému běhu operační systém FreeRTOS). Z hlediska uživatelského rozhraní jsou obě SDK totožné. Uživatelské rozhraní [9] [11] je nejlépe zdokumentovanou částí esp8266.

Problémem balíku je jeho uzavřenost. Zdrojové kódy klíčových funkcí jsou nedostupné. Místo toho balíky obsahují zkompilované knihovny (*.a) a k nim náležející hlavičkové soubory (*.h). Součástí knihoven jsou veškeré funkce potřebné pro běh wifi, tcp/ip stack a je zde také knihovna se standardními funkcemi jazyka C. Ve zdrojové podobě jsou zde pouze funkce obsluhující periferní moduly (SPI, I2S, UART ...).

Některé podstatné části jsou dostupné pouze jako binární soubory. Jedná se například o zavaděč ("bootloader"), který se musí nahrát na začátek adresového prostoru. Vlastní úpravy tedy nejsou možné.

Součástí balíku jsou ještě zkompilované interprety AT příkazů (pro různé velikosti flash paměti), sada linkovacích skriptů a ukázkový projekt.

Překladač není součástí balíku. Podle dokumentace SDK by měl být dostupný proprietální kompiler jazyka C xt-xcc jako součást obrazu pro virtuální stroj. Druhou varianou, kterou Espressif zmiňuje u svého RTOS SDK je využití komunitou vytvořeného překladače xtensa-xcc (gcc). Tato varianta je dokonce preferována.

Dalším nedostatkem SDK je absence nástroje použitelného k nahrání výsledného programu do paměti esp8266. Oficiální nástroje jsou tedy pro běžného vývojáře nepoužitelné. Kvůli jejich uzavřenosti jsou některé její části bohužel nezbytné pro jakékoliv jiné vývojové nástroje.

3.2.2 Komunitní SDK

Vzhledem ke stavu programové základny, ale i uzavřenosti zdrojových kódů oficiálních vývojových prostředí si komunita vytvořila své vlastnií SDK - esp-open-sdk [26]. Prostředí v sobě sdružuje různé otevřené projekty od esp8266 nadšenců do jednoho uceleného balíku. Balík je udržován ve veřejném repozitáři na serveru Github. Z programové základny obsahuje mimo již výše zmíněného gcc (verze 4.8) kompilátoru i program pro komunikaci s esp8266 (esptool.py). Ten umožňuje zápis a načtení programu do/z flash paměti, zjištění ID čipu, ID flash paměti(slouží ke zjištění velikosti této paměti), ale také převodník binárního formátu elf (výstup z gcc) do binární reprezentace zařízení.

SDK se kromě dodání vývojové programátorské základny snaží o maximální možnou eliminaci uzavřených knihoven, knihovnami volně dostupnými. Nahrazuje např. uzavřený TCP/IP stack za lwIP (A Lightweight TCP/IP stack). Úplná náhrada není vzhledem k nedostupnosti kvalitní dokumentace interních hardwarových částí možná.

Pro instalaci celého otevřeného SDK na linuxovém operačním systému stačí pouze stáhnout obsah z repozitáře a následně přeložit pomocí přiloženého makefile souboru. Pokud uživatel zvolí instalaci balíku zahrnující i stažení Espressif SDK (make standalone=y), nemusí se už o nic jiného starat.

3.2.3 Komunitní SDK s FreeRTOS

Podobně jako je tomu u oficiálního SDK, existuje i v otevřeném světě modifikovaná verze vývojového prostředí běžící na operačním systému FreeRTOS. Verze nese název esp-open-rtos [16] a je také dostupná v repozitáři na serveru github, ovšem pod jiným správcem. Tato verze je závislá na esp-open-sdk.

Kromě přidání operačního systému modifikuje i některé jiné částiti, jako např. implementaci kryptografického algoritmu TLS za aktuálnější implementace mbedTLS. Mimo to nahrazuje také uzavřenou standardní knihovnu jazyka C z oficiálního SDK otevřenou verzí.

Velkou výhodou esp-open-rtos je velká škála rozšiřujících modůlů. Pro tuto práci jsou významné zejména tyto:

- DHCP server
- HTTP server
- souborový systém SPIFFS pracující nad flash pamětí
- OTA upgrade - podpora aktualizací firmware přes WiFi
- knihovna pro práci s grafickými displeji
- I2C knihovna - softwarově emulovaná I2C

3.2.4 NodeMCU

NodeMCU[25] je modifikovaný interpret jazyka Lua, který abstrahuje hardware zařízení a pomocí svých knihoven jej zpřístupňuje vývojáři. Jazyk C je zde prakticky nahrazen jazykem Lua, což zrychluje fázi vývoje software. Součástí NodeMCU jsou i pokročilejší je např. i volitelný souborový systém. V podstatě by se dal NodeMCU označit za jakýsi druh operačního systému.

Ve světě mikrokontrolerů se nejedna o úplně ojedinělý projekt zaměřený pouze na esp8266. NodeMCU totiž vychází z projektu eLua, který zpřístupňuje tutéž funkcionalitu i na jiné MCU a je tedy poměrně dobře zdokumentovaný.

Abstrakce vrstvy HW přináší usnadnění vývoje aplikací, současně však spotřebovává nemalou část výpočetních prostředků, které mohou chybět pro samotnou aplikaci. Proto nebude v této práci NodeMCU použit. Řešení se zaměří na vestavění interpretu Lua pouze ke konfigurovatelnému sběru dat. Díky tomu bude možné v případě potřeby uvolnit alokovanou paměť dočasným ukončením interpretu. Jazyk Lua i kroky nutné k jeho integraci do esp jsou shrnuty v kapitole 3.5.

3.3 Snímače

Sekce popisuje princip funkčnosti čidel pro měření tlaku a střídavého proudu.

3.3.1 Měření tlaku

Pro měření tlaku se využívají 3 základní typy snímačů [29]

- **Tenzometrické tlakoměry** – Využívají piezorezistivního jevu (deformací kovu-polokovu dochází ke změně rezistivity tělesa). Tenzometr je uvnitř tlakoměru umístněn na tenkou membránu, která odděluje měřený a referenční tlak. Působením tlaku dochází k deformaci tenzometru. Změna odporu na tenzometru udává hodnotu měřeného tlaku.
- **Kapacitní tlakoměry** – Měřený a referenční tlak je oddělen dvěmi elektrodami fungujícími jako kondenzátor. Jedna je pevně přichycena a na druhou působí měřený tlak. Deformací této elektrody dochází ke změně kapacity.
- **Piezoelektrické tlakoměry** – Využívá piezoelektrického jevu (schopnost některých materiálů, jako např. křemík, při deformaci generovat elektrické napětí). Tento typ tlakoměru lze použít pouze u měnících se tlaků.

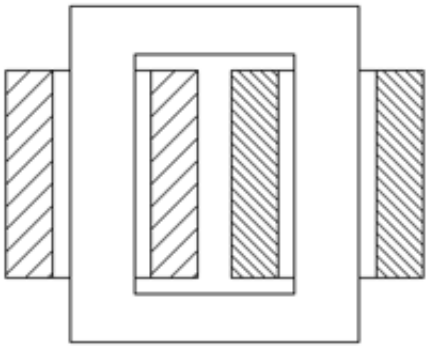
Dále se tlakoměry dělí podle typu referenčního tlaku:

- **Absolutní** – vstupní tlak je měřen proti vakuu
- **Relativni** – rozdíl tlaku vůči atmosferickému tlaku (okolí tlakoměru)
- **Diferenční** – rozdíl mezi dvěma vstupními tlaky

3.3.2 Měření střídavého proudu

Pro snímání střídávého proudu bez přerušení vodiče se nabízí použití snímače založeného na principu proudového transformátoru. Tedy převod elektrické energie mezi obvody pomocí elektromagnetické indukce.

Obecný transformátor je vyobrazen na obrázku 3.4. V principu jde o dvě cívky navinuté kolem materiálu (jádra) s dobrou magnetickou permeabilitou (jakým je např. železo). Jedna cívka (primární vinutí) je připojena na zdroj střídavého proudu. Na cívce vzniká magnetické pole. Pole je přeneseno jádrem na druhou cívku (sekundární vinutí), která převádí magnetické pole zpátky na proud.



Obrázek 3.4: Zjednodušený princip transformátoru [18]

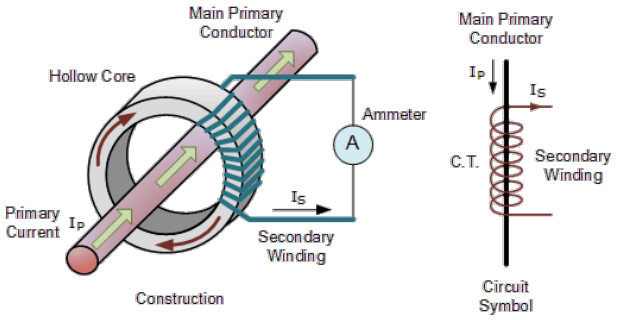
Převod transformátoru je dán počtem závitů na primární a sekundární cívce, resp. jejich poměrem. Poměr zachycyhe rovnice 3.1. N 1 a 2 představují počet závitů na primární a sekundární cívce [18].

$$p = \frac{N_1}{N_2} = \frac{U_1}{U_2} = \frac{I_1}{I_2} \tag{3.1}$$

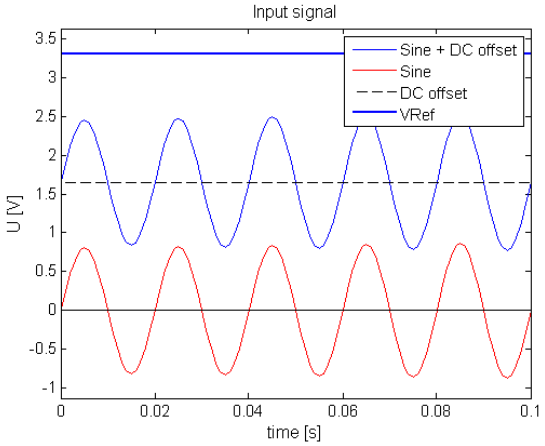
Úprava transformátoru pro měření proudu je vyobrazena na obrázku 3.5.

- Měřený vodič představuje primární vinutí transfromátoru. Při dosazení do vzorce 3.1 se $N_1 = 1$.
- "Hollow core"v ilustraci 3.5 představuje jádro. Tato část musí být ve vybraném snímači otevíratelná, aby bylo možné jej použít bez rozpojení měřeného obvodu.
- Na jádře ja navinuto sekundární vinutí ze kterého je vyčítán měřený proud.

Pro snadnější zpracování musí být proud převeden na napětí pomocí přesného rezistoru. Hodnotu je ještě potřeba převést do kladných hodnot (docházelo by ke zkreslení signálu v oblasti v oblasti spodní půlvlny). Toho se docílí přidáním stejnosměrné složky. Očekávané výsledné napětí ilustruje obrázek 3.6. Červeně je vykreslena detekovaná hodnota (Sine). Modře je znázorněna hodnota posunutá o stejnosměrnou složkou (Sine+Offset).



Obrázek 3.5: Transformátor v podobě proudového snímače [20]



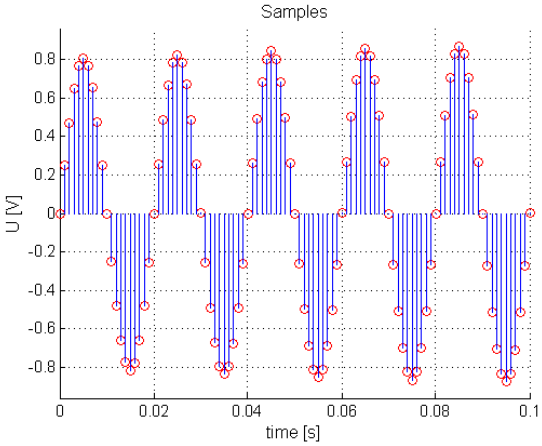
Obrázek 3.6: Výsledné střídavé napětí a posunuté střídavé napětí

Kvůli vlastnostem střídavého signálu nestačí odebrat pouze jeden vzorek. Výsledek by kmital od minimální po maximální hodnotu amplitudy, podle momentu sejmutí vzorku. Při měření střídavého proudu se očekává hodnota jeho efektivní hodnota I_{eff}. Pro ideální sinusový signál lze efektivní hodnota vypočítat podle vzorce 3.2. V praktické aplikaci však nelze s ideální sinusoidou počítat. Obecný výpočet efektivního proudu zachycuje vztah 3.3. K výpočtu integrálu musí být měřený proud navzorkován a poté aproximován některou z numerických integračních metod (obdelníkovou, lichoběžníkovou nebo jinou metodou). Podle Nyquistova teorému musí být frekvence vzorkování minimálně 2x větší než frekvence původního signálu. Pro co nejmenší zkreslení by měla být co nejvyšší. Obrázek 3.7 ukazuje očekávaný výstup po vzorkování.

Ještě před samotným výpočtem musí být od vzorků v MCU odečtena uměle přidaná stejnosměrná složka (Sine v obrázku 3.6).

$$I_{eff} = \frac{I_{MAX}}{\sqrt{2}} \tag{3.2}$$

$$\mathrm{I_{eff}} = \sqrt{\frac{1}{T}\int_0^T i(t)^2 dt} \tag{3.3}$$



Obrázek 3.7: Navzorkování střídavého napětí.

3.4 Napájení

Volba akumulátoru hraje roli při výběru napájecího obvodu.

Srovnání nejdostupnějších typů baterií.

	NiCd	NiMH	Li-Ion
Relativní kapacita (vztažená k NiCd)	1	1.4	3
Nabíjecích cyklů	300+	500+	300+
Samovybíjení %/měsic	30	20	3
Paměťový efekt [1]	velky	maly	-
Operační teplota (°C)	-30 – 140	-20 – 50	-10 – 50

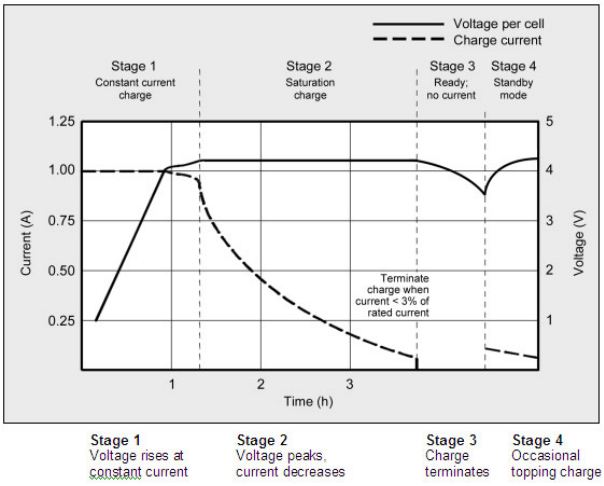
Tabulka 3.1: Vlastnosti nejdostupnějších typů akumulátorů [36]

Jako nejlepší se jeví využití Li-Ion článku. Udávané nominální napětí na těchto článcích je 3.6 V. Hodnota je však závislá na aktuálním stavu nabití článku. Obrázek 3.8 ukazuje napěťový rozsah při různých úrovních nabití.

Napětí pri plném stavu 3,6 V článku dosahuje 4,2 V. Naopak při vybití může klesnou i pod hranici 3 V. Tento rozsah napětí je poněkud nevýhodný v případě, že zařízení je napájeno stabilizovaným zdrojem 3.3 V. Stabilizace musí fungovat ve dvou režimech zvyšujícího nebo snižujícího měniče. Případně lze tento problém obejít řešením použitým v tomto zařízení (kapitola 4.5). Výrazný pokles napětí však také výrazně ovlivňuje životnost baterie. V případě poklesu pod kritickou hodnotu (závisí na konkrétním článku) může dojít až k nevratnému zničení článku.

3.5 Jazyk Lua

Lua je odlehčený vysokoúrovňový skriptovací jazyk, který byl navržen za účelem snadné integrace do jiných aplikací. Interpret jazyka je kompletně napsán v jazyce C a jeho funkce jsou poskytovány ve formě knihoven jazyka. Veškeré zdrojové kódy jsou zpřístupněny pod svobodnou licencí MIT, takže je možné interpret modifikovat a následně použít i v komerční aplikaci.



Obrázek 3.8: Nabíjeci cyklus Li-Ion článku [3]

3.5.1 Významné vlastnosti jazyka

Lua je primárně procedurální skriptovací jazyk, podporuje však i objektové a funkcionální programovací paradigma. Jeho syntaxe je podobná jazyku pascal.

Zde je malá ukázka syntaxe jazyka demonstrovaná na algoritmu "bubble sort"převzatá ze serveru rossetacode [27].

```
-- bubble sort v jazyce lua
function bubbleSort(A)
  local itemCount=#A
  local hasChanged
  repeat
    hasChanged = false
    itemCount=itemCount - 1
    for i = 1, itemCount do
      if A[i] > A[i + 1] then
        A[i], A[i + 1] = A[i + 1], A[i]
        hasChanged = true
      end
    end
  until hasChanged == false
end

list = { 5, 6, 1, 2, 9, 14, 2, 15, 6, 7, 8, 97 }
bubbleSort(list)
for i, j in pairs(list) do
    print(j)
end
```

Lua je dynamicky typovaný jazyk (pouze hodnoty mají definovaný typ). Celkem je k dispozici 8 datových typů. Zvláštností jazyka je, že obsahuje pouze jeden číselný typ. Za běhu lze proto používat pouze čísla s plovoucí řádovou čárkou nebo jenom celá čísla. Typ se určí při kompilaci.

O korektní uvolňování paměti se stará garbage collector. Jeho vlastnosti lze upravovat přes zabudované rozhraní.

Vstupní skripty nejsou interpretem přímo vykonávány, nejprve jsou totiž překládány do vlastního bytekódu, který je následně vykonán. Teoreticky lze v případě potřeby interpret zmenšit odstraněním překladače a ponechat jenom část interpretující bytekód. Bytekód však není kompletně nezávislý na platformě a toto řešení není zcela bezproblémové.

3.5.2 Paměťová náročnost

Projekt NodeMCU dokazuje, že se intepret jazyku LUA vejde i do omezeného paměťového prostoru jakým disponuje esp8266. Ostatně už samotný interpret přeložený na 64-bitovem linuxovém stroji, obsahující všechny standardní prostředky jazyka, zabírá přibližně 250 KB paměti a i ty nejméně výkonné moduly disponují flash pamětí s kapacitou alespoň 512 KB. Při spuštění bez vstupního skriptu (na prázdno) tento interpret naalokuje 27 KB operační paměti. Operační paměť esp8266 by tedy měla dostačovat i pro základní neupravenou verzi interpretu. Nezůstalo by však mnoho místa pro spuštění složitějších sktiptů, nebo jiných operací mimo interpret.

V dokumentaci modifikovaného interepretu Elua [19] je popsána technika jak paměťovou náročnost ještě více omezit. Redukce vychází ze způsobu jakým lua pracuje s uživatelskými moduly. Ve standardní implementaci lze moduly přidávat za běhu aplikace. Kvůli tomu jsou datové struktury uloženy v operační paměti. Nabízené řešení, označené jako LTR (lua tiny ram), přesouvá tyto tabulky do statického kódu. Po úpravě zabírá samotný intepret (bez vykonávaného skriptu) pouhých 5 KB datové paměti. Nevýhodou je nemožnost dynamického načítání uživatelských modulu za běhu programu, které však v této práci nebudou potřeba. Modifikace by měla být dostatečnou úsporou paměti pro bezproblémový běh aplikace.

3.5.3 Portování na esp8266

Díky tomu, že je Lua kompletné napsaná v jazyce C a využívá standardní knihovny, které jsou součástí dostupných vývojových prostředků, by neměl být problém ji přenést na esp8266. Měla by stačit úprava překládcích skriptů (změna kompilátoru a cesta ke standardním knihovnám).

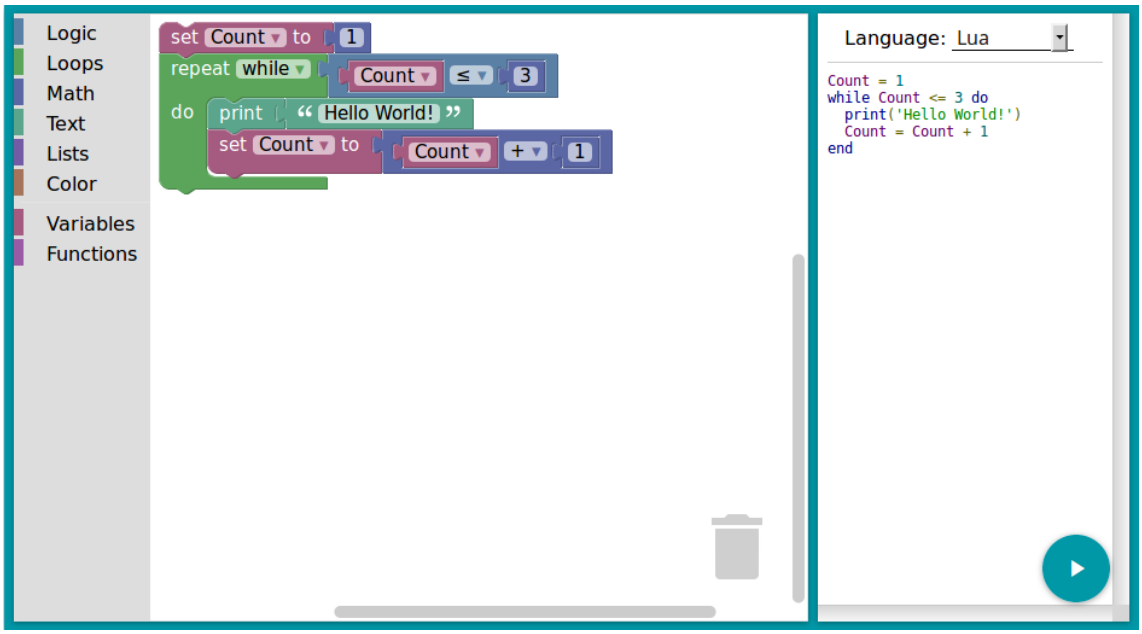
3.5.4 Vlastní funkce

Zbývá vyřešit problém využití jazyka Lua ke sběru senzorických dat. K tomu účelu bude třeba jazyk obohatit o vlastní funkce pracující s hardware. Vzhledem k primárnímu zaměření jazyka Lua coby vestavěného nástroje je volání funkcí jazyka C jeho přirozenou součástí. Vlastní funkce musí pouze dodržovat definované rozhraní pro předávání parametrů a návratových hodnot. Parametry jsou předávány prostřednictvím datového typu luastate. Pro návratové hodnoty používá Lua stejný zásobník jako C. Návratové hodnoty musejí být uloženy na zásobník (k tomu poskytuje Lua funkce lua_push()) a na poslední místo musí být vložen počet návratových hodnot. Vytvořené funkce poté stačí pouze předat interpretu (kvůli paměťové úpravě LTR) v době překladu.

Konstrukce interpretu dovoluje kromě volání uživatelských funkcí v jazyce C i přístup opačný, tedy volání funkcí napsaných v Lua z C.

3.6 Vizuální programování - Blockly

Google Blockly je knihovna napsaná v jazyce javascript sloužící pro tvorbu vizuálních programovacích editorů [15]. Možný výsledek editoru vytvořeného s touto knihovnou zachycuje obrázek 3.9. V levém panelu je zobrazeno samotné vizuální prostředí a v pravém panelu je zachycen výstupní kód. Jedná se o demonstrační ukázku z oficiálních stránek blockly.



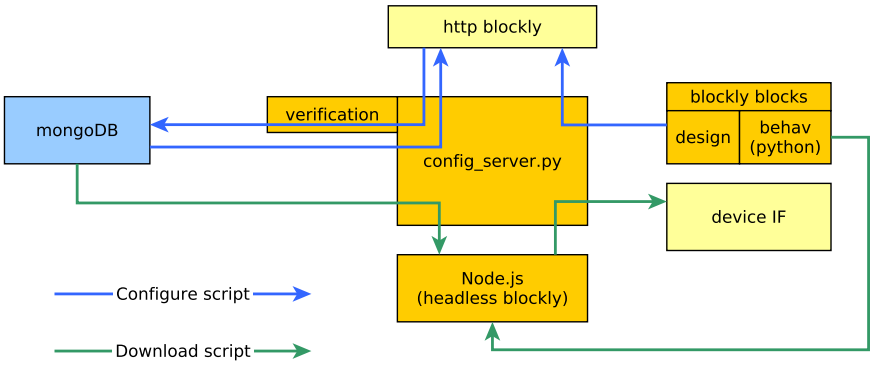
Obrázek 3.9: Demonstrační ukázka vizualizačního rozhraní Blockly [15]

Jednotlivým grafickým blokům je přiřazen výstupní kód. Podstatné je, že i když Blockly rozlišuje různé programovací jazyky, není těžké doplnit, případně upravit, libovolný blok obstarávající funkce přímo nepodporované vybraným jazykem. Vlastní kód je zadáván do struktur rozlišených podle generovaného jazyka a podpora více jazyků k jednomu vizuálnímu bloku je jednou z vlastností blockly.

Výsledný kód je možné buďto generovat přimo z webového prohlížeče, nebo lze použité prostředí exportovat, podle potřeby uschovat např. do databáze a kód generovat pomocí Node.js mimo prohlížeč. K exportní operaci samozřejmě existuje i operace pro import, takže je možné vizuální konfiguraci dále modifikovat.

3.7 Serverové prostředí

Ve firmě je již využívána výše zmíněná práce Pavola Vargovčíka [37], která zahrnuje serverovou aplikaci sloužící ke stejným účelům k jakým má sloužit i v této práci. Serverová aplikace sestává z části obsahující vizuální programovací prostředí Blockly (generující kód Pythonu), databáze uchovávající uživatelské skripty a z rozhraní sloužící k distribuci výsledných skriptů do koncových jednotek. Ukládání dat do externí databáze obstarávají samotné koncové jednotky.



Obrázek 3.10: Blokové schéma aktualního serveru

3.7.1 Použité SW technologie

Stávající serverová strana je vystavěna především na technologiích pythonu 3 a CoffeeScriptu (staticky kompilovaný do JavaScriptu). V Pythonu je vytvořen HTTP server a komunikační rozhraní s koncovými jednotkami. HTTP server je vytvořen s pomocí knihovny pro asynchronní HTTP client/server - aiohttp [2]. Skripty jsou ukládány do noSQL databaze MongoDB [24].

CoffeeScript (JavaScript) na straně klientského prohlížeče rozšiřuje Blockly o vlastní bloky, provádí základní předzpracování dat a předání vytvořených pracovních ploch Blockly na stranu serveru k jejich uložení. Komunikace se serverem probíhá přes protokol WebSocket. Překlad z vizuální reprezentace do výsledného kódu je prováděno až na vyžádání ze strany koncových jendotek. V takovém případě je vybrána patřičná konfigurace Blockly z databáze a s pomocí Node.js přeložen do požadovaného kódu Pythonu. Rozhraní je definováno metodami POST a GET protokolu HTTP (data lze snadno zobrazit ve webovém prohlížeči).

3.7.2 Konfigurační prostředí Blockly

Webové konfigurační prostředí je členěno do několika úrovní. Po rozkliknutí se zobrazí další úroveň. Na nejvyšší úrovni jsou výrobní podniky, níže budovy, haly a sekce. V sekcích se nachází bloky odpovídající výrobním linkám a teprve do nich jsou zasazeny konfigurační prostředí, tzv. "workflow".

Vlastní programovací bloky jsou také rozčleněny do logických kategorií. Patří tam kategorie pro

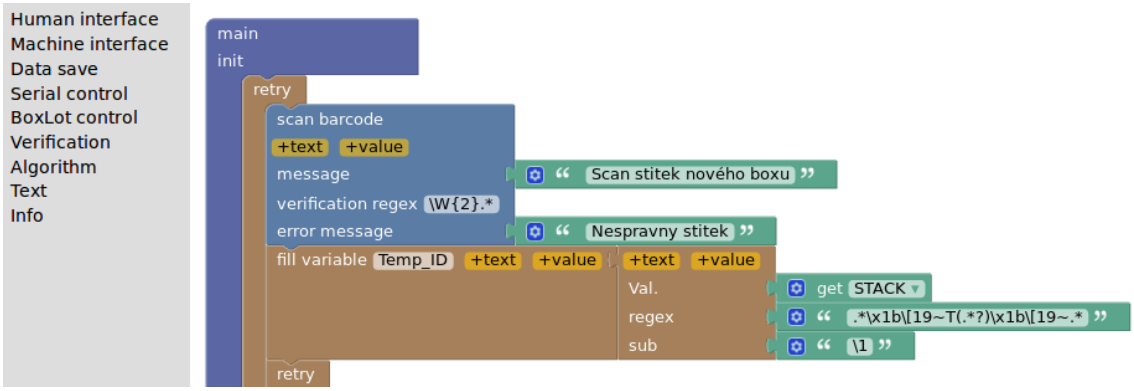
- Řízení toku programu (podmínky, cykly)
- Lidské rozhraní (manuálni vstup od operátora - text, číslo)
- Strojové rozhraní (definice chování pinů)
- Ukládání získaných dat
- Zaznamenávání výrobních surovin
- Informční textový výstup

Vstupní data jsou na koncových jednotkách ukládána buď na datový zásobník nebo do programátorem definované proměnné.

V rámci jednotlivých "workflow"si mohou programátoři výrobních procesů definovat vlastní procedury obdobně jako je tomu u běžných programovacích jazyků. V systému existují také procedury sdílené napříč všemi "workflow".

Kompletní výčet lze najít ve výše zmíněné práci [37].

Obrázek 3.11 zachycuje část konfiguračního skriptu použitého ve výrobním prostředí. Jak je vidět, bloky mohou být poměrně komplikované. Řada z nich povoluje programátorovi zadat regulární výraz, který slouží buďto k validaci uživatelského vstupu, nebo i následnou práci s různými řetězci.



Obrázek 3.11: Ukázka konfiguračního skriptu

Kapitola 4

Návrh technického řešení platformy

Tato kapitola se zabývá návrhem vývojové desky senzorické jednotky. Hlavní řídící jednotka byla specifikována v požadavích na platformu 2.1. Ostatní prvky jsou voleny s ohledem na specifikace tohoto modulu (napájecí napětí 3.3 V, GPIO piny).

4.1 Propojení periferních zařízení

Požadavky kladené na vstupně výstupní periferie nejsou právě malé. Celkem by jednotka měla být schopna obsloužit 3 vstupy, 3 výstupy, 3 senzorické snímače (tlak, teplota, proud), displej a kapacitní tlačítko a snímání stavu baterie. Modul ESP-WROOM-02 však nabízí pouze 11 GPIO pinů, přičemž externích periferií je celkem 12, nehledě na fakt, že na některých je potřeba připojit více než 1 pin. Kromě nedostatku GPIO pinů je zde také nedostatek analogově digitálních převodníků. Modul disponuje pouze jedním ADC. Řešení se nabizí v podobě externího rozšiřovače portů (port expanderu) a externího ADC. I ty je však nutné nějakým způsobem připojit na ESP modul. Jako nejvhodnější se jeví využití některé ze standardních sběrnic, které WROOM-02 podporuje. Většina druhů součástek je dostupná s rozhraními SPI a I2C. Obě umožňují připojení několika zařizení. SPI však vyžaduje větší množství vodičů. Kromě datové (plně duplexní - 2 datové vodiče) a hodinové cesty vyžaduje SPI jeden vodič do každého připojené jednotky, jenž určuje kdo má být příjemcem dat. Naproti tomu I2C stačí pro připojení většího množství zařízení pouze 2 vodiče. Koncová zařízení jsou vybrána prvním přeneseným bytem obsahujícím adresu (7 bitů) a směr přenášených dat (1 bit). Proto je zvoleno rozhraní I2C.

4.2 Volba čidel

Sekce popisuje vlastnosti vybraných čidel.

4.2.1 Měření tlaku

Pro měření tlaku se nepodařilo nalézt takový senzor, který by dokázal fungovat na napájecím napětí 3.3V. Na trhu jsou dostupné pouze 5V varianty. Kvůli tomuto snímači bude muset být napájení jednotky rozděleno do dvou větví - 3.3 V a 5 V.

Pro měření tlaku byl zvolen senzor MPX5050GP. Naměřená hodnota je vztažena vůči atmosferickému tlaku. Snímání probíhá na principu piezorezistivního jevu. Maximální udávaný měřený tlak je 50kP se základní přesností měření +-2.5%. Na výstupu dává výsledky v analogové formě, bude tedy potřeba využít AD převodník. Výstupní napětí je dáno vztahem 4.1. Vs představuje napájecí napětí a P měřený tlak.

$$V_{OUT} = V_S * (0.018 * P + 0.04) \tag{4.1}$$

4.2.2 Měření teploty

Ke snímání teploty bylo zvoleno čidlo LM75AD. Jeho předností je výstupní digitální formát. Odpadá tedy nutnost analogového zpracování dat, což značně usnadňuje aplikaci. Údaj o teplotě je v digiitální podobě vyčítán prostřednictvím sběrnice I2C. Operační teplota je od -55 do 125°C. Udávaná přesnost od -25 do 100°C je +-2°C. V čipu lze nastavit i teplotní limit po jehož překročení dojde k sepnutí výstupního pinu obvodu. Ten však nebude využit. Sensor je napájen napětím 3.3V. Udávaná maximální spotřeba je při aktivované sběrnici I2C 1mA.

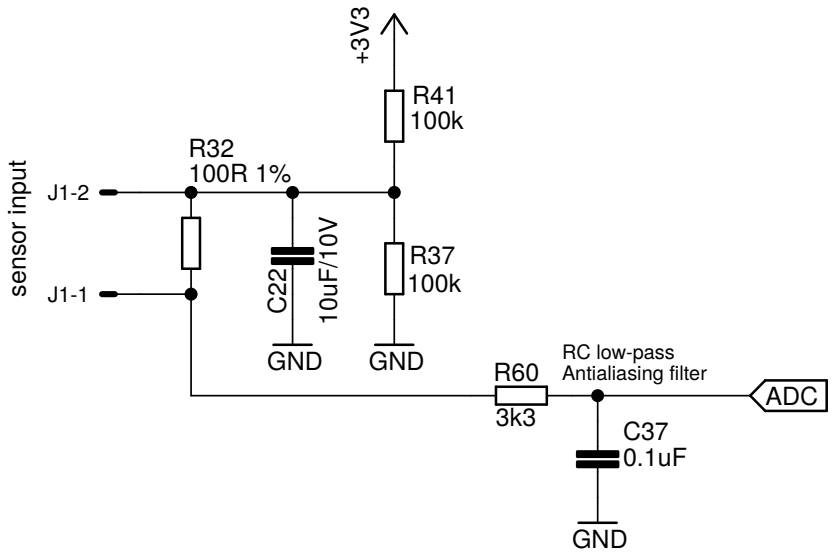
4.2.3 Měření Proudu

Byl vybrán snímač ECS1030-L72 [31] fungující na principu proudového transformátoru, který je popsán v kapitole 3.3.2. Vybraný snímač dovoluje instalaci bez nutnosti přerušení obvodu, což bylo jedním z požadavků zadavatele. Provedení snímače je vidět na obrázku 4.1.

Obrázek 4.1: Proudový snímač [31]

Pro frekvenci 50/60 Hz je udávaná přesnost 2%. Na výstupu dává zmenšený proud, který lineárně odpovídá měřenému proudu. Proud 30A tekoucí primárním vinutím (měřeným vodičem) odpovídá proudu 1,5mA v sekundárním vinutí. Aby bylo možné tento proud měřit, je nutné do obvodu zapojit resistor R32, který zároveň poslouží jako jednoduchý převodník proud - napětí. Měřicí odpor také slouží pro nastavení měřeného rozsahu díky přímé úměře mezi proudem a zvolenou hodnotou odporu. Limitem při měření napětí je pak rozsah AD převodníku. Obrázek ukazuje zvolený způsob zapojení. Stejnosměrná složka je polovina napájecího 3.3 V napětí. Čehož bylo dosaženo napěťovým děličem R34 a R41.

Zátěžový odpor byl zvolen na hodnotu 100 Ohm. Při maximálním měřeném proudu 30 A bude na výstupu napětí 3,15 V. Pro odečtení hodnotu bude dostačovat 3,3 V AD převodník.

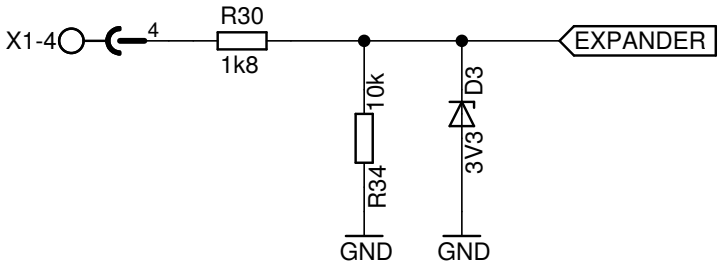


Obrázek 4.2: Vstupní obvod proudového snímače

4.3 Vstupní obvod

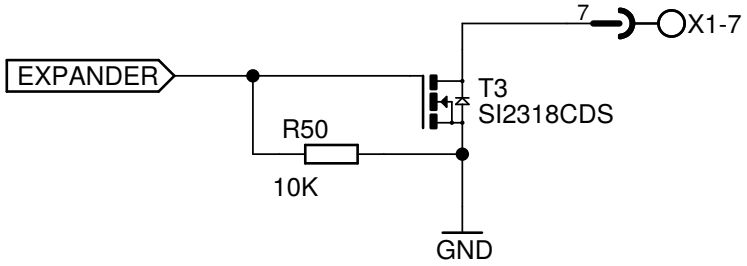
Řešení nedostatku obecně použitelných vstupních pinů řeší výše zmíněný port expander. Na trhu jsou běžně dostupné expandery s různým počtem pinů – od těch nejmenších 4 pinových až po 60 pinové. V tomto projektu je potřeba připojit 6 vstupně výstupních vodičů. Pro usnadnění práce s nimi, je vhodné připojit je tak, aby se se všemi komunikovalo jednotným stylem – tzn. všechny do expanderu (i kdyby byly volné vstupy na ESP modulu). 6 pinový expander není dostupný, proto je potřeba zvolit nejbližší vyšší počet, což je 8. Počtem pinů i propojovací sběrnicí odpovídá obvod PCA9557PWR [33] od firmy Texas instruments. Operační napětí obvodu je udáváno v rozsahu 2,3 V - 5 V. Porty jsou 5 V tolerantní. To je ovšem pro potřeby tohoto projektu nedostačující. Jednotka bý měla být schopna komunikovat se zařízeními pracujícími na napětí od 5 V do 24 V. Proto je potřeba piny ošetřit.

Nabízí se několik možností jak realizovat ochranu vstupů. Ideální ochranou je v průmyslovém použití galvanické oddělení vstupně výstupní části. To lze realizovat například použitím optočlenů nebo RF izolátorů. Galvanické oddělení je však poměrně složitou záležitostí zahrnující spoustu dalšch problémů včetně například nutnosti dvojího odděleného napájení soustav. Vzhledem k tomu, že chráněný obvod má být sám o sobě levný, takže jakákoliv sofistikovanější a tedy dražší metoda postrádá smysl. Proto byla zvolena jednodušší, avšak také méně robustní metoda založená na zenerových diodách. Zvolené zapojení je vidět na obrázku 4.3. Odpor R30 zde omezuje maximální proud tekoucí zenerovou diodou a R34 slouží jako pull-down rezistor (udržuje logickou 0 na portu expanderu v případě kdy není připojeno žádné externí zařízení). Oba odpory zároveň působí jako dělič napetí, čímž proudově odlehčuje namáhání zenerovy diody.



Obrázek 4.3: Ochrana vstupního obvodu zenerovou diodou.

Výstupní obvod je realizován tranzistorem typu MOSFET s N kanálem (metal-oxide field effect transistor) - SI2318CDS-T1-GE3. Maximální spínané napětí mezí Drain-Source je více než dostatečných 40 V a maximální proud 5 A. Zapojení je zachyceno na obrázku 4.4.

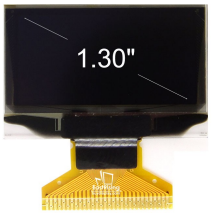


Obrázek 4.4: Spínání externího zařízení tranzistorem N MOSFET.

Externě budou vstupy i výstupy zpřístupněny jedním 8-pinovým konektorem (2 piny pro propojení země).

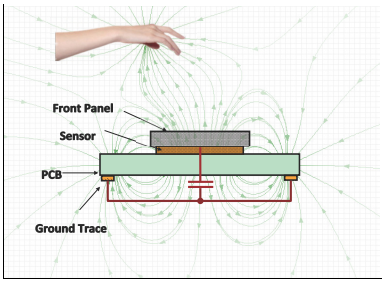
4.4 Uživatelské rozhraní

K zobrazení informací přímo na zařízení byl zvolen černobílý 1,3” OLED displej ER-OLED013-1W [4] zachycený na obrázku 4.5. Rozlišené displeje je 128x64. Řízen je pomocí ovladače SSD1306 [30]. Kvůli požadavku na umístnění kapacitního tlačítka pod displej, není možné použít displej v podobě hotového modulu zabudovaného na DSP. Tyto moduly mají vyvedeny pouze komunikační piny, naproti tomu u samostatného displeje je třeba připojit i konfigurační piny. Tím se o něco zkompikuje zapájení displeje.



Obrázek 4.5: 1.3"OLED displej [4]

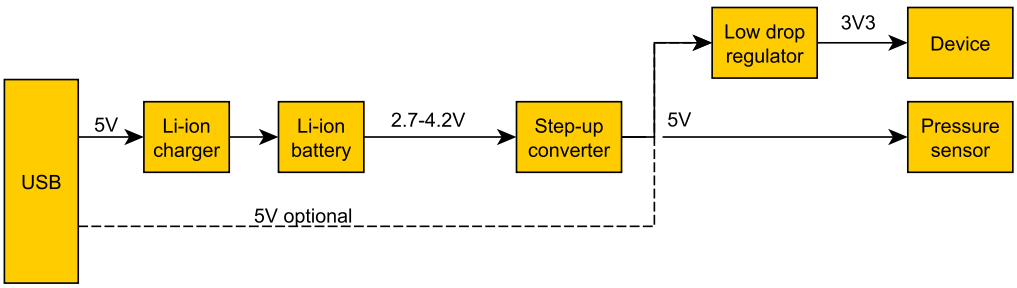
Vstupní tlačítko by mělo být ve formě zmíněného kapacitního snímače. I když je možno výstup tlačítka zaznamenat bez specializovaných komponent, pouze pomocí ADC spolu s vyhodnocením v MCU, byl k tomuto účelu vybrán dedikovaný čip MTCH101. Kvůli nedostatku ADC na ESP modulu by stejně bylo potřeba externí komponentu přidat. Informace o sepnutí tlačítka je z MTCH101 vyvedena v digitální podobě. Další výhodou obvodu je možnost nastavení citlivosti tlačítka. Obvod provádí detekci stisknutí přibližně každých 69–105 ms. Četnost snímání lze přepnout i do úsporného režimu, snímání v intervalech 572-805 ms, díky čemuž klesne spotřeba z 200uA na 50uA. Vzájemné umístnění kapacitní plochy a displeje ilustruje obrázek 4.6. "Front Panel"představuje displej.



Obrázek 4.6: Kapacitní snímač pod displejem. [4]

4.5 Napájení

Kvůli tlakovému snímači musí být napájení rozděleno do dvou větví. 5V pro tlakový senzor a 3.3V pro všechno ostatní. Napájení má být dodáváno z Li-Ion akumulátoru. Je žádoucí aby zařízení umožňovalo jeho dobíjení a kvůli vývoji i možnost běhu bez akumulátoru. Zvolené rozvržení napájení ilustruje blokové schéma 4.7. Externí napájení je vedeno z mini-USB konektoru. USB standard dodává 5V napětí. Z USB je napětí vyvedeno na dobíjecí obvod Li-Ion článků MCP73833-BT [21]. Velkou výhodou obvodu je jeho samostatnost. Obvod se sám postará o celou fázi nabíjení. Navíc dovoluje sledování průběhu nabíjení ze dvou stavových pinů (k pinům bude připojena 2 barevná LED dioda).



Obrázek 4.7: Blokové schéma napájecího obvodu.

Vzhledem k vlastnostem Li-Ion článku, nastíněným v kapitole 3.4 je napětí nejprve stabilizováno na 5 V pomoci DC-DC měniče MCP1640 [22] a následně sníženo LDO (low dropout) regulátorem LD1117 [32] na 3,3 V.

4.6 Programovací rozhraní

Zápis firmware do programu přes rozhraní UART bude prováděno USB-UART převodníkem cp2102 [28].

4.7 Ladění aplikace

Jelikož jde o první vývojovou desku, očekává se že některé části nebudou fungovat podle očekávání. Pro usnadnění diagnostiky jsou ke vstupům ESP-WROOM-02, ale i na jiná místa přidány nulové odpory umožňující odpojení a případné předrátování pinů. Jsou zde také vývody na využité komunikační sběrnice I2C a UART. Navíc je z ESP-WROOM-02 vyvedena TX linka druhé UART periferie modulu určená pouze k debugovacím výpisům.

4.8 Deska plošných spojů

Kompletní schéma zapojení je součástí přílohy B. V příloze C a D se nachází navržená DPS a seznam použitých součástek.

Zařízení je rozvržené na obdelníkovou dvou vrstvou DPS o rozměrech 60x90 mm.

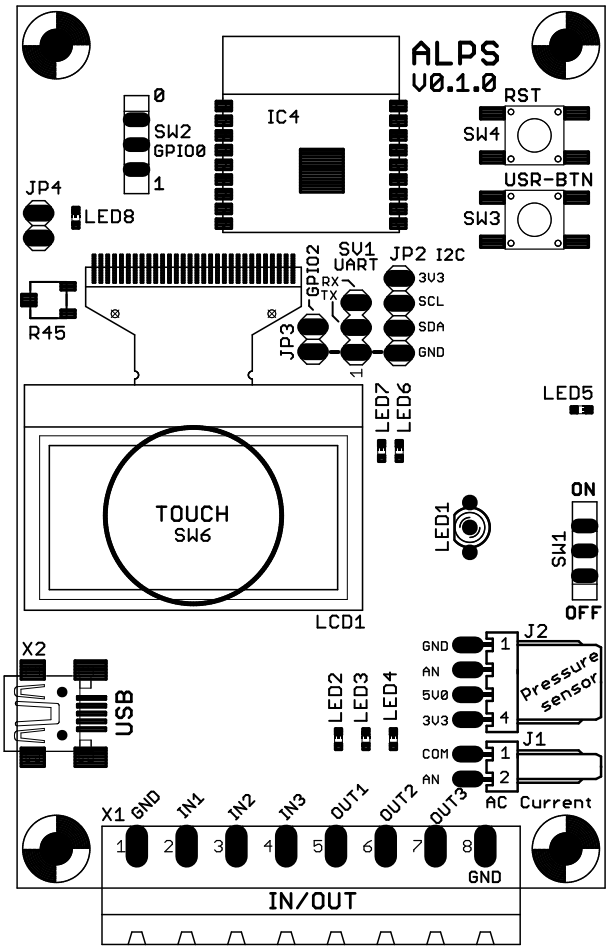
Šířky cest a rozměru otvorů jsou voleny s ohledem na vyrobitelnost specializovanou firmou zabývající se výrobou DPS. Pro usnadnění osazování bylo zvoleno rozmístnění komponent pouze na horní vrstvu. To umožní osazení DPS metodou, kdy je na pájecí plochy nanesena pájecí pasta, umístněny komponenty a k samotnému zatavení dochází ve speciální peci. Na spodní straně je poze konektor pro baterii.

Rozmístění komponent zachycuje obrázek 4.8. Většina komponent je volena v pouzdře typu SMD. Velikost pasivních komponent (rezistory a kondenzátory) je převážně standardní rozměr označován jako 0603. ESP-WROOM-02 je umístněn na horním okraji desky (IC4). Umístění je zvoleno na základě doporučení v oficiálním dokumentu "placement guide"[6]. Pod částí antény modulu je z důvodu co nejlepší kvality signálu odstraněna kovová vrstva z obou stran DPS.

Konektory jsou rozmístněny ve spodní části. Zleva je napájecí a programovací konektor mini-USB (X2). Na spodním okraji je vstupně výstupní sběrnice (X1). Vpravo od ní jsou konektory pro snímač proudu (J1) a snímač tlaku (J2). K tlakovému konektoru je vyvedeno napájecí napětí 5V i 3.3V. I když je provedení tlakového snímače 5 V, teoreticky je na tento konektor možné připojit libovolný jiný snímač. Nad těmito konektory je hlavní vypínač jednotky SW1. Vedle ESP modulu je debugovací (SW3) a resetovací tlačítko. Z druhé strany se nachází přepínač sloužící k výběru bootovacího režimu ESP na pinu GPIO0.

V centrální části je jasně rozeznatelný OLED displej (LCD1). Displej je kvůli omezenému prostoru DPS otočen vzhůru nohama, tak aby byly jeho vývody co nejblíže ESP modulu. V zobrazovací části displeje (pod displejem) se nachází plocha kapacitního tlačítka (SW6). Vlevo od displeje je umístněn potenciometr (R45) sloužící k nastavení citlivosti obvodu zpracovávajícího signál z kapacitní plochy a také pin header k přepnutí na úsporný běh téhož obvodu.

Na druhé straně od displeje jsou vyvedeny sběrnice I2C (JP2), programovací UART (SV1) a TX část UART pro ladící výpisy (JP3). Spolu s I2C je vyvedeno i napětí 3.3V, tak aby sem bylo možné případně zapojit další snímač. Na desce je také řada LED diod. Všechny diody, až na LED1 a LED5, slouží k debugovacím účelům. Tyto diody jsou připojeny k



Obrázek 4.8: Rozvržení komponent na DPS.

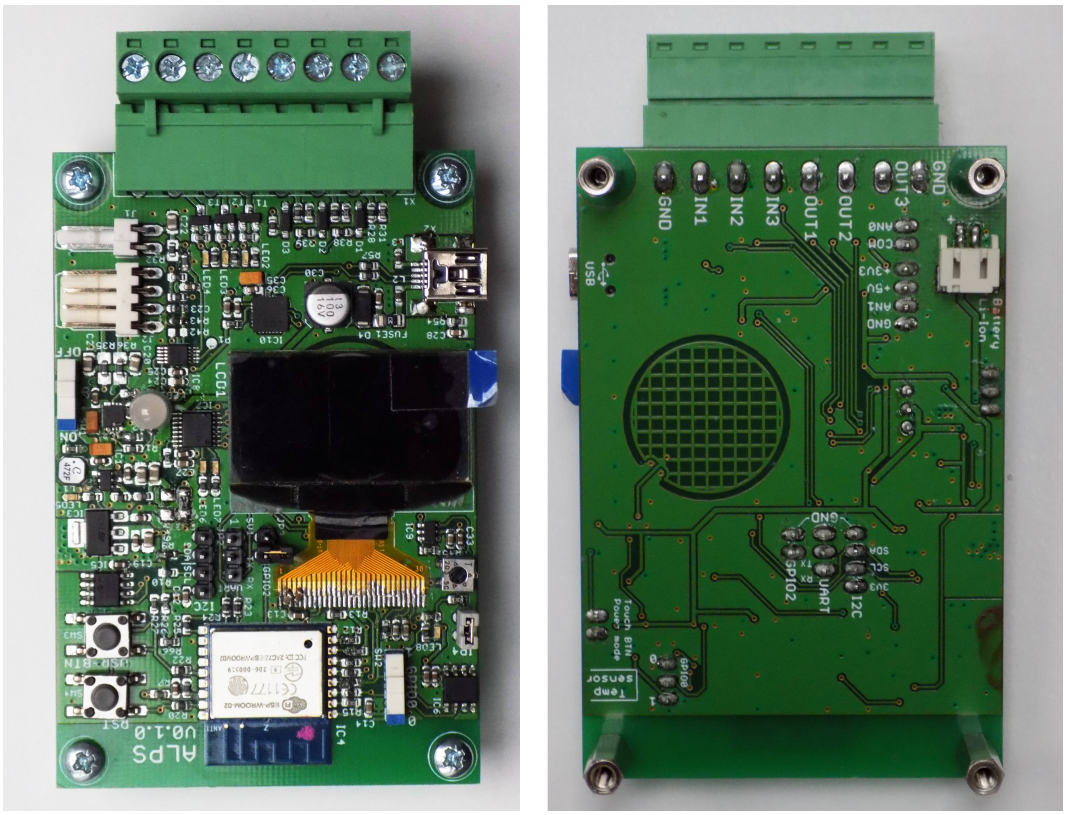
příslušným datovým vodičům. LED1 je připojena k nabíjecímu obvodu Li-Ion baterie a LED5 indikuje stav zapnutého zařízení. Udává fázi ve které se nabíjení právě nachází.

Plocha kapacitního tlačítka je realizována podle příručky [23] k použitému obvodu MTCH101. Vnitřní plocha má v poloměru 9mm, okolo je 0.8 mm izolace za kterou je 1 mm široký ochranný prstenec (GND). Prstenec je oddělený od okolní země kvůli zamezení nežádoucího účinků působících z okolních obvodů DPS. Dohromady fungují centrální plocha a okolní země jako otevřené desky kondenzátoru. Kapacita kondenzátoru je dána několika faktory: rozměrem desek, šířkou dielektrika a materiálem dielektrika. Příblížením prstu k vytvořené ploše dojde ke změně kapacity, která je detekována zmíněným obvodem. Ze spodní strany desky pod kapacitní plochou je rozvedena síť ("hatch") spojená s dotykovou plochou z důvodu zamezení aktivace tlačítka při přiblížení ze spodní strany.

Kapitola 5

Realizace a oživení senzorické platformy

Samotná dvouvrstvá deska plošného spoje byla vyrobena externí firmou. Osazení proběhlo již ve firmě ALPS Electric Czech za použití pájecí pasty a následného průchodu přetavovací pecí, kterou má firma k dispozici. Výsledné zařízení je zachyceno na obrázku 5.1.



Obrázek 5.1: Foto osazené DPS

5.1 Oživení HW

Připojení bateriového zdroje energie se obešlo bez viditelných známek chybné konfigurace.

Komunikace s modulem WROOM-02 prostředníctvím seriového rozhraní UART probíhala bez problémů a bylo možné přistoupit k ověření funkčnosti ostatních komponent. K tomu účelu byl využit výše zmíněný NodeMCU (3.2), který je díky přítomnosti I2C knihovny a možnosti zadávat Lua skripty přímo přes UART rozhraní k tomuto účelu ideální.

Veškeré komponenty na sběrnici I2C správně reagovaly na detekci vlastní adresy nastavením potvrzovacího ACK bitu. Při bližším prověření jednotlivých komponent se však ukázala řada nedostatků, které jsou rozepsány níže.

5.1.1 Teplotní čidlo

Teplotní čidlo pracuje podle specifikací, problém je však s umístěním komponenty na desce. Nachází se totiž asi pouze 2 cm od řídícího modulu WROOM-02, který se po krátkém čase své činnosti zahřívá, teplo je pak deskou plošného spoje vedeno až k samotnému snímači, což způsobuje zkreslení. Vzhledem k povaze aplikace zařízení, které slouží poze jako prvotní validační krok k plně funkčnímu výrobku, není tato chyba nijak závažná. Důvodem umístnění čidla přímo na desku plošného spoje byla potřeba připojení libovolného čidla komunikujícího na rozhraní I2C. Ve finální verzi desky bude potřeba čidlo přesunout na místo které nebude teplotně ovlivňováno okolními komponentami, nejspíše mimo desku na externí konektor.

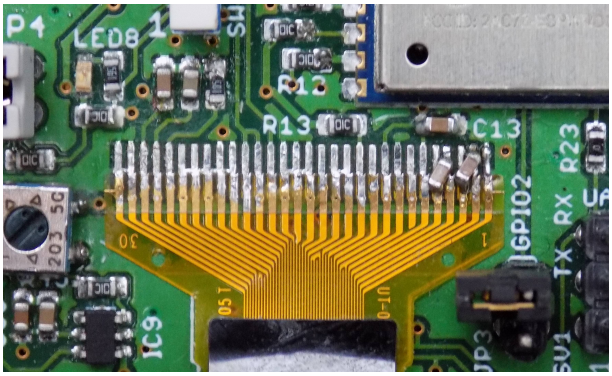
5.1.2 AD převodník

Zvolený AD převodník disponuje 2 vstupy, bohužel však pouze 1 kanálem (pracuje buď jako diferenční, nebo získává hodnotu z prvního vstupu vztažený vůči zemi). Lze tedy číst pouze z 1. vstupu. K použitému převodníku ADS1014 však existuje i rozšířená, 4 kanálová verze ADS1015 [34], zabudovaná do stejného pouzdra se stejnými výstupy (ADS1014 má 2 nevyužité piny). Na desce tak stačí pouhá záměna těchto komponent. Protože se jedná o významnou součást projektu, byla tato změna rovnou aplikována na desku.

5.1.3 Displej

Další problém se ukázal u displeje. Komunikace po I2C lince probíhala v pořádku, zprávy byly potvrzovány ACK bity, displej však nic nezobrazoval. Ukázalo se, že chyba je ve způsobu napájení. SSD1306 [30] má totiž uvnitř dva oddělené obvody, jeden pro řídicí logiku a druhý pro zobrazovací OLED část. Zatímco řídící část V_{DD} vyžaduje napájení od 1,65 do 3,3 V, zobrazovací část V_{CC} potřebuje ke správné funkci 7 - 15 V. V_{CC} je možné dodávat buď externě, nebo lze využít vnitřní DC-DC měnič ("charge pump") zvyšující vstupní V_{DD} napětí na 7.5 V. Zapojení bylo chybně zvoleno na verzi spoléhající na dodání externího zdroje V_{CC} ale přivedeno bylo pouze napětí 3,3 V. Protože na desce je k dispozici maximálně napětí pouze 5 V, nepřipadá dodatečné dodání externího napájení v úvahu. Pro zapnutí vnitřního měniče je potřeba přepojit V_{CC} přes kondenzátor na GND a dále přidat dvojici kondenzátorů pro DC-DC měnič.

Obě změny bylo možné celkem snadno aplikovat na stávající desku. Piny pro kondenzátory, vyžadované DC DC měničem, leží na konektoru displeje vedle sebe, takže bylo možné je připájet přímo na něj. V případě kondenzátoru pro V_{CC} stačilo přerušit spoj u blokovacího kondenzátoru od napájení. Zásahy do desky jsou viditelné na obrázku 5.2.



Obrázek 5.2: Oprava napájení displeje na desce

5.1.4 Kapacitní snímač

Součástí požadavků bylo ověření možnosti zabudovat dotykový snímač pod zobrazovací plochu displeje. Doteky jsou sice zaznamenávány, avšak ne příliš spolehlivě. Displej je k desce přilepen relativně širokou oboustrannou lepící páskou a je možné, že jsou doteky zaznamenávány spíše změnou vzdálenosti displeje od DPS než od prstu. Pro další vývoj bylo proto rozhodnuto použít pro uživatelské rozhraní běžné tlačítko.

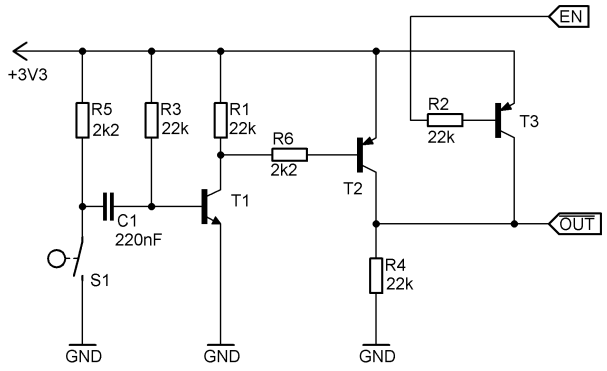
5.1.5 Běh z baterie

Při návrhu zařízení byl opomenut úsporný běh z baterií. Jednotlivé komponenty jsou napevno připojeny k napájení a při uspání řídicího modulu stále odebírají nemalé množství energie. V další verzi bude potřeba u některých komponent, zejména u displeje, mít možnost ovládat přívod energie k nim.

Při bližším prozkoumání vývojových prostředků se ukázalo, že WROOM-02 lze uspat pouze do režimu z nějž je možné zařízení probudit pouze uzeměním resetovacího pinu. Tímto způsobem je modul probuzen interními hodinami, kdy je propojen výstupní pin modulu na jeho resetovací pin. Ostatní periferie však nejsou schopny vzbudit modul ze spánku. Tím by byly výrazně omezeny možnosti využití modulu. V další verzi proto musí být signalizační piny periferií přivedeny na resetovací vývod řídicího modulu. K tomu bude třeba signalizační piny ošetřit tak, aby na svém výstupu signalizovali aktivní stav zařízení pomocí logické 0 a zároveň aby tento stav byl přítomen pouze na krátký okamžik. Jinak by řídicí modul zůstal zaseknutý v resetovacím stavu.

Jedno z možných řešení je propojení pinu přes obvod zachycený na obrázku 5.3. EN povoluje nastavení OUT do log. 0. Je li na EN nízká úroveň, pak je reset modulu WROOM-02 (OUT) připojen na napájecí napětí a nelze jej restartovat. Pokud by totiž mohl externí obvod kdykoliv manipulovat s hodnotou OUT, docházelo by k resetu i v momentě kdy jednotka není v režimu spánku. Spínač S1 reprezentuje obvod na jehož základě by měl být v MCU vyhodnocena externí událost. V rozepnutém stavu je tranzistor T1 otevřen, což otevírá tranzistor T3 (ten zde slouží jako negace hodnoty na T1) a na OUT je přivedena vysoká hodnota. Sepnutím S1 dojde k nabití kondenzátoru C1, tento děj dočasně uzavře T1 a tedy i T2 a na OUT se objeví nízká úroveň a dojde k restartu modulu.

Kromě změn v zapojení desky je nutné počítat i s jistými omezeními v programové části. Přechodem do úsporného režimu totiž dochází ke ztrátě všech dat v RAM paměti. Jediná možnost uchování dat během režimu spánku je použití malé (500 B) paměti RTC modulu.



Obrázek 5.3: Obvod pro probuzení modulu ze spánku přes reset

Ukládat data do flash paměti není z důvodu omezeného počtu zápisů možné. Přepínání do úsporného režimu totiž může nastávat i několikrát za vteřinu a flash paměť by byla rychle zničena.

5.2 Orientační cena

Cena všech součástek na DPS se pohybuje okolo 900 Kč. Do celkové ceny je také nutno zahrnout i samotnou senzorickou část. Její hodnotu však nezle přesně stanovit, jelikož právě tyto komponenty se mohou lišit podle aktuálních požadavků na měřenou veličinu. V případě této testovací konfigurace, kde je požadováno externí čidlo tlaku a snímač elektrického proudu vzůstá cena dále o hodnotu přibližně 500Kč (tlak: 400Kč, proudový snímač: 100Kč). V ceně navíc není zahrnuta výroba DPS, jejíž výroba je v malém počtu jednotek drahá, ale v případném koncovém řešení by byla rozložena do více kusů.

V rámci součástek na DPS se nejvíce podílejí na její vysoké ceně OLED display (150 Kč), sběrnice pro propojení digitálních vstupně výstupních pinů (150 Kč), řídicí modul WROOM-02 (75 Kč), AD převodník (75 Kč) a převodník z USB na UART (75 Kč). Zbytek je rovnoměrně rozložen mezi ostatní součásti.

Kapitola 6

Návrh software

Software zahrnuje kompletní programové vybavení koncového zařízení (firmware) a modifikaci serverové aplikace. Možnou koncovou konfiguraci systému zachycuje obrázek 2.1 v kapitole popisu požadavků. Kompletní programové vybavení se v mnohém prolíná a je proto popsáno v jedné kapitole.

Na straně serveru řešení vychází z již existující aplikace, ale značně ji rozšiřuje o možnost vytvářet konfigurační Lua skripty. To vyžaduje doplnění odpovídajícího Lua kódu ke každému Blockly bloku a editaci skriptů transofmujících bloky do výsledného kódu. Dále pak přidává komunikační rozhraní se zařízením a s operátorem výroby. Z důvodu omezených výpočetních prostředků esp8266 bude ukládání dat do externí databáze prováděno přes proxy server. Výhoda tohoto řešení je ve volnosti definice komunikačního rozhraní. Stačí tak odesílat jednoduchý textový řetězec, který bude zpracován do podoby vhodné pro zaznamenání do databáze až na straně serveru.

6.1 Identifikace jednotek

Každá jednotka musí být jednoznačně identifikovatelná. Protože by jednotky měly být snadno zaměnitelné (např. při poruše), odpadla možnost identifikace pomocí MAC adresy. Po konzultaci se zadavatelem bylo rozhodnuto, že jednotkám bude před jejich nasazením manuálně přiřazeno jednoznačné jméno. Stejné jméno bude přiřazeno i konfiguračním blokům v Blockly. Jméno má sloužit i v případě vzdáleného přístupu na zařízení v případě vzdálené konfigurace.

6.2 Konfigurace jednotek

Podle požadavků zadavatele bude na ESP8266 běžet HTTP server s konfiguračním rozhraním. Rozhraní musí minimálně umožňovat změnu SSID a hesla do operační sítě jednotek a v produkčním prostředí je navíc vhodné mít možnost manuálně nastavit IP adresu každého zařízení.

Přístup na zařízení bude buďto přes lokální přístupový bod nebo vzdáleně prostřednictvím sítě, kam je jednotka za běžného provozu připojena.

I když ESP8266 dokáže být připojeno do WiFi sítě jako klient a zároveň mít spuštěn lokální AP, není tato varianta vhodná. Při nasazení většího množství jednotek v omezeném prostoru by mohlo docházet k vzájemnému rušení v rámci 2.4 GHz WiFi pásma. Proto

bude zařízení přepnuto na lokální AP až po manuálním vyžádání pomocí stisknutí tlačitka po dobu 3s.

U konfiguraci využívající pracovní bezdrátovou síť jednotek je problém s jejich identifikací. V tomto případě bude potřeba rozšířit konfigurační server o jednoduchou funkcionalitu jakou běžně plní DNS server. Tedy přiřazení jmen jednotek k jejich IP adrese. K tomu postačí jednoduchá stránka se jmény jednotek a hypertextovými odkazy.

Údaje budou zadávány ve standardním html formuláři a po potvrzení se zadaná data přesměrují HTTP metodou GET na adresu `/config_device`. Díky jednotnému rozhraní bude v budoucnosti možné nastavovat jednotlivá zařízení hromadně.

6.3 Uživatelské rozhraní

Displej bude využit k zobrazení základních stavových informací jakou je stav akumulátoru a textový výstup definovaný v Lua skriptu. Zařízení má pouze jednen ovládací spínač, který bude snímán opět až na vyžádání ze skriptu. Spínač bude použit i k přepnutí jednotky do konfiguračního AP režimu (pokud bude zmáčknuto po startu jednotky).

Pokročilejší rozhraní nabídne jednotka na svém lokálním HTTP serveru. Na výběr budou stejné informace jako v případě displeje a navíc zde bude uživatelský interaktivní terminál, jehož operace budou definovány v Lua skriptu (např. zadání hodnoty po jejímž překročení má jednnotka provést definovanou akci).

6.4 Uživatelské skripty

Oproti verzi vytvořené v již zmíněné práci [37] běžící na Raspberry Pi bude výběr bloků značně omezený. Limitujícím faktorem je výpočetní výkon esp8266 i záměrně omezená implementace I/O rozhraní pro připojení externích periferií. Odpadají tak bloky určené pro řízení výroby. Veškeré zadavatelem požadované bloky shrnuje tabulka 6.1.

Po vytvoření nového blockly prostředí odpovídajícímu skriptu konkrétní jednotky bude implicitně vložen vstupní bod programu "init". Init blok bude plnit funkci vstupních bodů pro další sekvenci bloků. Bude spuštěn vždy pouze jednou po startu jednotky. Jako vstupní bod budou sloužit i bloky "at event"a "at time". K jejich vyvolání dojde až po určité změně události nebo po časovém intervalu. Hardware prozatím neumožňuje spouštění událostmi, tento blok proto nebude implementován.

Kvůli vzniklému problému s vymazáním většiny operační paměti během uspání jednoky musí být uchování prováděné instrukce řešeno na úrovni generovaného Lua kódu. Veškerý stavový prostor interpretu Lua se totiž do malé RTC paměti nevejde. Lua kód bude muset před každou operací vedoucí do úsporného režimu poznačit aktuální pozici v kódu. Nutná je tedy i podpora na straně zařízení v podobě funkce umožňující zápis do RTC paměti. Po probuzení a inicializaci Lua skriptu bude vyčtena návratová pozice a proveden přesun na tuto pozici. Protože použitá verze Lua nepodporuje instrukci `goto`, musí být skript rozdělen na patřičných pozicích do podmíněných bloků, které jsou vykonány pouze pokud odpovídá pozice bloku s uloženou pozicí. Obdobnou možností by bylo rozdělení souvisejících sekvencí do očíslovaných funkcí a jejich vyvolání při probuzení. Interpret by ale v tomto případě byl nucen držet všechny funkce celou dobu v paměti, čímž by mohlo dojít k jejímu vyčerpání.

text		**práce s textem**
	join	konkatenace řetězců
	segment	získání podřetězce
humant interface		**rozhraní pro komunikaci s lidmi**
	text input	libovolný textový vstup
	select number	číselný vstup
	select value	výběr z předdefinovaného seznamu
	manual inspection	pozastavení běhu (přepnutí do menu)
	delay	pozdržení běhu programu v sekundách
machine interface		**rozhraní pro komunikaci se zařízeními**
	set pin	nastavení pinu na hodnotu
	read binary	vyčtení všech 3 vstupů (vektor)
	write binary	nastavení všech 3 vstupů (vektor)
	wait pin	čekání na hodnotu na pinu
	read current	vyčtení hodnoty proudu
	read pressure	vyčtení hodnoty tlaku
	read temperature	vyčtení teploty
data save		**uložení dat**
	save processing data	uložení zpracovaných dat
	process setup	konfigurace procesu
	save measure data	uložení naměřených dat
algorithm		
	fill variable	nastavení hodnoty promoměnné
	case	switch
	for	cyklus (for x in list)
	repeat times	while i > N: i++; / while true; pro N = -1
at		**Hlavní bloky**
	at event	vykonání skriptu na základě události
	at time	vykonání skriptu v časových intervalech
init	vstupní bod programu	

Tabulka 6.1: Dostupné bloky v Blockly

6.5 Propojení serveru se zařízeními

Server poskytuje rozhraní prostřednictvím HTTP definovaný metodami POST a GET. Koncové zařízení je postaveno do role klienta a všechny změny provedené na serveru se projeví až na vyžádání ze zařízení.

Vyžadovaná je podpora stažení uživatelských skriptů a odesílání získaných dat. Součástí rozhraní jsou i pomocné dotazy na verzi skriptu. Všechna data budou předávána metodou GET.

/config/lua/<NAME>	konfigurační Lua skript (byte kód)	kom
/lua_version/<NAME>	verze konfiguračního Lua skriptu	
/data_save	odeslání získaných dat ve formě json řetězce	
/log_err	logování chybových stavů	

Tabulka 6.2: Komunikační protokol serveru s koncovými jednotkami.

Odesílaná data budou ve forátu JSON. JSON je dostatečně jednoduchý formát aby se dal snadno produkovat na zařízení s omezenými prosstředky a zároveň plně postačuje k jednoduchému předávádí strukturovaných dat.

Formát JSON je následující.

```
{"name": <dev_name>, "type": <data_type>, "value": <value || array of values>}
```

Firmware bude přístupný na konfiguračním serveru přes protokol TFTP.

6.6 Plánovač procesů

Procesy (z pohledu Lua skriptů) spouštěné v časových intervalech musí být řízeny centrálním časovačem. Procesy nejsou z pohledu reálného času kritické a jejich pozdržení nepředstavuje žádné riziko. Navíc se neočekává jejich velké množství na jendnom zařízení a proto postačí jednoduchý FIFO plánovač bez prioritního zpracování. Tím odpadá řada synchronizačních problémů a problémů spojených s paměťovou náročností. První proces na procesoru dokončí svoji operaci a uvolní místo případnému dalšímu procesu ve frontě. Stejně tak procesy spouštěné externími událostmi počkají na dokončení rozpracovaného procesu.

6.7 Úložiště dat

Protože na serveru bude spuštěna databáze mongoDB pro ukládání Blockly, budou i zaznamenaná dat ukládána do této databáze. V případném budoucím provozu by byla data pravděpodobně zaznamenávána na stejný databázový server kam jsou ukládána data z výchozího projektu [37]. V této fázi vývoje není potřebné a ani vhodné do tohoto úložiště zasahovat. Do databáze bude prozatím ulženo jméno jednotky odkud pocházejí data, časové razítko získané při přijetí dat na konfiguračním serveru, typ dat a získaná hodnota.

Kapitola 7

Implementace software

Kapitola popisuje implementační detaily programového vybavení koncové jednotky i serveru. Důraz byl kladen především na využití existujících knihoven. Implementace firmware je postavena na operačním systému RTOS z vývojového prostředí esp-open-rtos.

Veškeré zdrojové kódy jsou dostupné na přiloženém CD. Obsah CD je shrnut v příloze A.

7.1 Lua

Jako základ je použit interpret elua v0.9. Konkrétně byla eLua naklonována z jejího hlavního github repozitáře (commit "bf7de2de"). Z celého obsahu byly ponechány pouze zdrojové soubory ve složce elua/src/lua, které obsahují modifikovaný originální interpret jazyka Lua v5.1.4. Zbylé soubory jsou testy, dokumentace a soubory s kódem vázaným na podporované MCU a externí periferie. Kromě kompilace pro MCU může být eLua zkompilována i pro PC. V kódu jsou tyto varianty odděleny konfiguračními makry (LUA_CROSS_COMPILER).

7.1.1 Kompilace

Pro potřeby tohoto projektu bylo nutné zdrojové kódy mírně modifikovat. Seznam provedených změn pro kompilaci na PC je následující:

- Odstraněn konflikt v redefinici základního datového typu long v souboru luaconf.h.
- V Makefile definováno makro LUA_CROSS_COMPILER
- Odstraněny include makra bránící kompilaci (linenoise.h (podpora interaktivního interpretu) v lstate.c)
- Přejmenována vstupní funkce lua_main na main v lua.c a

Pro modul WROOM-02 je vytvořena statická knihovna liblua.a. Knihovna je automaticky vytvořena při každém překladu, zde tedy stačí zastavit překlad před procesem linkování. Kroky provedené k jejímu získání jsou následující.

- Do Makefile bylo přidáno pravidlo spouštějící překlad s překladačem přiloženým k esp-open-sdk - xtensa-lx106-elf-gcc.

- Odebrány hlavičkové soubory specifikující některé funkce pro konrétní platformy z lstate.c, lbaselib.c a linit.c spolu s funkcemi které byly v souborech deklarovány.
- Odebrány hlavičkový soubor devman.h z lauxlib.c (rozhraní správce zařízení pro práci se soubory na dané platformě - nahrazeno za vlastní funkce viz níže).

7.1.2 Uživatelské funkce

Kromě změn nutných k překladu bylo nutné obohatit interpret o vlastní funkce povolující přístup k HW z lua skriptu. Kvůli modifikaci LTR popsané výše (3.5.2) musí být uživatelský modul přidán již před kompilací.

Přidání vlastního modulu vyžaduje definici vlastních funkcí pracujících se stavovým zásobníkem interpretu. Funkce jsou definovány v souboru lua_module.c/h. Vstupní parametry i návratová hodnota jsou předávány přes zásobník. Pro potřeby kompilace na PC k získání bytekódu vracejí funkce konstantní hodnoty typem i počtem odpovídající původní implementaci. Do interpretu (linit.c) je přiloženo pole ukazatelů na funkce a řetězce s jejich pojmenováním sloužící k jejich vyvolání z Lua kódu. Z lua kódu lze funkce volat s prefixem "minim."(interně je projekt označen jako "minimantra"). Např. pro vyčtení teploty: `minim.read_temperature()`) Veškeré implementované funkce jsou dohledatelné v souboru `lua_modules` na přiloženém CD.

7.1.3 Spouštění skriptů ze souboru

Jedna varianta jak spouštět kód lua je načtení z řetězce. Řetězec musí odpovídat ucelenému bloku kódu (modul, funkce) a nelze jej do interpretu vkládat postupně (z internetového toku nebo ze souboru). Kvůli tomu by, v případě většího skriptu, byla spotřebována velká část operační paměti na bufferování. Kromě neefektivnímu využití je problém i s nedeterminističností - nelze předpovídat velikost uživatelského skriptu, pokud by byl kód příliš velký, tak by zařízení nemuselo vůbec fungovat.

Běžnější varianta spouštění skriptů je spouštění ze souboru. Elua zjednodušuje načítání ze souboru tím, že k této činnosti vyžaduje pouze několik základních funkcí pro práci se soubory. Jsou to funkce pro otevření (open) a zavření souboru (close), načtení řetězce (read), načtení znaku (getc), navrácení přečteného znaku mezi nepřečtené (ungetc) a ověření konce souboru (eof). V tomto projektu je eLua upravena pro běh ze souborového systému SPIFFS (viz níže).

7.2 Firmware

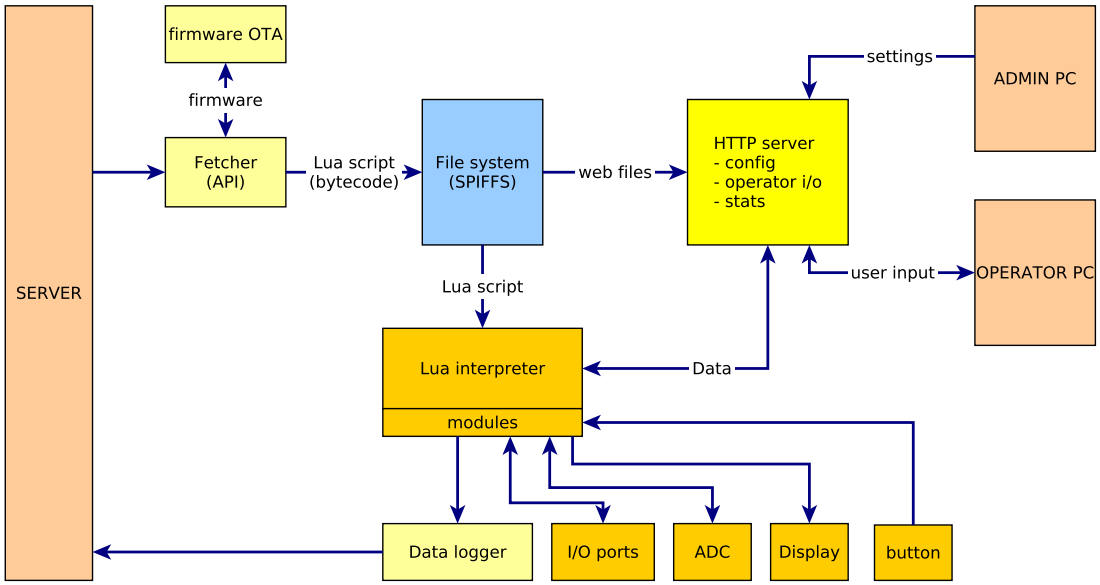
Firmware je vystavěn na operačním systému FreeRTOS, který je poskytován v rámci vývojového balíku esp-open-rtos. Ani jedna část aplikace neklade požadavky na zpracování v tvrdém reálném čase a proto je všem procesům přiřazena stejná prioritní skupina (všchny procesy se střídají po určitých časových kvantech).

Procesor esp8266 je taktován základním kmitočtem 80 Mhz.

Obrázek 7.1 zachycuje logické uspořádání firmware jednotky. Jednotlivé komponenty jsou popsány v dalších kapitolách.

Programové vybavení využívá řadu knihoven z vývojového balíku esp-open-rtos. Jsou to knihovny:

- SPIFFS



Obrázek 7.1: Blockový diagram firmware.

- I2C (bitbang) Knihovna emulující funkci sběrnice I2C v software. Při použití 80 MHz hodinového signalu jsou podporovány rychlosti: 100 kHz a 400 KHz.
- SSD1306 práce s grafickým OLED displejem.
- httpd http server poskytovaný spolu s IP stackem LWIP. Nabízí poměrně plnohodnotný http server s možností CGI, SSI a také s možností využití protokolu websocket.
- rboot-OTA Katualizace přes WiFi. Používa dva 1kB sloty. Nejdříve provede zápis do aktuálně nevyužívaného slotu a poté se do něj přepne. Jeho součástí je kontrola integrity stahovaného souboru přes hash sha256.

7.2.1 Komunikace se serverem

Komunikace je implementována pomocí socketů. Modul slouží ke stažení lua skriptu, nového firmware a odeslání nasbíraných dat prostřednictvím minimální implementace HTTP GET metody.

7.2.2 SPIFFS

SPI Flash file system, souborový systém vytvořený na flash paměti. Knihovna nabízí základní funkce pro práci se soubory jako je vytvoření, čtení a zápis do souboru. Adresářová struktura není podporována, v názvech souborů jsou však podporovány lomítka ("/") a při překladu jsou soubory přejmenovány tak, aby byly přístupné pod stejným názvem jako v původním systému (unixový styl). Výhoda systému oproti jiným, jako je např. FAT, který je také dostupný v podobě přiložených knihoven, je v rovnoměrném zatěžování flash paměti. Díky tomu se zvyšuje její životnost.

Použitá konfigurace SPIFFS povoluje 7 otevřených souborů najednou.

7.2.3 Lua

V rámci firmware je Lua spouštěna obdobně jako na PC - je inicializován stavový prostor interpretu a následně načten a spuštěn lua bytekód ze SPIFFS.

Kromě zásahů do interpretu byly do firmware přidány některé chybející funkce ze standardní knihovny jazyka c (newlib). Jsou to funkce _exit, _times_r, _getpid_r a _rename_r. V běžném provozu je interpret nepoužívá a tak funkce nic užitečného neprovádějí. Pouze pro jistotu by funkce v případě vyvolání vypsaly ladící informaci na komunikační UART výstup.

7.2.4 RTC paměť

Celková paměť má kapacitu 500 B. Část této paměti slouží k uložení dat bootloaderu. Kvůli špatné dokumentaci není jasné, zda je zbytek paměti nějak využíván a tak byla potřebná paměť pro uložení vlastních dat vybrána experimentálně. Platnost dat je po startu zařízení ověřena 16 bitovým CRC součtem. Díky tomu lze rozpoznat jestli bylo zařízení pouze probuzeno ze spánku, nebo poprvé zapnuto.

7.2.5 Úsporný režim

Časovač určený k probuzení ze spánku je z hardwarových důvodů omezen na maximální hodnotu 71 minut (32 bitový register). V případě, kdy by bylo potřeba jednotku uspat na delší časový úsek je požadovaný čas zaznamenán do RTC paměti a modul je po každé celé hodině probouzen a pokud daný čas nevypršel, znovu uspán.

7.2.6 Lokální HTTP server

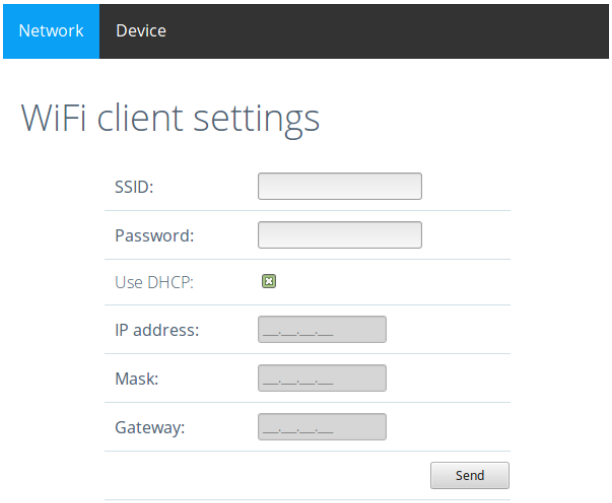
Soubory odesílané HTTP serverem jsou v základní verzi ukládány do programové části paměti jako data. Vzhledem k velikosti projektu, jehož kód zabírá nemalou část 1 kB programové paměti, by byly soubory značně omezeny. Proto je HTTP server upraven tak, aby namísto souborů v programové paměti četl soubory z přítomného SPIFFS. K tomuto účelu stačilo implementaci HTTP serveru zaměnit funkce pro otevření, čtení a uzavření souboru. Díky změně je možné na serveru používat poměrně velké frameworky a knihnovny usnadňující tvorbu uživatelsky přívětivých webových rozhraní. Pro grafický vzhled stránky byl zvolen minimalistický CSS framework siimple a k zjednodušení práce s javascriptem knihovna jquery. Konfigurační webové rozhraní je tvořeno HTML formulářem.

V kódu jsou data přístupná prostřednitvím callback funkce HTTP serveru.

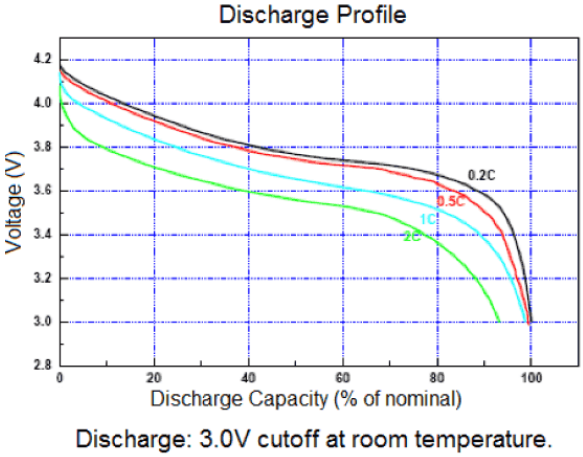
Protože se nepředpokládá příliš častá změna této konfigurace, jsou data uložena do SPIFFS.

7.2.7 Uživatelské rozhraní

Zvolené SDK nabízí poměrně propracovanou knihovnu pro práci s OLED displejem. Knihovna počítá se všemi variantami připojení tohoto displeje a k jeho využití stačí pouze doplnit příslušná čísla pinů I2C. Na displej jsou vypisovány zprávy z Lua skriptu. V pravém horním rohu je zobrazen stav akumulátoru v podobě piktogramu. Jeho výplň udává stav akumulátoru. Stav je vyčten z ADC periferie zabudované na WROOM-02. Míra nabití je přibližně odhadnuta podle napětí udaného na obrázku 7.3.



Obrázek 7.2: WiFi nastavení na koncovém zařízení.



Obrázek 7.3: Vybíjecí profil Li-Ion akumulátoru [1].

7.2.8 Sběr dat

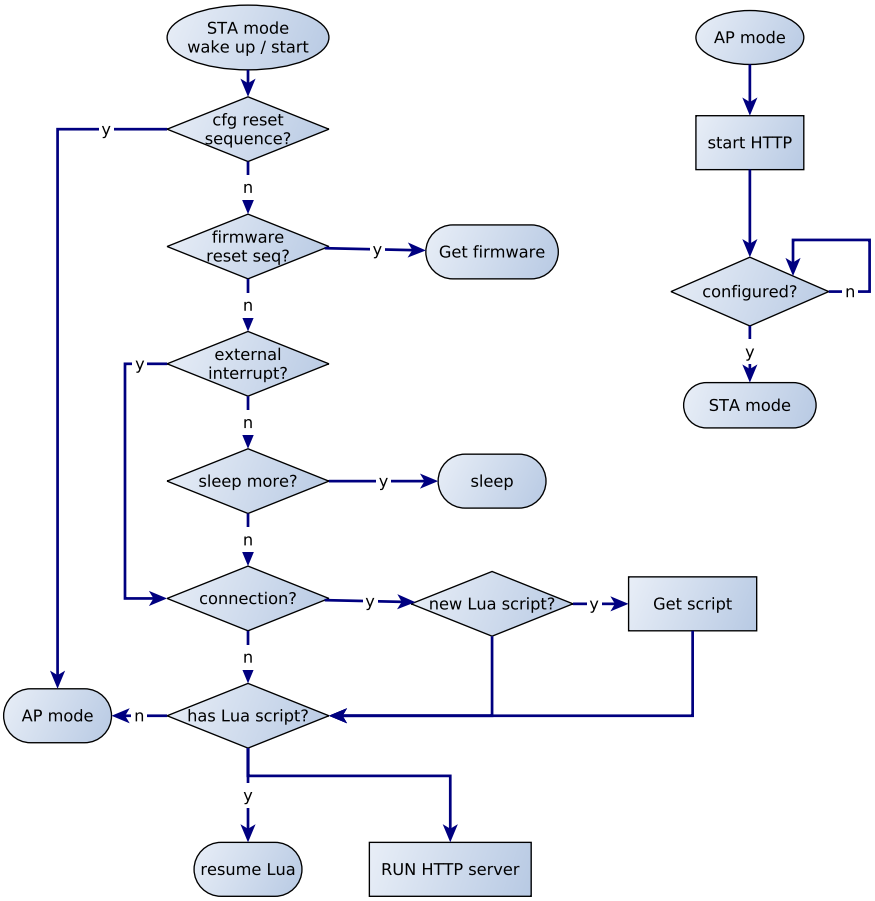
Sběr dat je iniciován z Lua skriptu. Podporováno je vyčtení hodnoty z teploměru a získání hodnoty z externího ADC prostřednictvím I2C sběrnice. U této části je předpoklad budoucí modifikace a rozšíření v závislosti na reálných měřicích požadavcích.

7.2.9 Režimy běhu

Podle návrhu popsané výše jsou implementovány 2 režimy běhu zařízení. Průběh obou režimů zaznamenává diagram na obrázku 7.4.

AP (access point) mód slouží pouze ke konfiguraci zařízení. Mód je spuštěn v případě, že se jednotka nemůže připojit do nakonfigurované sítě a mód opouští buďto restartováním zařízení nebo změnou konfigurace jednotky přes webové rozhraní.

Druhý, STA (station) mód, je mód pracovní. Zde probíhá spouštění získávání a spouštění uživatelských skriptů. V tomto módu je také spuštěn konfigurační HTTP server. Na schématu je zachycena celá inicializační sekvence.



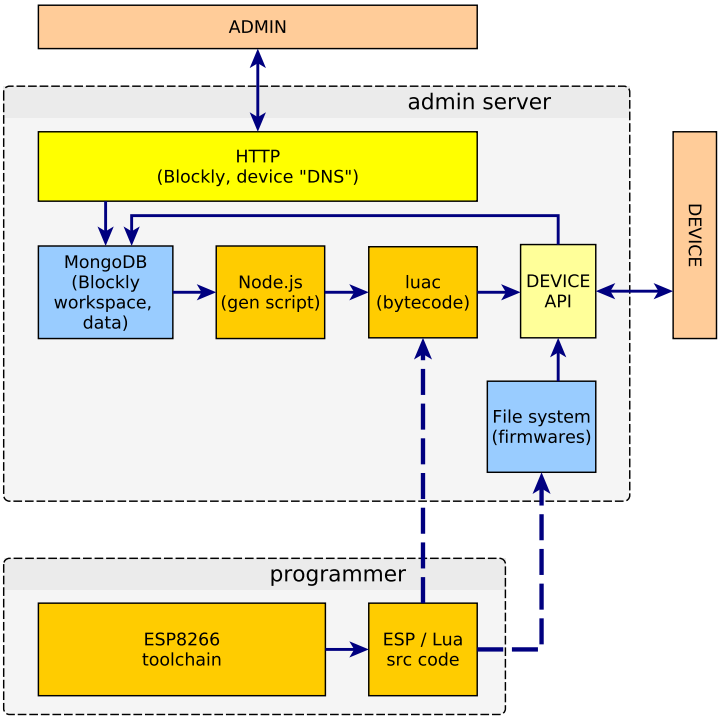
Obrázek 7.4: Tok programu koncového zařízení.

7.3 Server

Serverová aplikace je postavená z větší části na existujícím používaném řešení. Obrázek 7.5 zachycuje server po úpravě. Úplně nový je zde překlad skriptu do bytekódu Lua, rozhraní zprostředkovávající komunikaci mezi jednotkou a datovou databází, samotná databáze a uživatelské rozhraní umožňující vzdálený přístup na jednotky (v podobě odkazů) a rozhraní kde jsou potvrzeny provedené změny skriptu.

7.3.1 Blockly

Nasazené Blockly vychází ze starší verze, která ještě neměla zabudovanou podporu pro jazyk Lua. Kvůli kompatibilitě s existujícím systémem není možné přímo použít novější verzi Blockly. Proto byl z novější verze odebrány soubory rozšiřující funcionalitu o Lua a doplněny ke starší použité verzi. Blockly se v části generující výsledný skript nezměnilo, takže zde nebyly potřeba žádné další zásahy.



Obrázek 7.5: Blokové schéma serverové aplikace.

Uživatelsky definované speciální bloky ve výchozí aplikaci mají svázaný vzhled a chování bloků spolu s výstupním kódem. Pro větší přehlednost, ale hlavně kvůli způsobu jakým je generován výsledný kód jsou chování a vzhled rozděleny do vlastních souborů. Výstupní jazyk je odvozen z přiloženého blockly souboru, který je pro každý jazyk jiný. Spolu s rozdělením generujícího kódu je upraven generující shell skript (generace probíha na straně serveru pomcí Node.js)

Na úroveň sekcí v hierarchii úrovní (viz 3.7) byl přidán blok reprezentující jednotky navržené v této práci.

Na přiloženém CD se nachází výstup z linuxové utility diff popisující všechny provedené změny v konfiguračním serveru oproti originální verzi.

Bloky "at"z tabulky 6.1 nejsou implementovány ani na serveru (je definován pouze vzhled) a ani ve firmware. Proto není realizován ani plánovač procesů zmíněný v návrhu.

7.3.2 Firmware

Firmware je umístěn na serveru v binární podobě, tak jak je vytvořen při kompilaci, v adresáři firmware. Ke sdílení firmware je použit TFTP server tftp-hpa. Použít však lze libovolný TFTP server pracující na standardním TFTP portu 69.

7.3.3 Uživatelské rozhraní

Rozhraní pro přistup uživatele k jednotkám a potvrzení změny uživatelského kódu jsou na adrese serveru `/devices`. Stránka obsahuje seznam všech konfigračních skriptů spojený s jednotkami, které si daný skript vyžádali. Každá položka obsahuje navíc potvrzení změny

jedtnotlivých skriptů. Změněné skripty si může jednotka stáhnou až po potvrzení a nikoliv ihned po změně skriptu jako je tomu nyní.

7.3.4 Databáze

Data jsou vkládána do mongoDB databáze "data_log"na konfiguračním serveru prostřednictvím python modulu pymongo. Uvnitř databáze jsou uložena v kolekci data. Formát ukládaných dat je specifikován v návrhu.

7.4 Překlad a spuštění

Zdrojové kódy jsou rozděleny do 3 složek:

firmware

Překlad firmware se provádí pomocí GNU Make - příkaz "make flash"přeloží a naprogramuje koncovou jednotku. Ke správnému překladu je vyžadována instalace vývojových prostředí esp-open-sdk, esp-open-rtos a správně nastavené cesty k binárním souborům.

Lua

Lua se také překladá příkazem "make". Na základě parametru překladu je vytvořena buďto spustitelná verze ("make generic", využívá gcc) obsahující překladač Lua skriptu do bytekódu kompatibilního s verzí v ESP8266, nebo je vytvořena statická knihovna "liblua.a"("make esp", využívá kompiler dodaný s esp-open-sdk) nutná pro překlad firmware.

config_server

Vyžaduje python verze 3 a překladač jazyka CoffeeScript. Do pythonu je nutné doinstalovat další moduly importované na začátku souboru config_server.py. Server se spouští příkazem "python3 config_server.py –devel"a musí být spouštěn s oprávněním správce.

Kapitola 8

Ověření funkčnosti

Zde jsou popsány postupy jaké byly použity k ověření požadovaných funkcí.

8.1 GPIO

Vstupní a výstupní piny byly ověřeny připojením stabilizovaného laboratorního zdroje. Na vstupní porty bylo přivedeno napětí 5 V a 24 V, tedy minimální a maximální požadovaný rozsah. Při testu byl u obou případů v modulu WROOM-02 správně detekován stav log. 1 a zároveň nedošlo k poškození jednotky. Výstup byl ověřen obdobně. Na výstupní piny zařízení bylo připojeno napětí v maximálním a minimálním rozsahu a byl sepnut výstup v modulu. Zde také došlo ke správnému sepnutí bez poškození jednotky.

Vstupní i výstupní piny jsou po zapnutí automaticky inicializovány jako vstupy, takže nemůže dojít k nechtěnému sepnutí přístroje, který by byl připojen ještě před zapnutím.

8.2 Interpret jazyka Lua

Jedním z hlavních cílů tohoto projektu bylo ověřit možnosti zabudování interpretru jazyka Lua z hlediska dostupných zdrojů na SoC.Jeho funkčnost byla ověřena základními testovacími skripty z oficiální distribuce Lua verze 1.5.4, ze které použitá verze vychází (vynechány byly skripty vyžadující některý z Lua modulů, který byl v této práci odstraněn). Skripty jsou přiloženy na CD. Základní testy ověřující standardní funkcionalitu proběhly v pořádku. Problém nastal u skriptů vyžadujících větší množství paměti jakým je např. hra Life.

Vlastní implementované funkce ovládající hardware fungují korektně.

Při spuštění samotného interpretu na skriptu jehož jedinou úlohou je výpis množství zabrané paměti běžícím interpretem je využíváno 5 KB paměti. Nevyužité paměti zůstává 26 KB. Množství použité paměti je zjišťováno z Lua skriptu voláním:

```
print(collectgarbage("count")*1024)
```

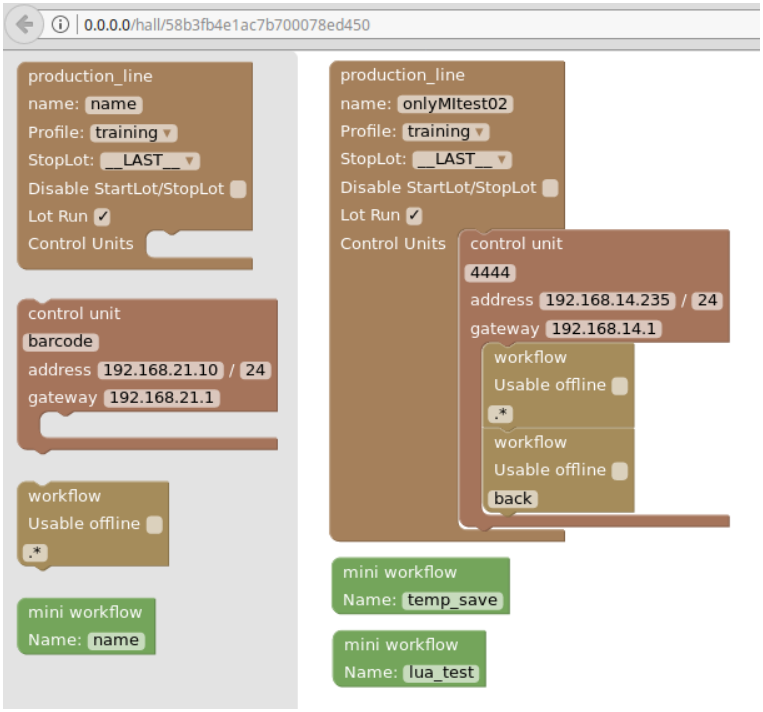
Nevyužitá paměť je mapována z firmware pomocí FreeRTOS funkce:

```
xPortGetFreeHeapSize()
```

Při ověření na skriptu provádějícím funkce, které lze předpokládat při případném nasazení - tj. cycklický sběr dat se spínáním výstupního portu a logováním dat se využitá paměť pohybovala okolo 10 KB.

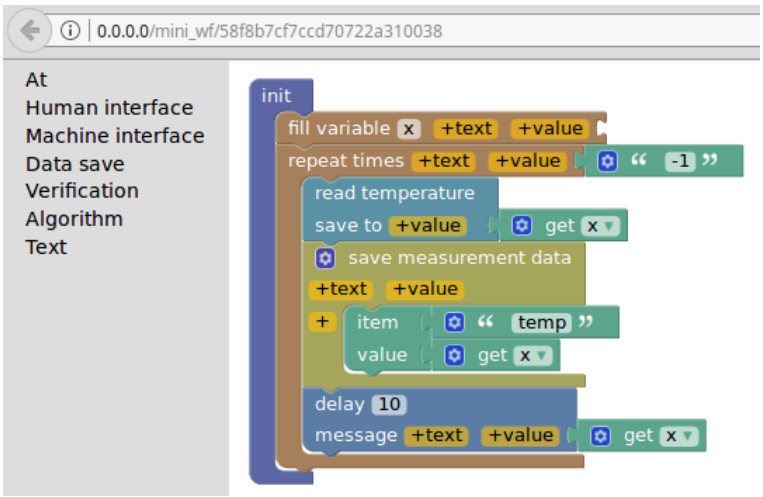
8.3 Uživatelský skript

V uživatelském prostředí blockly byly vytvořeny tři bloky odpovídající třem koncovým jendotkám. Jeden z bloků byl vytvořen v jiné sekci uvnitř téže haly. Dvojici bloků zachycuje obrázek 8.1.



Obrázek 8.1: Nově přidané bloky (mini workflow).

V bloku náležejícím jednotce pojmenované temp_save byl vytvořen jednoduchý skript, který po 10 sekundových intervalech sbírá data z teplotního čidla a odesílá je na server (viz obrázek 8.2).

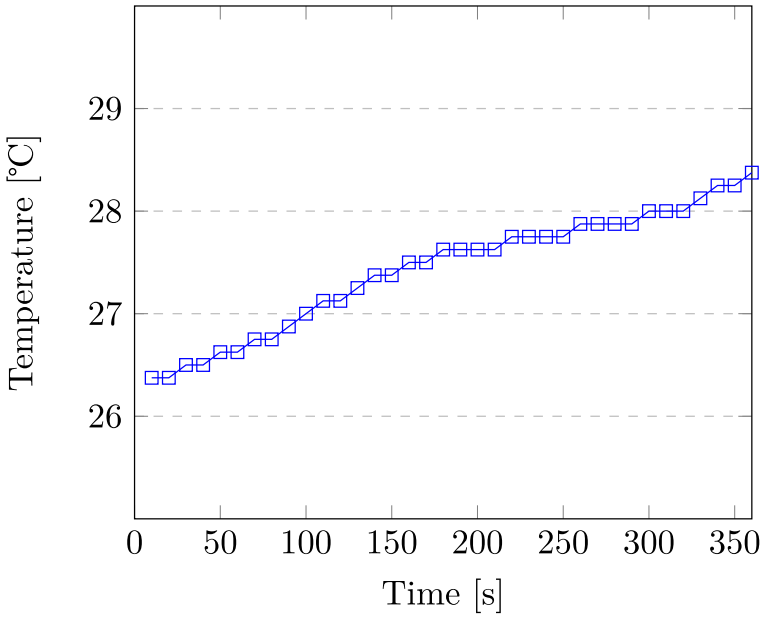


Obrázek 8.2: Skript pro sběr teploty v Blockly.

Aby mohla jednotka skript stáhnout, byly provedené změny nejprve potvrzeny v konfiguračním seznamu jednotek. Seznam je zachycen na obrázku 8.3. Obrazovka byla zaznamenána až po stažení skriptu jednotkou a je zde tedy vidět adresa jednotky, která tento skript stáhla. Ostatní skripty nejsou vuyžívány.

Obrázek 8.3: Seznam spárovaných jednotek.

Zaznamenaná data byla korektně uložena do databáze. Skript byl spuštěn na jednotce, která byla před tím delší dobu vypnutá a v okolí byla stabilní pokojová teplota. Získané hodnoty jsou vykresleny v grafu 8.4. Z grafu je vidět dopad umístění teplotního čidla do blízkosti modulu WROOM-02.



Obrázek 8.4: Skript pro sběr teploty v Blockly.

Kapitola 9

Závěr

V první části práce byly analyzovány požadavky zadavetele kladené na výslednou bezdrátovou senzorickou platformu. Požadavky se dotýkají jak hardwarového provedení (zařízení pro sběr dat), tak také serverové části (uživatelská konfigurace a uložení dat).

V následující části byly teoreticky popsány prvky potřebné ke zhotovení měřicí jednotky. Kapitola zahrnuje bezdrátový modul ESP-WROOM-02 jenž byl zvolen zadavatelem jako hlavní řídicí prvek a dostupné programovací prostředky nezbytné k naprogramování modulu. Shrnuty byly také dostupné principy měřicích senzorů pro požadovaná data. V poslední části kapitoly byla zvážena možnost nasazení interpretu skriptovacího jazyka Lua na zvoleném bezdrátovém modulu - především z pohledu paměťové náročnosti a použitelnosti pro měřící účely.

Ve třetí kapitole bylo přistoupeno k samotnému návrhu vývojové desky senzorické platformy. Centrálním řídícím prvkem platformy se stal bezdrátový WiFi modul ESP-WROOM-02. Výsledný návrh dovoluje připojení různých druhů měřících čidel. Ze zařízení jsou vyvedeny vstupy umožňující připojení libovolného analogového čidla napájeného 3,3 V nebo 5 V s výstupem do 3,3 V. Jeden vstup je navržen specificky pro zavedení proudového snímače fungujícího na principu proudového transformátoru. Jednotka dále umožňuje připojení snímače s digitálním I2C výstupem. Platforma je uzpůsobena i k propojení externích systémů - může spínat externí zařízení i číst jeho výstupy. K napájení byl zvolen Li-Ion akumulátor, případně lze jednotku napájet mini USB konektorem. K interakci s uživatelem byl zvolen OLED displej a jedno kapacitní tlačítko.

Obvod byl navržen na dvouvrstuvou desku plošných spojů. Deska byla vyrobena externí specializovanou firmou a následně ručně zapájena. Po zhotovení se ukázalo několik nedostatků opomenutých v návrhu. Některé menší nedostatky byly opraveny přímo na desce, část ovšem bude muset být opravena až na případné další verzi jednotky. Mezi závažnější nedostatky patří nemožnost probudit jednotku ze spánku na základě externí události a nemožnost odpojení externích periferií od napájení právě v době spánku. Tento problém byl odhalen až při samotných pokusech se spánkovým režimem obvodu ESP8266. Ukázalo se také, že využití kapacitního snímače pod plochu OLED displeje není dostatečně spolehlivé a mělo by být dále nahrazeno za běžný spínač. Vyrobená deska nicméně postačovala k ověření základních principů.

Následně byl proveden návrh firmware a navrženy změny software u existující serverové části. Návrh firmware zahrnuje možnost konfigurace koncových jednotek z lokálního HTTP (nastavení WiFi sítě a jednoznačného názvu jednotky), aktualizaci firmware prostřednictvím bezdrátové sítě a získání konfiguračního Lua skriptu a jeho spuštění. Dále v rámci Lua skriptu získávání a ukládání naměřených dat do databáze prostřednicvím konfigurač-

ního serveru. Lua skripty jsou tvořeny v uživatelsky přívětivém vizualním programovacím nástroji Blockly. Návrh serverové části obsahuje modifikační úpravy rozšiřující server o nový typ bloku reprezentující koncovou jednotku. Uspořádání bloků uvnitř tohoto bloku představuje skript náležející jednomu koncovému zařízení.

U implementace software byl kladen důraz na zabudování interpretu jazyka Lua. V projektu byl úspěšně použit interpret vycházející z verze eLua. Interpret byl rozšířen o vlastní funkce umožňující práci s hardware ze samotného skriptu. Byl také zaveden konfigurační HTTP server. Na serveru byly provedeny změny potřebné ke generování Lua skriptu, jeho distribuci a ukládání dat.

Z původně navržených snímačů byl prakticky ověřen pouze teplotní snímač. Ukázalo se, že se na desce nachází příliš blízko modulu ESP-WROOM-02, který zahřívá své okolí a zkresluje tím měření. Senzory proudu a tlaku bohužel z časových důvodů nebyly použity. K ověření základní funkčnosti jednotky je však teplotní snímač dostačující. Další původně zamýšlená část, která byla v projektu opomenuta je prouzkoumání možnosti dlouhodobějšího běhu jednotky z bateriového napájení. Nestalo se tak z podobného důvodu jako u senzorů, ale také kvůli nedostatkům v návrhu hardware. Ten totiž nyní neumožňuje vypínání nepotřebných externích periferií při spánku.

Z hlediska software projekt dokázal, že je využití bezdrátového modulu WROOM-02 dostačující i když je potřeba počítat s jistými omezeními (režim spánku vymaže RAM paměť, uchová se pouze malá RTC paměť). Z pohledu hardware je modul také použitelný, avšak kvůli svému omezenému počtu vstupně výstupních pinů a dalších periferií je nutné doplnit tuto funkčnost externě. To značně zvyšuje cenu a také rozměry výsledné jednotky. Možným řešením je nahrazení všech externích periferií jedním Mikrokontrolérem, který by řešil veškerou obsluhu hardware a řídil režimy spánku. Nevýhodou by bylo zvýšení komplexity FW. Dalším možným řešením by byla výměna WROOM-02 za výkonnější postavené např. na SoC ESP32. Odpadla by nutnost použití další externí součásti starající se o sběr dat nebo digitální výstup.

Literatura

[1] Adafruit: *Li-Ion and LiPoly batteries - voltages*. 2015, [Online; navštíveno 10.5.2017].
URL https://learn.adafruit.com/li-ion-and-lipoly-batteries/voltages

[2] Aiohttp_contributors: *aiohttp: Asynchronous HTTP Client/Server*. 2017, [Online; navštíveno 8.5.2017].
URL https://aiohttp.readthedocs.io/

[3] BatteryUniversity: *BU-409: Charging Lithium-ion*. 2016, [Online; navštíveno 30.12.2016].
URL http://batteryuniversity.com/learn/article/charging_lithium_ion_batteries

[4] BuyDisplay: *Serial SPI 1.3"128x64 OLED Display Module SSD1306*. 2013, [Online; navštíveno 2.1.2017].
URL http://www.buydisplay.com/default/serial-spi-1-3-inch-128x64-oled-display-module-ssd1306-white-on-black

[5] Espressif: *ESP-WROOM-02 Datasheet*. [Online; navštíveno 29.12.2016].
URL http://espressif.com/sites/default/files/documentation/0c-esp-wroom-02_datasheet_en.pdf

[6] Espressif: *ESP-WROOM-02 PCB Design and Module Placement Guide*. [Online; navštíveno 29.12.2016].
URL http://espressif.com/sites/default/files/documentation/esp-wroom-02_pcb_design_and_module_placement_guide_0.pdf

[7] Espressif: *ESP8266 AT Instruction Set*. Červenec 2016, [Online; navštíveno 29.12.2016].
URL https://espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf

[8] Espressif: *ESP8266 Datasheet*. Prosinec 2016, [Online; navštíveno 29.12.2016].
URL https://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[9] Espressif: *ESP8266 Non-OS SDK API Reference*. Prosinec 2016, [Online; navštíveno 29.12.2016].
URL https://espressif.com/sites/default/files/documentation/2c-esp8266_non_os_sdk_api_reference_en.pdf

[10] Espressif: *ESP8266 NONOS SDK V2.0.0 20160810*. Srpen 2016, [Online; navštíveno 29.12.2016].

URL https://espressif.com/sites/default/files/sdks/esp8266_nonos_sdk_v2.0.0_16_08_10.zip

[11] Espressif: *ESP8266 RTOS SDK Programming Guide*. Srpen 2016, [Online; navštíveno 29.12.2016].
URL https://espressif.com/sites/default/files/documentation/20a-esp8266_rtos_sdk_programming_guide_en.pdf

[12] Espressif: *ESP8266 RTOS SDK V1.5.0*. Listopad 2016, [Online; navštíveno 29.12.2016].
URL https://github.com/espressif/ESP8266_RTOS_SDK

[13] Espressif: *ESP8266 System Description*. Červenec 2016, [Online; navštíveno 29.12.2016].
URL https://espressif.com/sites/default/files/documentation/0b-esp8266_system_description_en.pdf

[14] Forum, E. C.: *ESP8266 Community Forum*. 2016, [Online; navštíveno 2.1.2017].
URL http://www.esp8266.com/

[15] Google: *A library for building visual programming editors*. 2016, [Online; navštíveno 8.1.2017].
URL https://developers.google.com/blockly/

[16] Gratton, A.; Others: *esp-open-rtos*. [Online; navštíveno 29.12.2016].
URL https://github.com/SuperHouse/esp-open-rtos

[17] Kolban, N.: *Kolban's Book on the ESP32 & ESP8266*. Listopad 2016, [Online; navštíveno 29.12.2016].
URL https://leanpub.com/ESP8266_ESP32

[18] Majling, E.: *Transformátor – základní vlastnosti a dělení*. Duben 2015, [Online; navštíveno 30.12.2016].
URL http://oenergetice.cz/technologie/elektroenergetika/transformator-zakladni-vlastnosti-a-deleni/

[19] Marinescu, B.: *Lua Tiny RAM*. 2011, [Online; navštíveno 8.1.2017].
URL http://www.eluaproject.net/doc/v0.9/en_arch_ltr.html

[20] Mertens, P.: *Current transformer*. Červenec 2012, [Online; navštíveno 31.12.2016].
URL http://www.openelectrical.org/wiki/index.php?title=Current_transformer

[21] Microchip: *Stand-Alone Linear Li-Ion / Li-Polymer Charge Management Controller*. 2006, [Online; navštíveno 2.1.2017].
URL http://ww1.microchip.com/downloads/en/DeviceDoc/22005a.pdf

[22] Microchip: *MCP1640/B/C/D*. 2011, [Online; navštíveno 2.1.2017].
URL http://ww1.microchip.com/downloads/en/DeviceDoc/22234B.pdf

[23] Microchip: *Microchip Capacitive Proximity Design Guide*. 2013, [Online; navštíveno 4.1.2017].
URL http://ww1.microchip.com/downloads/en/AppNotes/01492A.pdf

[24] MongoDB: *MongoDB Atlas Database as a Service*. 2017, [Online; navštíveno 8.5.2017].
URL https://www.mongodb.com/

[25] NodeMCU_Team: *NodeMCU*. 2014, [Online; navštíveno 8.1.2017].
URL http://www.nodemcu.com/index_en.html

[26] pfalcon; Others: *esp-open-sdk*. [Online; navštíveno 29.12.2016].
URL https://github.com/pfalcon/esp-open-sdk

[27] ROSSETACODE: *Sorting algorithms*. 2016, [Online; navštíveno 8.1.2017].
URL https://rosettacode.org/wiki/Sorting_algorithms/Bubble_sort

[28] SiliconLabs: *SINGLE-CHIP USB TO UART BRIDGE*. 2016, [Online; navštíveno 8.1.2017].
URL
https://www.silabs.com/Support%20Documents/TechnicalDocs/CP2102-9.pdf

[29] Slavíček, O.: *Dynamické charakteristiky běžně používaných snímačů tlaku*. Vysoké učení technické v Brně. Fakulta strojního inženýrství. 2016, [Online; navštíveno 29.12.2016].
URL http://hdl.handle.net/11012/60905

[30] SolomonSystech: *SSD1306*. Duben 2008, [Online; navštíveno 2.1.2017].
URL https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf

[31] Sparkfun: *Non-Invasive Current Sensor - 30A*. 2014, [Online; navštíveno 2.1.2017].
URL https://www.sparkfun.com/products/11005

[32] ST: *LD1117*. 2013, [Online; navštíveno 2.1.2017].
URL www.st.com/resource/en/datasheet/ld1117.pdf

[33] TexasInstruments: *PCA9557 Remote 8-Bit I2C and SMBus Low-Power I/O Expander*. 2014, [Online; navštíveno 2.1.2017].
URL http://www.ti.com/lit/ds/symlink/pca9557.pdf

[34] TexasInstruments: *ADS101x Ultra-Small, Low-Power I2C-Compatible, ADC*. 2016, [Online; navštíveno 3.5.2017].
URL http://www.ti.com/lit/ds/symlink/ads1015.pdf

[35] TME: *ESPRESSIF ESP-WROOM-02*. [Online; navštíveno 8.1.2017].
URL http://www.tme.eu/en/details/esp-wroom-02/communication-modules-various/espressif/

[36] TwoWayRadioGear: *Comparison of NiCd, NiMH, and Li-Ion Batteries* . [Online; navštíveno 30.12.2016].
URL https://www.twowayradiogear.com/uploads/nicd%20nimh%20li-ion.pdf

[37] Vargovčík, P.: *Řídicí jednotka výrobní linky*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. 2015, [Online; navštíveno 8.1.2017].
URL http://hdl.handle.net/11012/39170

Přílohy

A Obsah přiloženého CD

B Schéma zapojení

C Výsledná DPS

D Kusovník

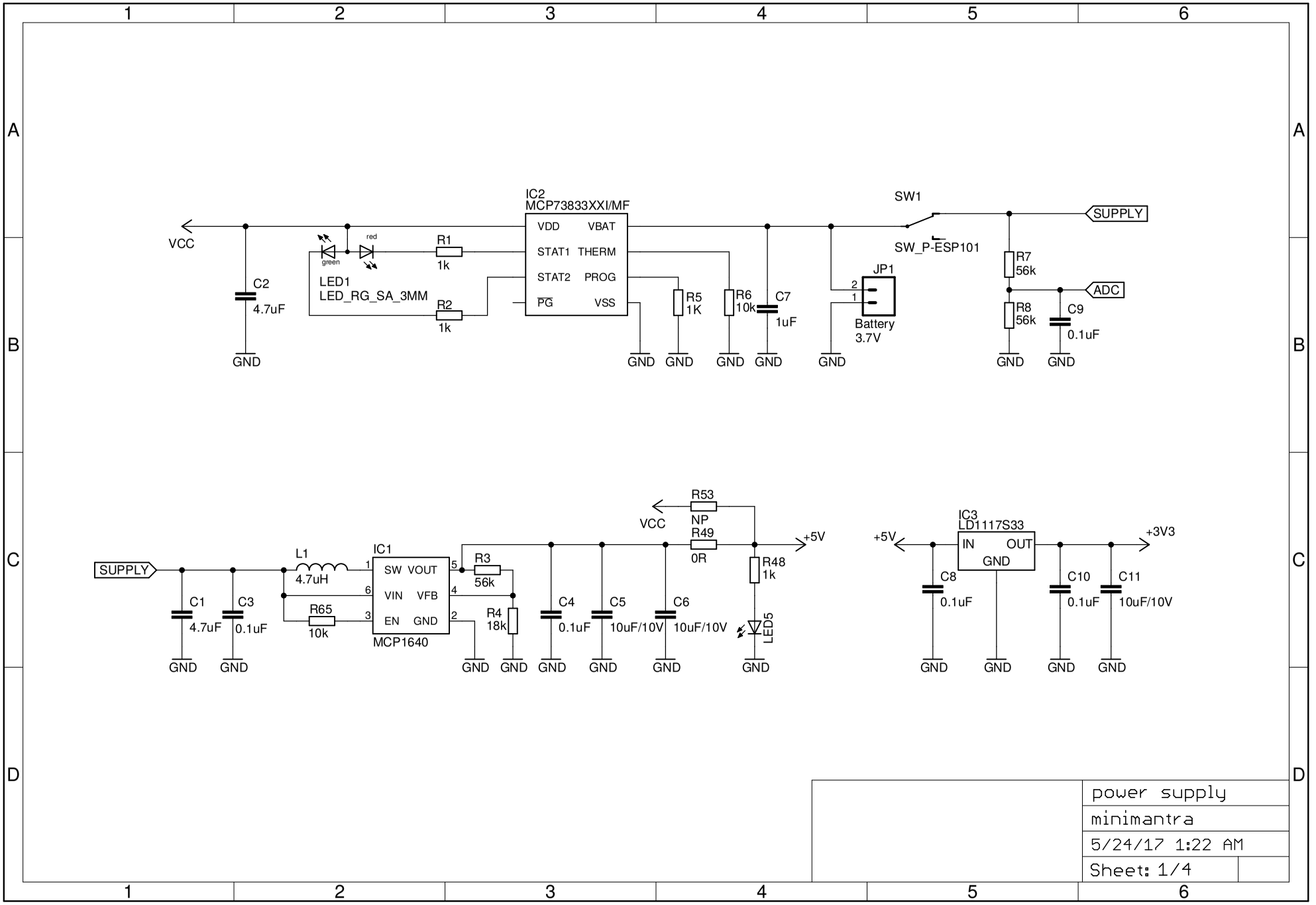
Příloha A

Obsah přiloženého CD

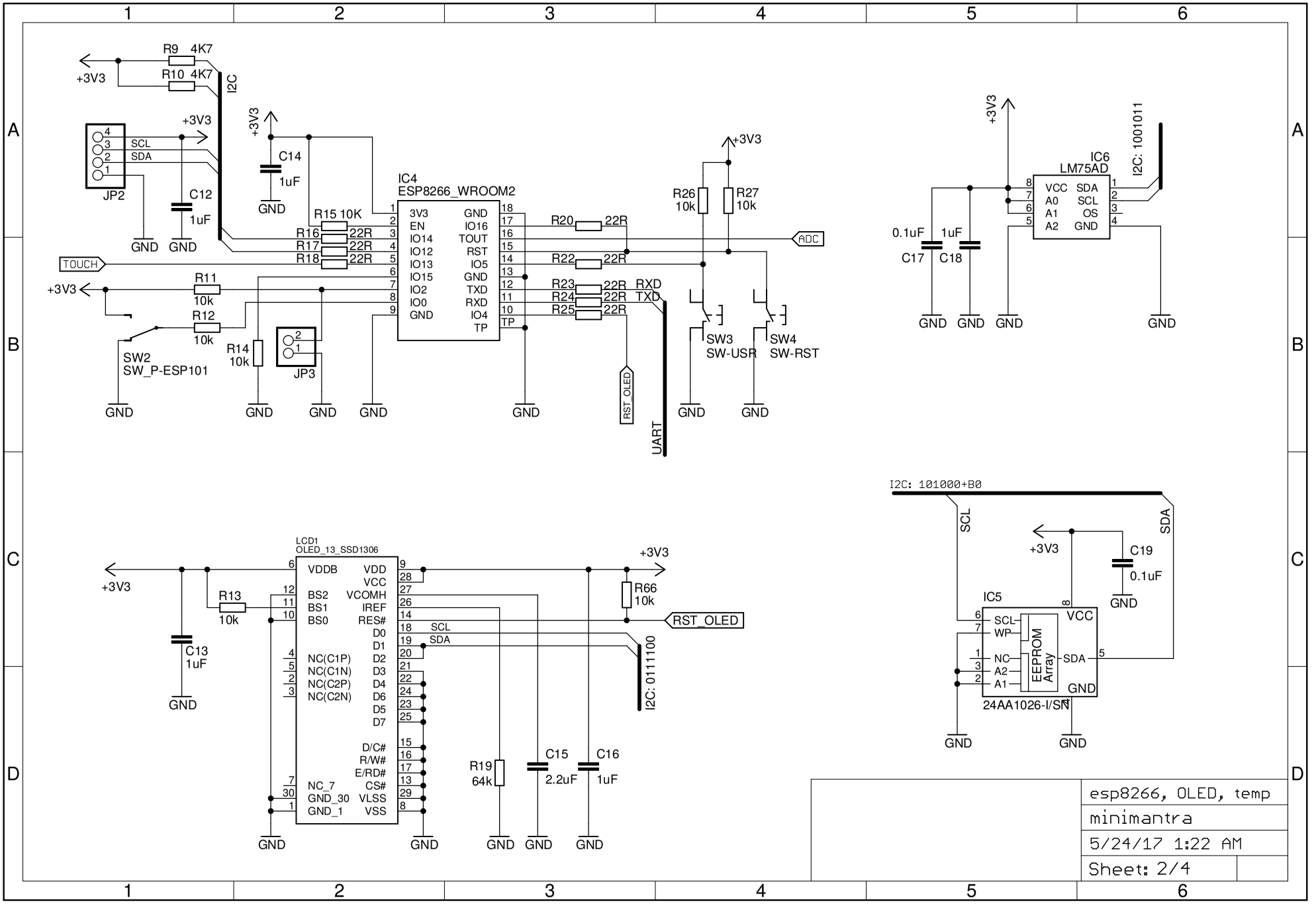
- Soubor readme.txt stručne popisující obsah CD.
- Tento dokument - pdf i jeho zdrojová verze.
- Fotografie zhotoveného zařízení.
- Obvodová schémata ve formátu PDF.
- Schémata a navržená deska ve formátu programu Eagle.
- Zdrojové soubory firmware a serverové aplikace včetně návodu k překladu.

Příloha B

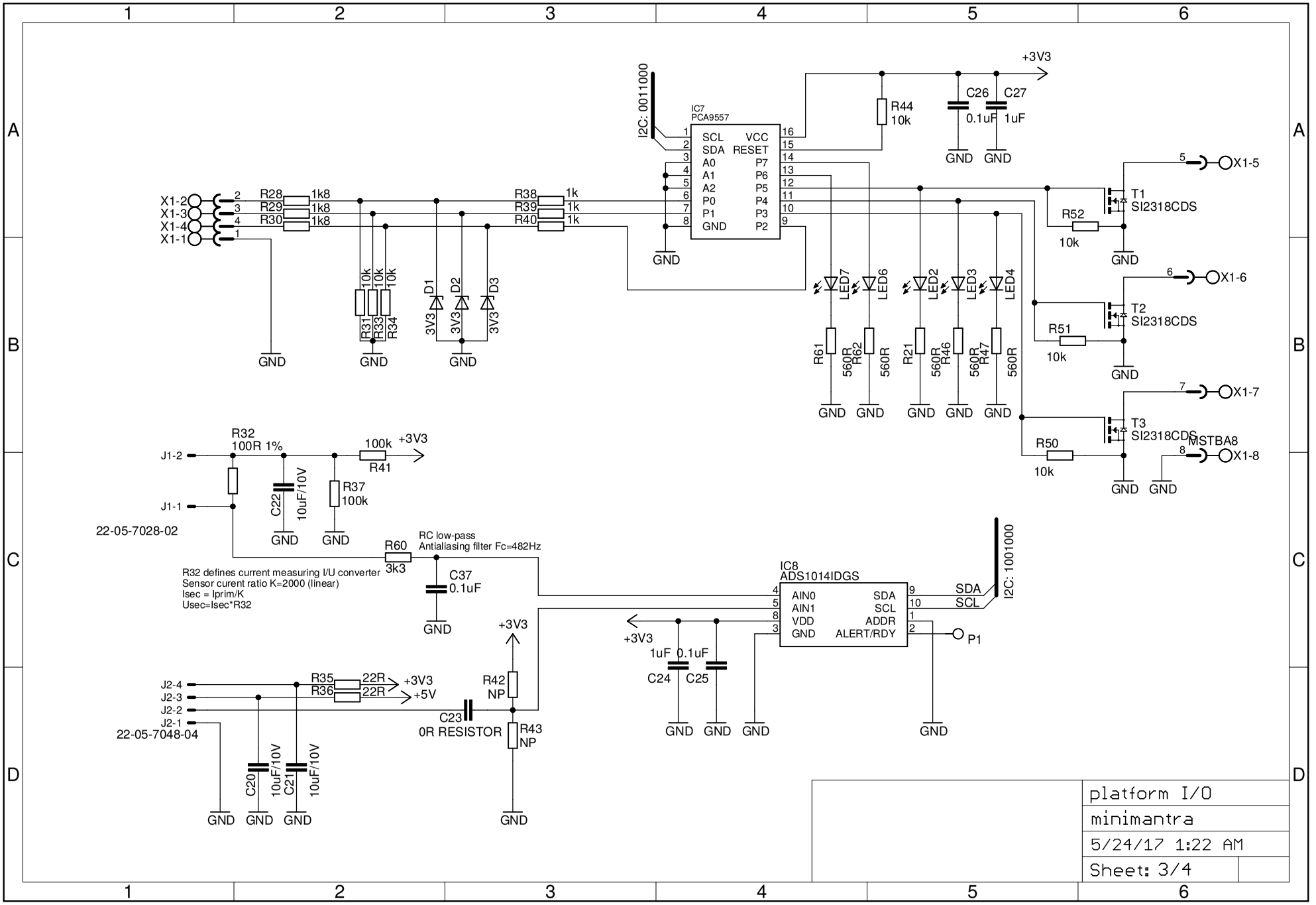
Schéma zapojení

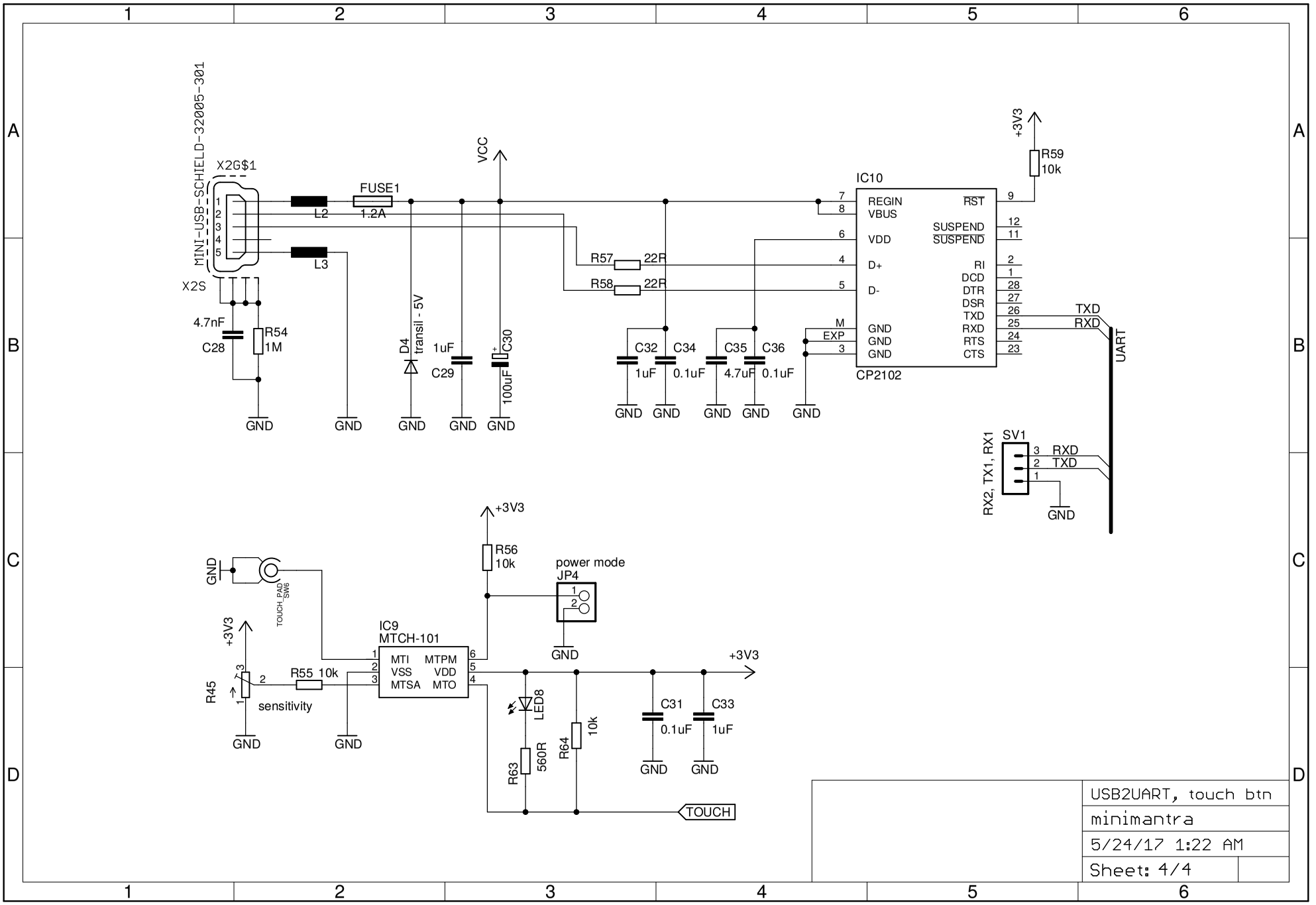


	power supply
	minimantra
	5/24/17 1:22 AM
	Sheet: 1/4



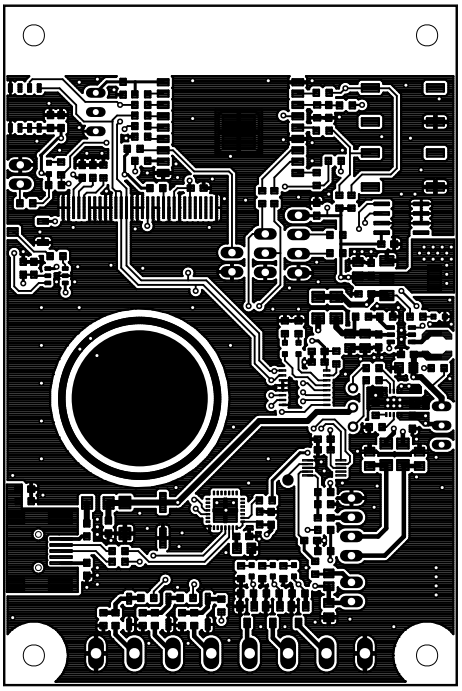
	esp8266, OLED, temp
	minimantra
	5/24/17 1:22 AM
	Sheet: 2/4


1
2
3
4
5
6
A
B
C
D
I2C: 0011000
IC7
PCA9557
SCL
SDA
A0
A1
A2
P0
P1
GND
VCC
RESET
P7
P6
P5
P4
P3
P2
+3V3
R44
10k
C26
0.1uF
C27
1uF
X1-2
X1-3
X1-4
X1-1
R28
R29
R30
1k8
R38
R39
R40
1k
R31
R33
R34
10k
D1
D2
D3
3V3
LED7
LED6
LED2
LED3
LED4
R61
R62
R21
R46
R47
560R
T1
T2
T3
SI2318CDS
R52
R51
R50
X1-5
X1-6
X1-7
X1-8
MSTBA8
J1-2
J1-1
22-05-7028-02
R32
100R 1%
R41
100k
C22
10uF/10V
R37
100k
R60
3k3
C37
0.1uF
RC low-pass
Antialiasing filter Fc=482Hz
R32 defines current measuring I/U converter
Sensor curent ratio K=2000 (linear)
Isec = Iprim/K
Usec=Isec*R32
IC8
ADS1014IDGS
AIN0
AIN1
VDD
GND
SDA
SCL
ADDR
ALERT/RDY
I2C: 1001000
P1
C24
C25
1uF
0.1uF
J2-4
J2-3
J2-2
J2-1
22-05-7048-04
R35
R36
22R
+5V
C20
C21
R42
NP
C23
0R RESISTOR
R43
platform I/O
minimantra
5/24/17 1:22 AM
Sheet: 3/4

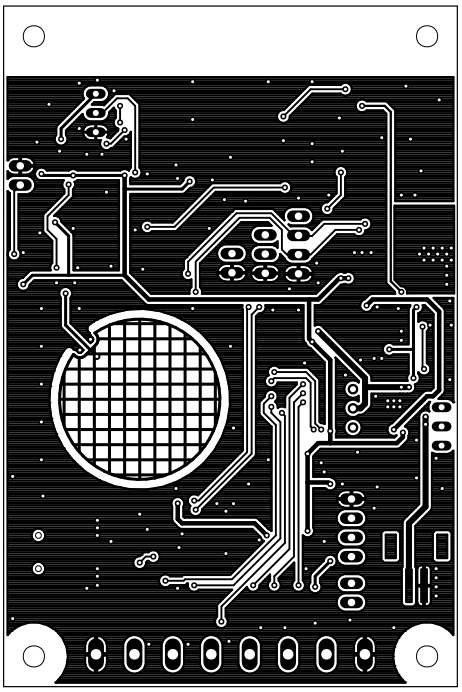

MINI-USB-SCHIELD-32005-301
X2G$1
X2S
L2
L3
FUSE1
1.2A
VCC
4.7nF
C28
R54
1M
D4
transil - 5V
1uF
C29
100uF
C30
R57
22R
R58
22R
IC10
REGIN
VBUS
VDD
D+
D-
GND
GND
GND
RST
SUSPEND
SUSPEND
RI
DCD
DTR
DSR
TXD
RXD
RTS
CTS
CP2102
+3V3
R59
10k
C32
1uF
C34
0.1uF
C35
4.7uF
C36
0.1uF
TXD
RXD
UART
SV1
RX2, TX1, RX1
RXD
TXD
GND
TOUCH_PAD
SW6
+3V3
R56
10k
power mode
JP4
IC9
MTCH-101
MTI
VSS
MTSA
MTPM
VDD
MTO
R45
sensitivity
R55 10k
LED8
R63
560R
R64
10k
C31
0.1uF
C33
1uF
TOUCH
USB2UART, touch btn
minimantra
5/24/17 1:22 AM
Sheet: 4/4

Příloha C

Výsledná DPS



Obrázek C.1: Horní vrstva DPS



Obrázek C.2: Spodní vrstva DPS

Příloha D

Kusovník

Hodnota	**Součástka**	**Označení**	**Popis**
	LEDCHIP-LED0603	LED2, LED3, LED4, LED5, LED6, LED7, LED8	LED
	MA03-1	SV1	PIN HEADER
	PINHD-1X2	JP3, JP4	PIN HEADER
	PINHD-1X4	JP2	PIN HEADER
	SOLD_PADR1.5	P1	SOLDPAD
	TRIM_EU-3223J	R45	POTENTIOMETER
0.1uF	C-EUC0603	C3, C4, C8, C9, C10, C17, C19, C25, C26, C31, C34, C36, C37	CAPACITOR, European symbol
0R	R-EU_R1206	R49	RESISTOR, European symbol
0R RESISTOR	C-EUC0603	C23	CAPACITOR, European symbol
1.2A	FUSE-1206	FUSE1	Fuse
100R 1%	R-EU_R0805	R32	RESISTOR, European symbol
100k	R-EU_R0603	R37, R41	RESISTOR, European symbol
100uF	CPOL-EUUD-6,3X8,8	C30	POLARIZED CAPACITOR, European symbol
10K	R-EU_R0603	R15	RESISTOR, European symbol
10k	R-EU_R0603	R6, R11, R12, R13, R14, R26, R27, R31, R33, R34, R44, R50, R51, R52, R55, R56, R59, R64, R65, R66	RESISTOR, European symbol
10uF/10V	C-EUC0805	C5, C6, C11, C20, C21, C22	CAPACITOR, European symbol
18k	R-EU_R0603	R4	RESISTOR, European symbol
1K	R-EU_R0603	R5	RESISTOR, European symbol

Tabulka D.1: Výčet použitých součástek 1/3

1M	R-EU_R0603	R54	RESISTOR, European symbol
1k	R-EU_R0603	R1, R2, R38, R39, R40, R48	RESISTOR, European symbol
1k8	R-EU_R0603	R28, R29, R30	RESISTOR, European symbol
1uF	C-EUC0603	C7, C12, C13, C14, C16, C18, C24, C27, C29, C32, C33	CAPACITOR, European symbol
2.2uF	C-EUC0603	C15	CAPACITOR, European symbol
22-05-7028-02	22-05-7028-02	J1	CONNECTOR
22-05-7048-04	22-05-7048-04	J2	CONNECTOR
22R	R-EU_R0603	R16, R17, R18, R20, R22, R23, R24, R25, R57, R58	RESISTOR, European symbol
22R	R-EU_R1206	R35, R36	RESISTOR, European symbol
24AA1026-I/SN	24AA1026-I/SN	IC5	MICROCHIP 24AA1026-I/SN EEPROM, 1 MB, 128K x 8bit, 400 kHz, Serial I2C, SOIC, 8 Pins
3V3	ZPD	D1, D2, D3	Z DIODE
3k3	R-EU_R0603	R60	RESISTOR, European symbol
4.7nF	C-EUC0603	C28	CAPACITOR, European symbol
4.7uF	C-EUC0805	C1, C2, C35	CAPACITOR, European symbol
4.7uH	L_LPS4018	L1	Výrobce: FERROCORE
4K7	R-EU_R0603	R9, R10	RESISTOR, European symbol
560R	R-EU_R0603	R21, R46, R47, R61, R62, R63	RESISTOR, European symbol
56k	R-EU_R0603	R3, R7, R8	RESISTOR, European symbol
64k	R-EU_R0603	R19	RESISTOR, European symbol
ADS1014IDGS	ADS1014IDGS	IC8	ADS101x Ultra-Small, Low-Power, I2C-Compatible, 3.3-kSPS, 12-Bit ADCs

Tabulka D.2: Výčet použitých součástek 2/3

Battery	M02-JST-2MM-SMT	JP1	Header 2
CP2102	CP2102	IC10	Single-Chip USB to UART Bridge
ESP8266 WROOM2	ESP8266 WROOM2	IC4	ESP8266 based WiFi module
LD1117S33	LD1117SXX	IC3	LDO stabilizer
LED RG SA 3MM	LED RG SA 3MM	LED1	
LM75AD	LM75AD	IC6	Digital temperature sensor and thermal watchdog
MCP1640	MCP1640X-Y/CHY	IC1	0.65V Start-up Synchronous Boost Regulator with True
MCP73833XXI/MF	MCP73833XXI/MF	IC2	Stand-Alone Linear Li-Ion / Li-Polymer Charge
MINI-USB-SCHIELD-32005-301	MINI-USB-SCHIELD-32005-301	X2	MINI USB-B Conector
MSTBA8	MSTBA8	X1	PHOENIX
MTCH-101	MTCH-101	IC9	MTCH101, capacitive touch
NP	R-EU_R0603	R42, R43	RESISTOR, European symbol
NP	R-EU_R1206	R53	RESISTOR, European symbol
OLED_13_SSD1306	OLED_13_SSD1306	LCD1	1.3"OLED Display
PCA9557	PCA9557	IC7	PCA9557 - Remote I2C 8-Bit I/O expander
SI2318CDS	SI2318CDS	T1, T2, T3	VISHAY SI2318CDS-T1-GE3 MOSFET Transistor, N Channel, 3.9 A, 40 V, 0.036 ohm, 10 V, 3 V
SW-RST	SW-MICROMICRO-SMD	SW4	
SW-USR	SW-MICROMICRO-SMD	SW3	
SW_P-ESP101	SW_P-ESP101	SW1, SW2	Switch ON-ON
TOUCH_PAD	TOUCH_PAD	SW6	
transil - 5V	DIODE-DO214AC	D4	DIODE

Tabulka D.3: Výčet použitých součástek 3/3