VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Petr Ondráček

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ÚSPORNÉ ZABEZPEČOVACÍ ZAŘÍZENÍ S MODULEM (ESPRESSIF) ESP8266/ESP32
LOW-ENERGY SECURITY DEVICE WITH (ESPRESSIF) ESP8266/ESP32 MODULE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Petr Ondráček

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. Vladislav Škorpil, CSc.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Petr Ondráček ***ID:*** 186153
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Úsporné zabezpečovací zařízení s modulem (Espressif) ESP8266/ESP32

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte koncepci jednoduchého detektoru otevření dveří/oken/prostoru pomocí magnetického kontaktu a WiFi modulů ESP32 nebo ESP8266. Detektor bude napájen bateriově a připojen do domácí WiFi sítě. Bude reportovat emailem i zápisem události do vhodné cloudové služby záznam o narušení prostoru pro případné podrobné zpětné prohlížení. Důraz při návrhu je přenositelnost zařízení a dlouhá provozní doba. Navržené řešení realizujte.

DOPORUČENÁ LITERATURA:

[1] MANN, B. C pro mikrokontroléry, BEN, 2003. ISBN 80-7300-077-6.

[2] KOSEK, J. PHP tvorba interaktivních internetových aplikací, Grada Publishing 1999, 492 stran, ISBN 80-716--373-1.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 29.5.2018

Vedoucí práce: doc. Ing. Vladislav Škorpil, CSc.
Konzultant: Ing. Ondrej Pavelka, Honeywell

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Bakalářská práce se zabývá návrhem a realizací úsporného zabezpečovacího zařízení pomocí Wifi modulu ESP32. Zařízení bude detekovat otevření dveří/oken/prostoru pomocí magnetického kontaktu a tuto událost nahlašovat na vhodnou cloudovou službu a e-mail. Teoretická část je zaměřena na problematiku Internetu Věcí, cloudové služby a popis jednotlivých komponent zařízení. Druhá část práce se zabývá návrhem schéma zapojení, řídícího programu a výpočtu spotřeby zařízení. Třetí část je již zaměřena na praktickou realizaci navrženého zařízení, návrh a tvorbu desky plošných spojů, ochranné krabičky pro zařízení, programování, vytvoření cloudového kanálu pro vizualizaci nasbíraných dat a měření spotřeby konečného zařízení.

KLÍČOVÁ SLOVA

Wifi modul, ESP32, magnetický kontakt, cloudová služba, Internet Věcí, deska plošných spojů

ABSTRACT

Bachelor's thesis discusses the design and realization of low-energy security device using Wifi module ESP32. The device will detect the opening of door/window/space by using magnetic contact and this event will be reported on appropriate cloud service and by e-mail. The theoretical part is focused on the Internet of Things problematics, cloud services and description of individual components of the final device. Second part of thesis is focused on designing of the linkage scheme, control program and calculation of power consumption of the device. Third part is already focused on practical realization of the designed device, designing and creation of printed circuit board and security box, programming, creation of the cloud channel that will visualize collected data and measurement of power consumption of the final device.

KEYWORDS

Wifi module, ESP32, magnetic contact, cloud service, Internet of Things, printed circuit board

ONDRÁČEK, Petr. *Úsporné zabezpečovací zařízení s modulem (Espressif) ESP8266/ESP32*. Brno, 2017, 67 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Vladislav Škorpil, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Úsporné zabezpečovací zařízení s modulem (Espressif) ESP8266/ESP32“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu doc. Ing. Vladislavu Škorpilovi, Ph.D. a konzultantovi Ing. Ondřeji Pavelkovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VÝPISŮ

ÚVOD

Cílem této práce je vytvořit jednoduchý detektor otevření dveří, oken či libovolného prostoru pomocí magnetického kontaktu a Wifi modulu ESP32. Zařízení bude napájeno bateriově a navrženo tak aby mělo dlouhou provozní dobu (tj. minimální spotřebu elektrické energie). Výsledné zařízení se bude schopno připojit do domácí sítě po zadání základních parametrů sítě přes sériovou linku. V běžném provozu bude zařízení pravidelně e-mailem informovat o stavu napětí baterie a teploty. Dále bude e-mailem hlášeno narušení prostoru (rozepnutí/sepnutí magnetického kontaktu) a pokus o otevření krabičky s modulem (pomocí mikrospínače). Všechny zmíněné informace o provozu budou také zaznamenávány na vybranou cloudovou službu.

Práce je psána postupně od teoretických základů, popisu jednotlivých komponent, návrhu zapojení, řídícího programu a způsobu měření spotřeby. Následuje praktická část ve které je popsán samotný proces výroby celého zařízení, jeho programování a připojení k vybraným cloudovým službám. Na závěr práce je pomocí měření ověřen úsporný design zařízení.

1 INTERNET OF THINGS

Internet věcí (Internet of Things, zkráceně IoT) je pojem označující síť nejrůznějších zařízení připojených k internetu. Tyto zařízení (někdy označované jako „věci“ v rámci IoT) nemusí představovat pouze fyzický objekt obsahující elektroniku, senzory a jiné, ale můžeme do této skupiny zařadit i objekty virtuální. Hlavní charakteristika takového objektu, ať už fyzického nebo virtuálního, je schopnost samostatně poskytovat data a sdílet je s dalšími věcmi nebo systémy přes síť Internet. [1]

1.1 Důvod použití IoT

Hlavním cílem IoT je propojení více zařízení a tím i získávání více užitečných dat (informací), které můžeme následně analyzovat a dále využít. Čím více informací budeme mít k dispozici, tím více toho o daném systému můžeme zjistit a tyto znalosti můžeme využít například pro zjednodušení každodenního života v domácnostech či zefektivnění využívaných zdrojů v průmyslu a tím dosažení značných úspor. Tato možnost efektivity a úspory je jeden z hlavních lákadel pro investování do IoT pro malé i velké firmy.

1.2 Požadavky na IoT

Základním požadavkem pro funkčnost IoT zařízení je připojení k internetu. V dnešní době se může zdát, že tento požadavek není již žádný problém, ale i dnes se v České republice najdou oblasti kde je úroveň připojení k internetu velmi nízká.

Jedním z důležitých požadavků je jednoznačná adresace zařízení a s tím je i spojen problém nedostačujícího adresového prostoru. Tento problém je podrobněji rozebrán v podkapitole „IP adresace a nedostatek adresního prostoru“.

Bezpečnost je v dnešní době neopomenutelná záležitost a IoT zařízení nejsou výjimkou. IoT systém musí podporovat bezpečnostní funkce koncových uzlů i přenosu a sdílení dat, šifrování komunikace a řadu dalších.

Vzájemná kompatibilita je dalším požadavkem pro funkční IoT zařízení. V této oblasti dochází k problémům, neboť zařízení jsou často vytvářena různými dodavateli a používají tedy i různé datové modely a způsoby komunikace. V dnešní době existují dvě dominantní aliance (Open Interconnect Consortium sdružující firmy Intel, Samsung, Dell, Broadcom aj. a AllSeen Aliance sdružující firmy Cisco, LG,

Microsoft a další), které prosazují různé standardy jakými zařízení komunikují se síti i mezi sebou. [2]

Další požadavky byly již zmíněny v úvodu kapitoly. Je to schopnost zařízení umožnit sběr, ukládání, analýzu a sdílení dat přes internet.

1.2.1 IP adresace a nedostatek adresního prostoru

Každé zařízení, které je připojeno k internetu musí mít přiřazenou jedinečnou veřejnou IP adresu. Pokud bychom ale museli přiřadit každému jednotlivému zařízení svoji veřejnou IP adresu, došlo by tak k rychlému vyčerpání adresového prostoru. Proto vznikly tzv. lokální sítě, které sdružují více zařízení pod jednu veřejnou IP adresu, což umožnilo připojení ještě více zařízení k internetu. Adresový prostor IPv4 poskytující přibližně 4 miliardy jedinečných 32-bitových adres byl i přes to vyčerpán na začátku roku 2011. Naštěstí tento problém byl očekáván, a proto byla vytvořena další verze IP protokolu IPv6, která již používá 128-bitové adresy a je tedy schopna poskytnout $3{,}4 \times 10^{38}$ unikátních adres. Tento protokol ale starší verzi nenahradil ihned. V dnešní době je využívána jak dosluhující IPv4 tak IPv6, a proto by mělo i každé IoT zařízení oba protokoly podporovat. [3]

Tím byl prozatím na dostatečnou dobu problém s nedostatkem adres vyřešen. Ovšem z uživatelského hlediska není připojování ke každému IoT zařízení zvlášť podle jeho IP adresy ideální. Proto byly vytvořeny tzv. centrální stanice. Tato stanice má jediná svoji přiřazenou IP adresu a veškerá zařízení jsou k ní připojena pomocí kabelu, sběrnice nebo bezdrátově. Při použití kabelu je možné použít komunikační protokoly jako jsou RS232 či RS485, pro sběrnice například I2C, SPI, apod. Při použití bezdrátového připojení je možno komunikovat s centrální stanicí například pomocí technologie WiFi, Bluetooth, atd.

1.3 Využití

Internet věcí se díky své široké využitelnosti stal součástí mnoha oborů lidské činnosti. Příkladem může být například tzv. „chytrá domácnost“, která ulehčuje uživateli běžný život například automatickou regulací teploty pomocí topení a klimatizace, ovládání osvětlení pomocí kombinace světel a žaluzií, zabezpečení domácnosti pomocí detektorů, řízených zámků, kamerového systému, alarmu ... Dále může být domácnost vybavena například lednicí, která je schopna hlásit nedostatek jednotlivých potravin nebo varná konvice, která automaticky ráno v daný čas uvaří vodu. Toto bylo jen několik příkladů z mnoha co může taková chytrá domácnost uživateli poskytnout.

Další významná oblast využití IoT je Průmyslový Internet Věcí (IIoT - Industrial Internet of Things). Tato oblast zahrnuje především průmyslovou automatizaci, energetický průmysl a dopravní průmysl. Systémy IoT jsou zde využívány pro efektivnější využívání zdrojů, minimalizaci provozních nákladů, předcházení selhání a haváriím pomocí monitorování a včasné údržby, což má za důsledek i zvýšení celkové bezpečnosti.

Internet věcí má své využití i v zemědělství a chovu zvířat. Je využíván například pro měření vlhkosti a PH půdy. Díky těmto informacím mohou zemědělci ušetřit náklady za zavlažování a hnojení. Další využití může být hlídání zemědělských strojů, objektů, zvířat... Zajímavostí je systém pro hlídání a sledování zdravotního stavu skotu nazývaný „Internet of Cows“.

Zajímavé využití IoT jsou tzv. „wearables“ (v doslovném překladu „nositelnosti“). Jedná se o zařízení navržené jako součást lidského oděvu. Příkladem jsou chytré hodinky, brýle s monitorem, fitness náramky nebo ve zdravotnictví to jsou senzory monitorující zdravotní stav člověka (získávání údajů jako je krevní tlak, srdeční tep, teplota, délka spánku nebo i hladina glukózy v krvi). Lékař je díky těmto informacím schopen přizpůsobovat léčbu a sledovat vývoj nemoci na dálku.

Využití Internetu věcí je samozřejmě více a každým dnem vznikají další, výše uvedené příklady jsou dnes ty nejpoužívanější a nejrozšířenější.[4]

1.4 Perspektiva

Počet zapojených IoT zařízení má bezpochybně rostoucí tendenci. Mezi roky 2008 a 2009 celkový počet zařízení připojených k internetu překročil počet tehdejší světové populace, což činilo přibližně 6,7 miliardy. V roce 2015 to již činilo 25 miliard zařízení. A předpoklad pro rok 2020 je téměř 50 miliard zařízení připojených k internetu. Z toho budou tvořit přibližně polovinu IoT zařízení (tj. bez počítačů, chytrých telefonu, tabletu aj.). Internet věcí se za velmi krátkou dobu stal velkou součástí téměř všech oborů lidské činnosti. V České republice je předpokládán největší rozvoj v oblasti monitorování dopravy, provozu průmyslové výroby či monitorování a zabezpečení domácností. Internet věcí má tedy u nás i ve světě obrovský potenciál a nepředpokládá se, že by se tento trend měl v budoucnosti měnit.

2 POUŽITÝ HARDWARE

V této kapitole budou podrobněji popsány jednotlivé komponenty, které budou použity pro realizaci výsledného zařízení.

2.1 ESP32

Hlavní součást zařízení bude Wifi modul ESP32. Jedná se SoC (system on chip) mikrokontrolér s integrovanou Wifi a Bluetooth technologií. Modul byl vyvinut firmou Espressif Systems z Šanghaje a na trh byl uveden začátkem září 2016. Jedná se o nástupce velmi úspěšného o dva roky staršího Wifi čipu ESP8266 se spoustou vylepšení a nových funkcí.

2.1.1 Základní parametry modulu

Modul ESP32 je velmi propracované zařízení disponující mnoha parametry a funkcemi. V této kapitole budou zmíněny pouze ty nejpodstatnější vlastnosti, které je potřeba znát pro základní porozumění možností, které modul ESP32 uživateli poskytuje. [5, 6, 7]

Klíčové parametry zařízení jsou:

- 240MHz dvoujádrovým mikrokontrolérem Tensilica LX6.
- Integrovaná 520 KB SRAM
- 16 MB flash paměť namapovaná do CPU kódového prostoru.
- Integrovaný Wifi vysílač a přijímač podporující standardy IEEE 802.11 b/g/n/e/i v 2,4 GHz pásmu.
- Možnost zabezpečení Wifi provozu pomocí WEP,WPA/WPA2 PSK/Enterprise
- Integrovaný dvoumódový Bluetooth (Classic i BLE)
- Zabudovaná PCB anténa i IPEX konektor umožňující připojení externí antény
- Pracovní napětí 2,3 V až 3,6 V
- Pracovní teplota -40°C až +125°C
- 34 GPIO pinů (Podporující nejrůznější rozhraní jako UART, SPI, I2C. Dále jsou některé piny vybaveny A/D a D/A převodníky.)
- Pokročilý Low-power Management (správa nízké spotřeby) umožňující přepínání mezi pěti různými módy spotřeby a tím výrazně zvýšit dobu provozu zařízení napájené pouze z vlastní baterie.

2.1.2 Ovládání a programování modulu

Běžný způsob ovládání modulu umožňuje ovládat pouze jeho Wifi části s původním firmwarem modulu ESP32 je pomocí tzv. AT příkazů přes sériovou linku. Další možnost ovládání modulu je použití jiného firmwaru, například NodeMCU, který umožňuje krom ovládání Wifi části i vytvoření vlastního řídícího programu a ovládání GPIO pinů (volně použitelné vstupně výstupní piny). Pro účely projektu bude použit právě firmware NodeMCU.

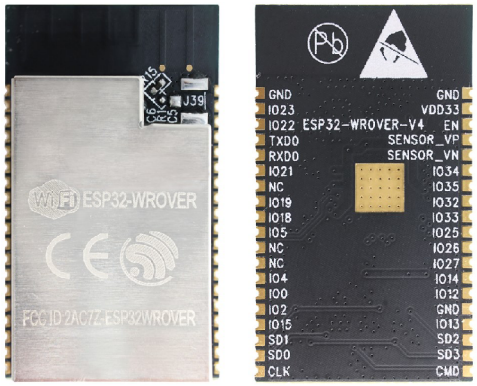
2.1.3 Typy modulů

Firma Espressif pro různé potřeby uživatelů navrhla několik typů modulů jako jsou:

- ESP-WROOM-32 je nejmenší modul určený přímo pro konečné produkty a je použit i v této bakalářské práci.

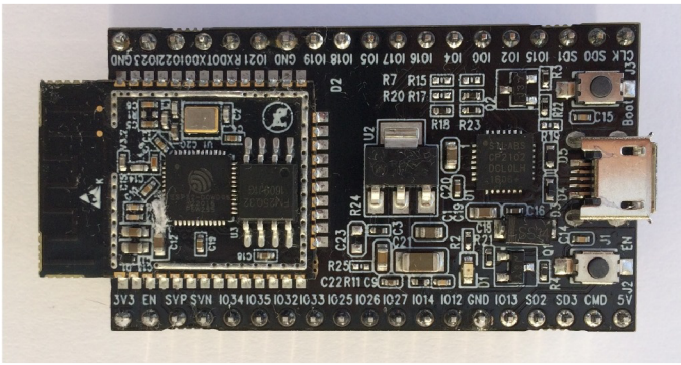
Obr. 2.1: ESP-WROOM-32 [8]

- ESP-WROVER je vylepšený ESP-WROOM32, vybavený navíc 4MB pseudo statickou RAM (PSRAM). Modul je k dostání ve dvou verzích. První s vestavěnou PCB anténou a druhý s IPEX anténou.



Obr. 2.2: ESP-WROVER [8]

- ESP32/DevKitC je vývojový modul s již zabudovaným USB rozhraním pro programování přes sériovou linku a zároveň poskytující napájení celého modulu.



Obr. 2.3: ESP32/DevKitC [8]

- a jiné přepracované verze development kitů využívajících ESP-WROOM-32 nebo ESP-WROVER [8]

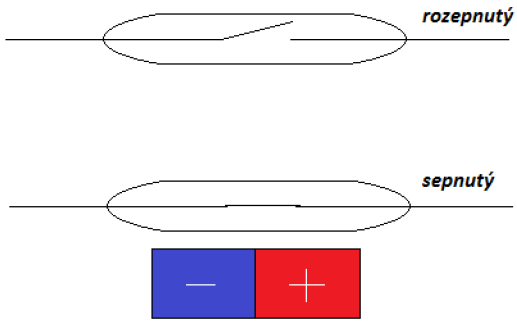
2.2 Magnetický kontakt

V projektu bude použit běžný magnetický jazýčkový kontakt typu NC (normally closed). Většina magnetických kontaktů vydrží proudy v řádu ampér, což pro potřeby projektu není žádným způsobem omezující. V následující podkapitole bude popsán samotný princip funkce magnetického kontaktu.

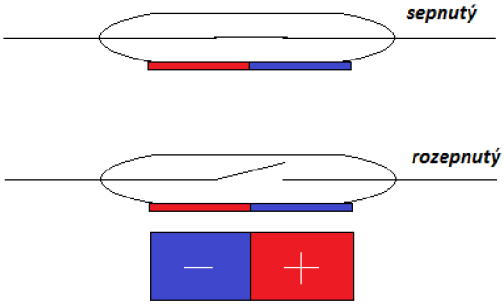
2.2.1 Princip funkce

Magnetický kontakt (často označován jako jazýčkový kontakt) je v principu mechanický spínač ovládaný magnetickým polem. Spínač je tvořen dvěma jazýčky, které jsou vyrobeny z feromagnetického materiálu a jsou uloženy v utěsněné skleněné trubici vyplněné inertním plynem. Začne-li na spínač působit magnetické pole, například při přiložení permanentního magnetu nebo cívky (kterou prochází proud), feromagnetické jazýčky se zmagnetizují a pokud je magnetické pole dostatečně silné, dojde k jejich ohnutí, vzájemnému doteku a tím dojde k sepnutí kontaktu. Přestane-li působit magnetické pole, dojde k opětovnému rozepnutí kontaktu. Tento princip platí pro tzv. „normally open (NO)“ typ kontaktu a je zachycen na obr. 2.4. Druhý typ kontaktu „normally closed (NC)“ (který je použit v projektu) funguje opačným způsobem. Bez přiloženého magnetu je kontakt sepnutý a přiložením dojde k jeho rozepnutí. Tohoto lze docílit permanentním připevněním magnetu jak v případě NO

kontaktu. Přikládaný magnet pak musí mít opačnou polaritu než připevněný magnet. Jeho přiložením se magnetické pole vyruší a kontakt se rozepne viz obr. 2.5. [9]



Obr. 2.4: NO magnetický kontakt



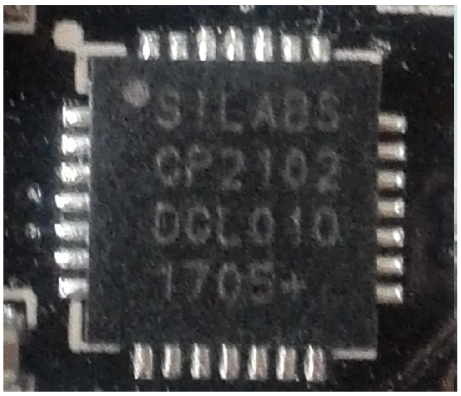
Obr. 2.5: NC magnetický kontakt

2.3 Baterie

Jako zdroj napájení pro celé zařízení slouží LiFePo4 akumulátor poskytující 3,2 V napětí. Napájení čipu ESP32 se může pohybovat v intervalu od 2,3 do 3,6 V, což LiFePo4 akumulátor splňuje (včetně stavu, kdy je akumulátor plně nabitý a jeho napětí může dosáhnou až 3,6 V). Nebude tedy potřeba zahrnovat do projektu regulátor napětí. Konkrétně bude použita baterie ANR26650 od firmy A123 Systems, která při plném nabití poskytuje kapacitu 2500 mAh.

2.4 Převodník USB/UART

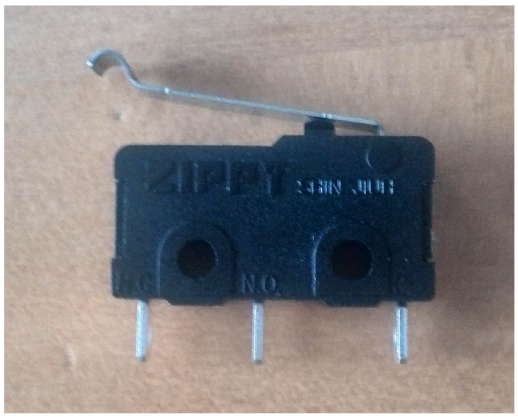
Pro potřeby debugování, nahrávání firmwaru a všeobecně jakékoliv komunikace po sériové lince mezi zařízením a počítačem pomocí USB je nutné použít využít převodník USB/UART, který zaručuje kompatibilitu mezi těmito protokoly. V projektu bude použit převodník CP2102, který je běžně používán ve vývojových deskách ESP32-Devkit. Převodník pracuje rychlostí až 1 Mbit/s. Převodník nebude součástí konečné desky plošných spojů, bude připojován uživatelem ručně.



Obr. 2.6: Převodník CP2102

2.5 PANIC kontakt - mikrospínač

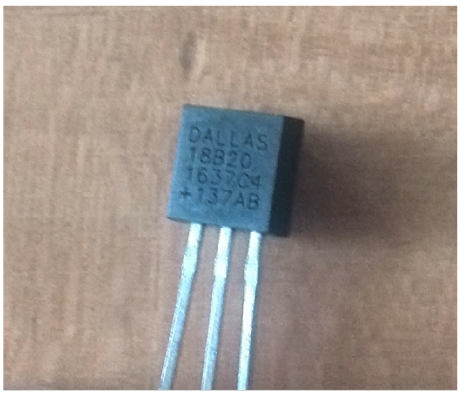
Řešení PANIC kontaktu bude v projektu realizováno mikrospínačem, který bude sepnut pokud bude krabička s modulem uzavřená. Pokud dojde k otevření/porušení krabičky a tím pádem i rozepnutí mikrospínače, modul bude náležitě reagovat, viz návrh programu.



Obr. 2.7: Mikrospínač

2.6 Teplotní senzor - DS18B20

Jako teplotní senzor v projektu bude použit DS18B20 od firmy Maxim. Senzor umožňuje měřit teplotu v rozmezí od -55 do +125°C. V rozmezí -10 až +85°C je garantována přesnost ± 0,5°C. DS18B20 umožňuje konfigurovatelné 9 - 12 bitové měření s využitím sběrnice Onewire. Každý senzor má svoje unikátní 64 bitové sériové číslo, což umožňuje použití více senzoru na jedné Onewire sběrnici. Potřebné napájení senzoru je udáváno v rozmezí od 3 do 5,5 V a maximální provozní proud je 1,5 mA. Těmto požadavkům GPIO piny Wifi modulu ESP32 vyhovují a senzor tedy může být v projektu použit. [10, 11]



Obr. 2.8: Teplotní senzor DS18B20

3 POUŽITÝ SOFTWARE

V této kapitole se podrobněji seznámíme se softwarem, který byl použit při realizaci projektu.

3.1 Vývojové prostředí PlatformIO

PlatformIO je open source vývojové prostředí, které je podporováno většinou populárních operačních systémů jako je Windows, Mac a Linux. PlatformIO podporuje více než 400 vývojových desek, 24 vývojových platform a 14 Frameworků. Program má velice dobře organizovaný systém knihoven, které mohou být jednoduše zahrnuty do daného projektu. PlatformIO bylo původně navrhnuto jako program využívající příkazový řádek. Později však začalo být používáno jako plugin pro jiné textové editory a vývojové prostředí, jako jsou například Eclipse, Visual Studio nebo Atom.io. V práci je vývojové prostředí použito pro vytvoření řídícího programu modulu ESP32. [12]

3.2 Editor Eagle

Eagle je editor plošných spojů nabízející tři moduly: Editor schémat, Editor spojů a Autorouter (pro automatický návrh plošného spoje). Tyto moduly jsou sloučeny do jednoho jednoduše ovladatelného uživatelského rozhraní. Při použití freeware verze je uživatel omezen maximální plochou desky 100 × 80 mm, možnost použití pouze dvou signálových vrstev spojů (vrchní a spodní strana desky) a editor schématu může vytvořit schéma pouze na jednom listu. Eagle je v práci použit pro návrh schéma zapojení a pro návrh desky plošných spojů (PCB). [13]

3.3 NodeMCU

NodeMCU je open-source firmware (operační systém) umožňující vytvořit vlastní řídící program a ovládat GPIO piny čipu ESP32. Pro použití NodeMCU je nutné přehrát původní firmware, který je v čipu nahrán od výrobce. Vytvořený program se nahrává přímo do čipu, díky čemuž je Wifi modul schopen samostatné funkce bez pomocného mikrokontroleru. Vývojové prostředí PlatformIO ulehčuje práci programátora, neboť nahrávání firmwaru NodeMCU provádí automaticky.

3.4 Cloud

Pojem Cloud je používán pro všeobecně jakékoliv poskytování pronajímaných služeb přes internet. Cloud se dá v zásadě rozdělit na tři různé modely, podle služeb které poskytují. [14]

- **Model SaaS** - Software-as-a-Service (Software jako služba) - tento model je pro běžného uživatele asi nejznámější, jedná se o tzv. "software na vyžádání". Poskytovatel poskytuje software ve formě stáhnutelného výkonného kódu, který volá přes internet platformově nezávislé programové rozhraní (API - Application Programming Interface), které se nachází na serverech poskytovatele služby. Toto programové rozhraní je obvykle realizováno pomocí protokolu HTTP (Hypertext Transfer Protocol). Příklady SaaS služeb jsou například GMail, Facebook, Google Docs, Dropbox, Spotify, Youtube a mnoho dalších.

- **Model PaaS** - Platform-as-a-Service (Platforma jako služba) - model určen vývojářům pro vývoj a běh aplikací. PaaS poskytuje programovací jazyky a k nim standardní knihovny a frameworky. Příklady PaaS služeb jsou například Google App Engine, Microsoft Azure Websites, Red Hat OpenShift a jiné.

- **Model IaaS** - Infrastructure-as-a-Service (Infrastruktura jako služba) - model poskytující tzv. virtuální stroje, které umožňují běh celých operačních systémů. Model IaaS také poskytuje obrazy disků, síťové služby (VLAN,VPN,firewall) a ukládání dat do virtuálních uložišť. Příklady IaaS služeb jsou například Microsoft Azure, Google Compute Engine, Amazon Elastic Compute Cloud a jiné.

V projektu bude cloudová služba sloužit k ukládání a vizualizaci odeslaných dat modulem ESP32 (tj. stav baterie, teplota, upozornění o narušení - rozpojení magnetického nebo PANIC kontaktu).

V následujicích podkapitolách jsou stručně uvedeny vybrané cloudové služby, které vyhovovaly požadavkům a byly v rámci práce použity. Podrobnější popis implementace a jejich vzájemné srovnání je potom součástí kapitoly 8 zaměřené na programování zařízení.

3.4.1 ThingSpeak

Thingspeak je open source služba podporující REST API (rozhraní orientované na práci s daty) pro IoT. Umožňuje pomocí HTTP protokolu přijímat a odesílat data z nejrůznějších IoT zařízení včetně ESP32. Odeslaná data lze zobrazit přímo na internetových stránkách ThingSpeaku pomocí jednoduchých editovatelných grafů, dále je možné provést analýzu přijatých dat pomocí funkcí programu MATLAB (bez nutnosti zakoupení oficiální licence). Nashromážděná data lze ze stránek také exportovat v podobě souborů typu JSON, XML nebo CSV. Neplacená verze Thingspeaku je oproti placené různě omezená. Implementace služby a srovnání s ostatními cloudy, které byly v práci použity je popsáno v kapitole 8 .[15]

3.4.2 Thinger.io

Thinger.io je další open source služba umožňující připojení IoT zařízení k internetu. Služba opět využívá API a protokol HTTP pro komunikaci se zařízením. Thinger.io umožňuje nejen ukládání dat ale i oboustrannou komunikaci v realném čase se zařízením, což umožňuje větší škálu aplikací. Pro potřeby této práce je důležité, že služba umožňuje ukládání dat, jejich vizualizaci na webových stránkách a možnost exportu v různých formátech jako jsou například CSV, ARFF nebo JSON. I zde existuje placená a neplacená verze s různými možnostmi a omezeními. [16]

3.4.3 Ubidots

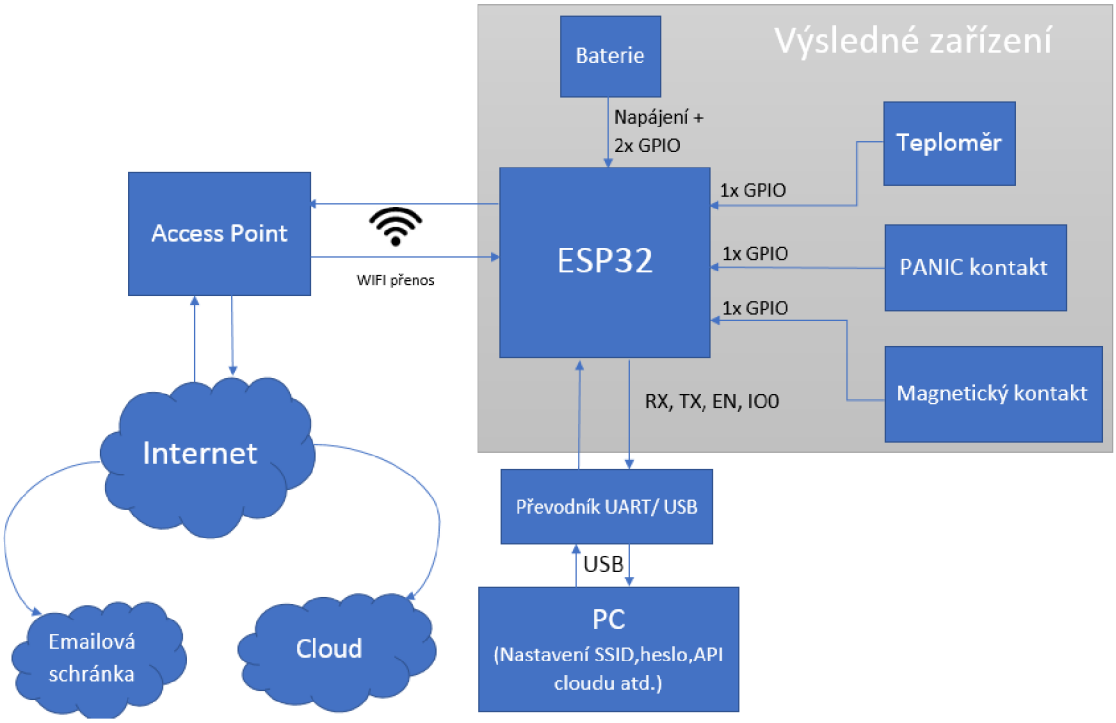
Ubidots je další použitá cloudová služba pro IoT zařízení. Využívá opět API a protokoly jako jsou HTTP/MQTT/TCP/UDP pro komunikaci. Ubidots umožňuje ukládání dat, jejich analýzu a různé zpracování, vizualizaci na webových stránkách a také možnost exportu dat ve formátu CSV. Podobně jako thinger.io umožňuje také další funkce jako je oboustranná komunikace v reálném čase a podobně. Tyto funkce ovšem nebudou v této práci využívány. Klasická verze Ubidots je k dostání pouze placená. V práci je využita verze Ubidots for Education, která má oproti plné placené verzi různá omezení. Implementace služby, její specifikace a porovnání s ostatními cloudy je uvedeno v kapitole 8. [17]

4 NÁVRH ZAPOJENÍ

Nyní jsme již seznámeni s veškerými komponenty a nástroji, které k realizaci projektu potřebujeme a můžeme začít se samotným návrhem zařízení.

4.1 Blokový diagram zapojení

Celková architektura zařízení je zachycena na obr.4.1. Hlavním řídícím blokem zařízení je již několikrát zmíněný Wifi modul ESP32. Modul zprostředková komunikaci či sběr dat ze všech periférií výsledného zařízení, což zahrnuje teplotní senzor, PANIC kontakt, magnetický kontakt a měření napětí baterie. Dále modul tyto informace vhodným způsobem zpracuje a pomocí bezdrátového připojení k místní síti (k Access pointu) tyto data distribuuje na internet do předdefinovaných cloudových služeb a e-mailové schránky. Součástí zadání bylo také zahrnutí možnosti připojení počítače k zařízení pomocí USB. Pro umožnění komunikace mezi USB sběrnicí a UART rozhraním modulu používajícím protokol RS-232 je zapotřebí například USB/UART převodník CP2102, který zajišťuje vzájemnou kompatibilitu.



Obr. 4.1: Blokový diagram zapojení

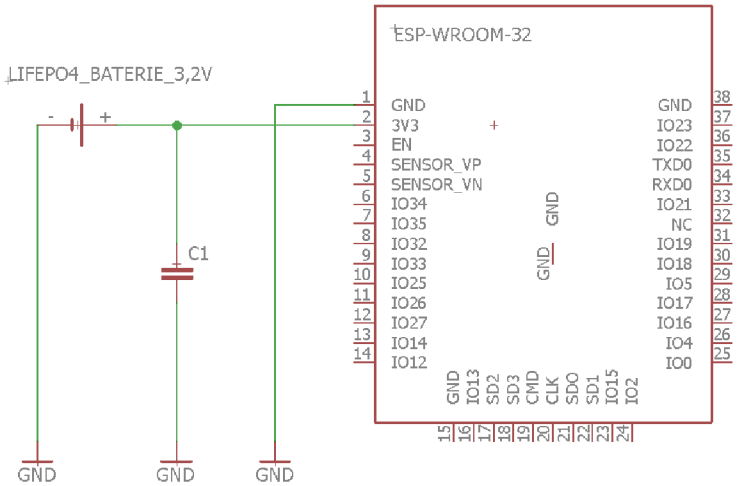
4.2 Schéma zapojení

Nyní se již můžeme pustit do samotného návrhu zapojení. Výstupem této podkapitoly bude schéma zapojení, které bude následně využito pro návrh desky plošných spojů PCB (Printed Circuit Board). V návrhu zapojení je kladen důraz především na minimalizaci spotřeby zařízení.

4.2.1 Modul a napájení

Pro čip ESP32 je doporučeno napájení 3,3 V, ovšem je schopen pracovat v rozmezí od 2,3 do 3,6 V. Proto byl zvolen napájecí akumulátor LiFePo4 schopný dodávat napětí 3,2 V. Toto napětí ovšem nemá po celou dobu své životnosti. Plně nabitý akumulátor může dosahovat až hodnoty 3,6 V což podle ESP32 datasheetu je modul schopen zvládnout. Není tedy zapotřebí zakomponovat do zapojení regulátor napětí, který by vedl k energetickým ztrátám (což je v souladu s požadavky projektu na minimalizaci spotřeby). [7]

Zapojení baterie k modulu je jednoduché, kladný pól akumulátoru je připojen k pinu „3V3“, který slouží jako přívod napájení pro celý čip. Záporný pól akumulátoru a pin „GND“ jsou vyvedeny na společnou zem. Modul ESP-WROOM-32 disponuje 4 piny „GND“, které jsou navzájem propojeny uvnitř čipu a není je tedy nutné zapojovat všechny. Dále je dobré připojit v co nejmenší vzdálenosti mezi napájecím a zemnícím pinem čipu kondenzátor pro zamezení většího poklesu napájecího napětí při náhlém zvýšení odebíraného proudu.

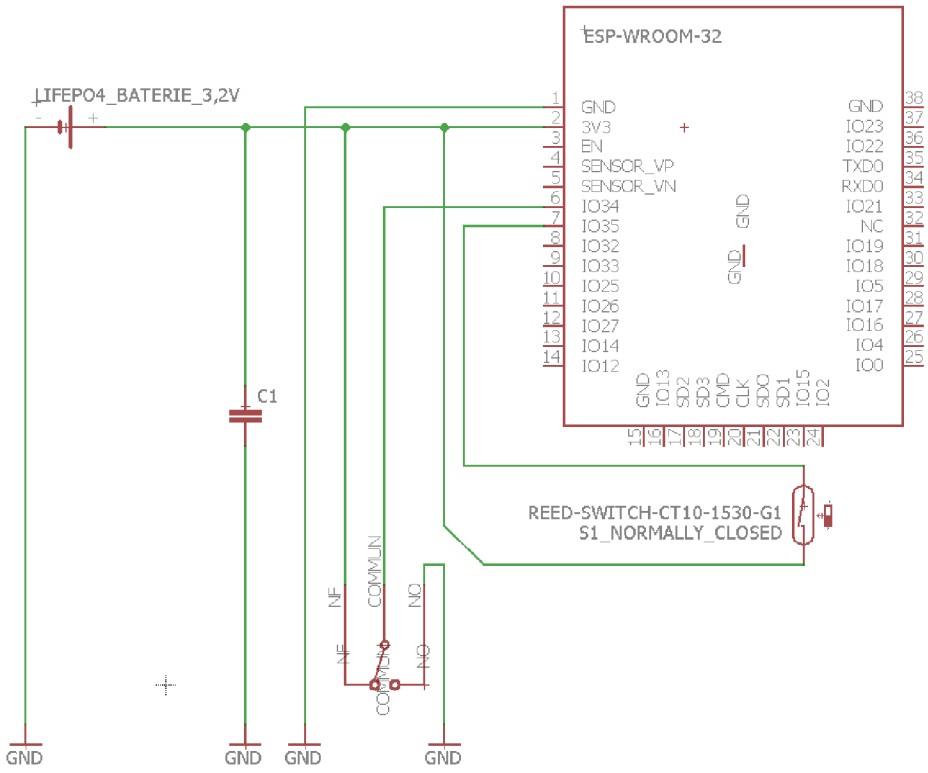


Obr. 4.2: Zapojení napájení k modulu ESP32

4.2.2 Magnetický kontakt a PANIC kontakt

V zapojení je použit magnetický kontakt typu NC (normally closed - normalně zavřený), který je za přítomnosti magnetu rozepnutý. Kontakt spojuje napájení (zde sloužící jako zdroj logické jedničky) a vhodný GPIO pin. Použitím typu kontaktu NC docílíme toho, že při klidovém stavu bude na GPIO pinu logická nula a při rozepnutí kontaktu (narušení) logická jednička. Magnetický kontakt bude sloužit k probouzení čipu z deep sleep módu (režim hlubokého spánku), je tedy nutné použít RTC GPIO pin, který je možné využívat i v režimu spánku (RTC - Real Time Clock periférie zůstávají v deep sleep módu aktivní s minimální spotřebou).

PANIC kontakt je realizován pomocí mikrospínače, který slouží k detekci otevření (narušení) krabičky, ve které se zařízení nachází. Kontakt opět spojuje napájení (zdroj logické jedničky)a vhodný RTC GPIO pin (samozřejmě jiný než, který bude použit pro magnetický kontakt aby byl čip schopen rozeznat důvod probuzení). Kontakt je tedy za klidového stavu (uzavřená krabička) mechanicky sepnutý stěnou krabičky. Je tedy aktivní NC pin mikrospínače, který spojuje GPIO pin se zemí (logická nula). Při narušení se mikrospínač rozepne a bude aktivní pin NO (normally open - normálně otevřen), čímž dojde k přivedení logické jedničky k GPIO pinu.

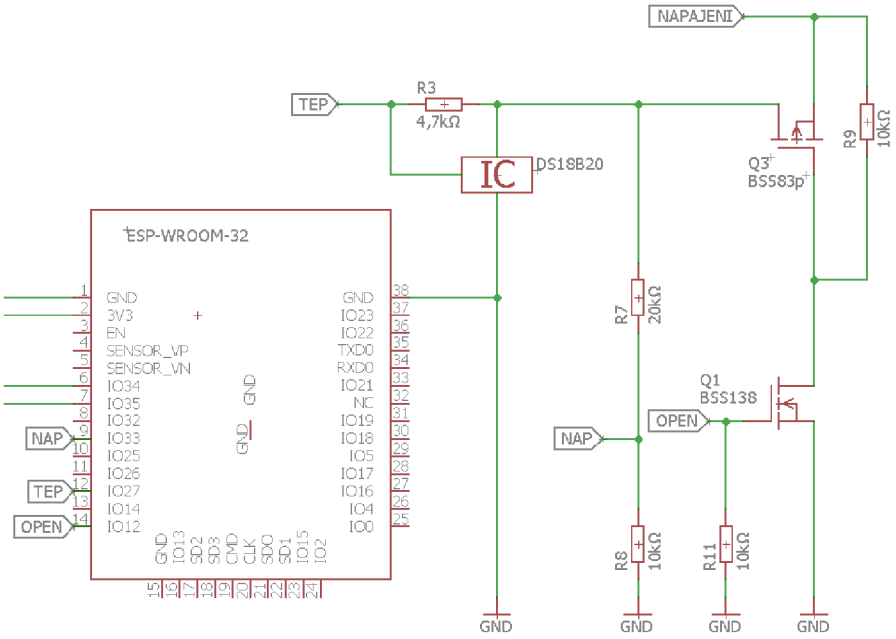


Obr. 4.3: Zapojení magnetického a PANIC kontaktu

4.2.3 Teplotní senzor a měření napětí baterie

Teploměr DS18B20 má vyvedeny 3 piny - přívod napájení, komunikační pin a zem. Pro minimalizaci spotřeby zařízení bude napětí přivedeno na napájecí pin senzoru pouze při samotném měření teploty. Tato funkce bude zajištěna pomoci kombinace N-channel a P-channel MOSFET tranzistorů (s P a N kanálem), které budou ovládat přívod napájení z baterie k napájecímu pinu senzoru na základě napěťové úrovně přivedené řídícím GPIO pinem. Zapojení je realizováno tak, že při přivedení logické nuly řídícím pinem na elektrodu GATE N-channel tranzistoru dojde k uzavření tohoto tranzistoru, neprochází tedy žádný proud (nebo jen minimální - např. gate-source leakage current) mezi DRAIN a SOURCE elektrodou a napětí na elektrodě GATE P-channel tranzistoru je rovno napájecímu napětí a proto je i tento tranzistor uzavřen. V okamžiku potřeby přivedení napájecího napětí k senzoru je řídícím pinem přivedena logická jednička, N-channel tranzistor se otevře, na GATE elektrodě P-channel tranzistoru se nyní nachází napětí menší než na elektrodě SOURCE, a dojde k jeho otevření a proto i přivedení napětí k napájecímu pinu teploměru. Komunikační pin teploměru bude přiveden na další GPIO pinu. Sběrnice Onewire, kterou senzor DS18B20 používá, vyžaduje aby komunikační linka byla v jasně definované logické hodnotě jedna (high). Toho v zapojení docílíme pomocí pull-up rezistoru mezi komunikační linkou a napájením. Master zařízení (v našem případě ESP32) je potom schopno žádat o data uvedením pinu do logické hodnoty low a stejně Slave zařízení (DS18B20) při poskytování dat. Zemnící pin senzoru je přiveden na společnou zem.

Měření napětí baterie bude realizováno pomocí vnitřního A/D převodníku čipu ESP32. Převodník A/D disponuje částečně přepínatelným rozsahem. Pro naše účely bude využit základní rozsah 1,1 V. Měřené napětí baterie bude tedy nutné přizpůsobit rozsahu převodníku pomocí vhodného odporového děliče. A/D převodník má nastavitelnou přesnost od 9 do 12 bitů, což odpovídá maximální hodnotě 511 pro 9 bit a 4095 pro 12 bit. Maximální hodnota představuje nejvyšší hodnotu rozsahu, v našem případě 1,1 V, což bude reprezentovat stav, kdy je baterie plně nabitá. Pro minimalizaci spotřeby bude opět napětí na dělič přivedeno pouze v okamžik měření napětí baterie. I pro tuto funkci využijeme zmiňované zapojení MOSFET tranzistorů, které ovládá přívod napětí k teplotnímu senzoru. Měření napětí baterie bude probíhat v době, kdy je komunikace po sběrnici Onewire neaktivní aby nedošlo ke zkreslení měřené hodnoty.



Obr. 4.4: Zapojení DS18B20 a měření napětí

4.2.4 Převodník UART/USB, reset pin EN a LED dioda

Pro možnost komunikace zařízení s počítačem pomocí USB portu je použit převodník UART/USB CP2102. Způsob zapojení převodníku k čipu ESP32 je převzat (kromě drobných přizpůsobení) ze zapojení vývojové desky ESP32 DevKitC, která využívá stejné komponenty pro komunikaci přes USB. Výhodou tohoto zapojení je možnost ovládání pinů boot(GPIO0) a reset(EN) pomocí signálů DTR a RTS což umožňuje automatické naprogramování čipu hned po překladu. Převodník je napájen pomocí externího napájení připojeného počítače přes USB (opět z důvodu minimalizace odběru z baterie). Převodník není součástí výsledného zařízení a bude pouze připojován uživatelem do příslušného konektoru piny DTR, RTS, RX, TX a GND (viz upravené schéma pro PCB).[8]

Pin označený jako EN slouží k resetování celého čipu. Čip je restartován ve stavu, kdy se na pinu EN nachází logicka nula (pin je uzemněný). Proto je nutné pro běžný provoz tento pin pomocí pull-up rezistoru držet do logické hodnoty high. Dále pro potřeby manuálního resetu je v zapojení zakomponováno tlačítko, které pin EN umožní uzemnit.

Posledním komponentem v zapojení je indikační LED dioda ovládaná opět dalším GPIO pinem. LED dioda bude sloužit k indikaci stavů čipu (například neúspěšný pokus o připojení k lokální Wifi síti)

4.2.5 Kompletní návrh zapojení

Celkové schéma zapojení se všemi připojenými komponenty je v příloze dokumentu na obr. A.1. Tímto je hardwarová část návrhu zařízení kompletní.

4.2.6 Upravené schéma pro desku plošných spojů

Na výsledné desce plošných spojů se nebudou nacházet všechny komponenty, a proto je zapotřebí udělat drobné úpravy v zapojení. Magnetický kontakt bude potřeba připojit ke dveřím či oknu a pro krabičku by nemuselo být v daném prostoru místo. Proto bude magnetický kontakt přiveden vodičem (2 žíly) ke krabičce a připojen k zařízení pomocí šroubovací svorkovnice pro PCB. Dále se nebude na desce nacházet RS232/USB převodník, ale bude pouze vyveden konektor s piny EN,IO0,TX,RX a GND. Upravené schéma je v příloze dokumentu na obr. A.2.

5 NÁVRH ŘÍDÍCÍHO PROGRAMU

Výstupem této kapitoly bude blokový diagram řídícího programu čipu ESP32, který bude zachycovat princip funkce celého zařízení. Konkrétní realizace kódu programu a jeho implementace je rozebrána v kapitole Programování zařízení.

5.1 Požadavky a priority pro návrh programu

Řídící program čipu ESP32 bude schopen připojit zařízení k lokální síti a vhodným způsobem zpracovat veškeré vstupní data, což v tomto případě představuje reakci na logickou jedničku přivedenou na dané GPIO piny při rozepnutí magnetického kontaktu nebo mikrospínače, zpracování teploty ze senzoru DS18B20 a naměřeného napětí (resp. číselnou reprezentaci naměřeného napětí získaného pomocí A/D převodníku). Tyto naměřené hodnoty program upraví do vhodného formátu a následovně je bude schopen odeslat na cloudovou službu a do e-mailové schránky. Dále je nutné zajistit možnost komunikace se zařízením a možnost nastavení různých parametru pomocí sériové linky (připojení pomocí USB). Prioritou celého projektu je minimalizace spotřeby zařízení. V programové části to znamená efektivní přepínání mezi režimem spánku a aktivním režimem.

5.2 Blokový diagram řídícího programu

V této kapitole bude podrobněji popsán blokový diagram, který je v příloze B. Tento diagram zachycuje posloupnost operací, které bude řídící program během svého běžného chodu provádět.

Po připojení napájení nebo restartu zařízení budou v RTC paměti vytvořeny proměnné, které si bude program pamatovat i v režimu spánku. Kdyby tyto proměnné byly ukládány do běžné paměti RAM, došlo by k jejich ztrátě pokaždé, když by byl čip uveden do režimu spánku. Některé konfigurační informace, které nebudou často měněny, budou ukládány přímo do flash paměti čipu aby nedošlo k jejich ztrátě v případě odpojení napájení zařízení. Tyto proměnné (parametry) ponesou informace o SSID (identifikátor bezdrátové sítě), heslo do bezdrátové sítě, interval probouzení, e-mailovou adresu a jiné. Po načtení veškerých konfiguračních informací program zjistí, z jakého důvodu bylo zařízení probuzeno a bude podle této informace dále pracovat. Zařízení je naprogramováno takovým způsobem, že k jeho probuzení z režimu spánku může dojít za tří různých podmínek.

1. Timer (časovač) - Za klidového stavu bude čip probouzen pouze pravidelných intervalech definovaných parametrem *interval probouzení*. Při této události dojde k měření napětí baterie a teploty. Z důvodu kolísání napětí a nepřesnosti měření bude hodnota napětí průměrována z několika předchozích měření. Poté se zařízení pokusí připojit k lokální síti a pokud bude připojení úspěšné, odešle naměřené hodnoty na cloudovou službu a zařízení se opět uspí.

2. Magnetický kontakt (GPIO34) - Při rozpojení kontaktu dojde k uvedení pinu GPIO4 do logické úrovně high a čip bude okamžitě probuzen z režimu spánku. Událost bude indikována pomocí led diody a po připojení do lokální sítě zaznamenána do cloudové služby a na e-mail uživatele. Poté bude opět zařízení uvedeno do režimu spánku.

3. PANIC kontakt (GPIO35) - Obdobně při rozpojení PANIC kontaktu (otevření krabičky) dojde ihned k probuzení čipu. Událost je indikována pomocí led diody a po připojení k lokální síti zaznamenána do cloudové služby a na e-mail uživatele. Ovšem otevření krabičky nemusí nutně znamenat pouze sabotáž zařízení. K otevření krabičky může dojít i v případě, kdy chce uživatel upravit nastavení zařízení přes sériovou linku. Proto po odeslání záznamu o narušení bude zařízení čekat na zadání hesla uživatelem. Pokud nebude po danou dobu přicházet žádný vstup, zařízení se automaticky uspí aby se zbytečně nevybíjelo. Při úspěšném přihlášení bude již uživatel schopen vybrat a upravovat jednotlivé parametry. Po dokončení všech úprav bude uživatel vyzván k uzavření krabičky a zařízení se opět uvede do režimu spánku.

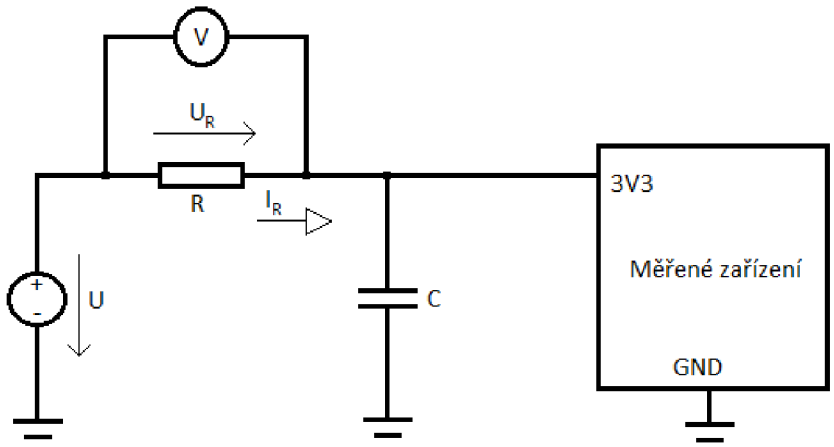
Před každým uspáním zařízení ještě dojde k uložení změněných konfiguračních dat do flash paměti a ke kontrole napětí baterie. Pokud bude menší než minimální povolená hodnota, bude o tom uživatel informován prostřednictvým e-mailu (jednou denně).

6 NÁVRH VÝPOČTU SPOTŘEBY ZAŘÍZENÍ

Navržené zařízení je napájeno bateriově, proto je zapotřebí zjistit jeho spotřebu pro určení maximální provozní doby bez dobíjení či výměny baterie. Bohužel, navržené zařízení nemá konstantní odběr proudu z důvodu střídání aktivního režimu a režimu spánku. Podle datasheetu ESP32 se odběr proudu zařízení v aktivním režimu pohybuje v rozmezí 100 - 200 mA. V režimu spánku je odběr proudu v rozmezí 2,5 - 100 μA. Měření skutečné spotřeby při malých proudech je problém. Při měření obyčejným multimetrem se hranice přesnosti a tolerance pohybuje v řádu mikroampér, což je pro účely práce nepřijatelné. Proto je zapotřebí zvolit jinou metodu měření. V následujících podkapitolách se seznámíme se dvěma možnými způsoby měření jejichž přesnost je pro účely práce dostačující.

6.1 Nepřímé měření - Měření náboje

Spotřebu zařízení lze měřit nepřímo například pomocí měření spotřebovaného náboje. Touto metodou nejsme schopni určit okamžitou spotřebu, ale to v běžné praxi ani není zapotřebí. Pro tento typ měření se používá velmi jednoduché zapojení pomocí rezistoru a kondenzátoru, které je zachyceno na obr.6.1. Měřené zařízení je napájeno z kondenzátoru s velkou kapacitou v řádu jednotek faradů. Tento kondenzátor je nabíjen ze zdroje napětí přes rezistor o velikosti například 1 kΩ. Zařízení si při špičkách (aktivní režim - větší spotřeba) odebírá proud z kondenzátoru, který se trvale přes rezistor dobíjí. Dále již stačí měřit napětí na zmíněném rezistoru a integrací tohoto napětí získáme náboj, který musel kondenzátor dobít při provozu, což odpovídá náboji, který zařízení spotřebovalo za dobu měření.



Obr. 6.1: Zapojení pro měření spotřeby

Pro odvození vzorce pro výpočet náboje použijeme Ohmův zákon pro proud procházející rezistorem

$$I_R = \frac{U_R}{R} \tag{6.1}$$

a definici proudu pomocí náboje

$$I_R = \frac{\mathrm{d}Q}{\mathrm{d}t}. \tag{6.2}$$

Pravé strany obou rovnic jsou si rovny

$$\frac{U_R}{R} = \frac{\mathrm{d}Q}{\mathrm{d}t} \tag{6.3}$$

a z rovnice vyjádříme náboj Q

$$\int_0^t \frac{U_R}{R}\,\mathrm{d}t = Q. \tag{6.4}$$

Průměrnou spotřebu získáme pomocí vzorce

$$\frac{Q}{t} = I_{\mathrm{prum}}. \tag{6.5}$$

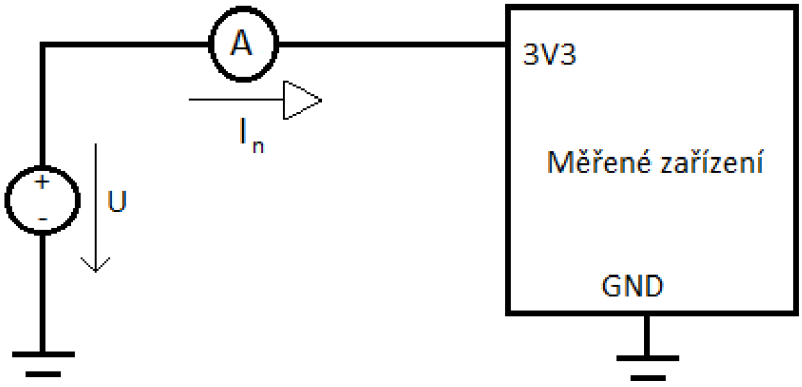
Pro určení teoretické maximální provozní doby (bez dobíjení či výměny baterie) dosadíme do vzorce vypočtený průměrný odběr proudu (I_{prum}) a kapacitu použité baterie (C_{bat}).

$$t_{max} = \frac{C_{bat}}{I_{\mathrm{prum}}} \tag{6.6}$$

Tato hodnota ovšem skutečnosti neodpovídá z několika důvodů. Jedním z nich je fakt, že u reálné baterie není možné využít celou kapacitu. Dále hraje roli samovolné vybíjení baterie. Měnící se výstupní napětí baterie a odběr proudu v závislosti na vybíjení baterie. Reálná provozní doba je z těchto důvodu menší.

6.2 Přímé měření - měření proudu

Při tomto typu měření je nutno použít přesnější stolní multimetr s vysokou třídou přesnosti. Měřící přístroj je při měření zapojen sériově mezi zdroj napětí a zařízení (zapojení ampérmetru). Zapojení je zachyceno na obr.6.2 Měřením je získávána okamžitá hodnota proudu, kterou právě zařízení odebírá. Tato hodnota není v čase konstantní a je proto pravidelně zaznamenávána (dále jsou hodnoty ukládány do vnitřní paměti a zobrazeny v grafu apod.). Interval mezi dvěma naměřenými hodnotami je tzv. vzorkovací interval (často je používán vzorkovací kmitočet což je



Obr. 6.2: Zapojení pro měření spotřeby

pouze převrácena hodnota vzorkovacího intervalu). Celkový spotřebovaný náboj Q za měřený interval je potom dán vztahem

$$Q = \sum_{n=1}^{P} I_n \cdot t_{vz}, \tag{6.7}$$

kde I_n je hodnota naměřeného proudu n-tého vzorku, t_{vz} je vzorkovací interval a P je celkový počet měřených vzorků (prakticky jde o integraci obdelníkovou metodou).

Průměrnou spotřebu získáme obdobně jako v předchozí metodě měření podle rovnice 6.5 s tím, že celkový čas je roven součinu počtu vzorků a vzorkovacího intervalu.

$$I_{\text{prum}} = \frac{Q}{t} = \frac{Q}{t_{vz} \cdot P} \tag{6.8}$$

Určení teoretické maximální provozní doby je totožné s předchozí metodou podle vzorce 6.6.

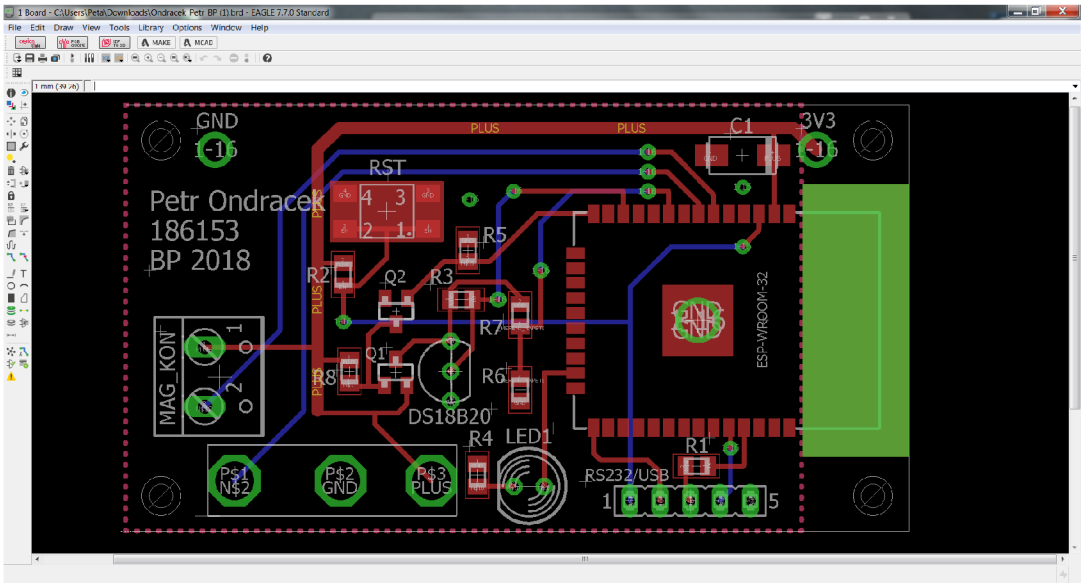
Touto metodou bylo měření spotřeby v práci realizováno (viz kapitola Měření spotřeby zařízení).

7 VÝROBA ZAŘÍZENÍ

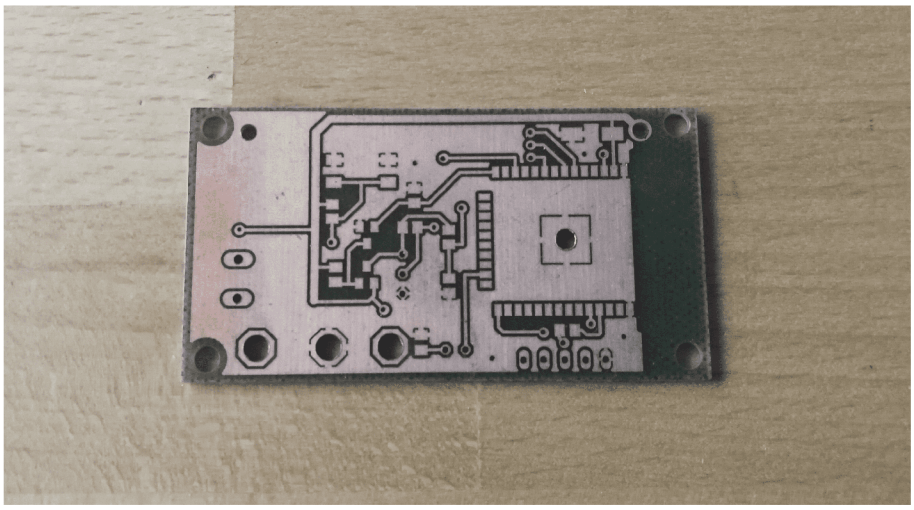
Nyní je již návrh zařízení kompletní a můžeme začít se samotnou výrobou zařízení.

7.1 Návrh a výroba desky plošných spojů

Prvním krokem je vytvoření desky plošných spojů na které budou všechny komponenty usazeny. V editoru Eagle lze jednoduše vytvořit z již vytvořeného schéma zapojení návrh desky plošných spojů s již předchystanými komponenty s naznačenými spoji (tzv. „airwires“), které korespondují se zapojením ve schématu. Při rozmísťování komponent je dobré umístit anténu čipu ESP32 tak, že v jejím okolo neprochází žádné další vodiče (nebo co nejméně) z důvodu možného rušení. Proto byla anténa umístěna na kraj desky. Dále by blokovací kondenzátor spojující napájecí a zemnící pin čipu měl být v co nejmenší vzdálenosti od samotného čipu. Všechny plošné spoje by měly být co nejkratší kvůli minimalizaci jejich odporu a jejich šířka by měla být úměrná velikosti proudu, který jimi prochází pro redukci případného ohřevu vodičů. Ostatní komponenty nemají žádné další požadavky pro jejich umístění a proto byly umístěny tak aby výsledné rozměry desky byly co nejmenší. Baterie nebude součástí desky plošných spojů a byly pro ni tedy pouze vyvedeny dva piny pro externí připojení. Baterie bude usazena společně s deskou v ochranné krabičce. Na okrajích desky byly umístěny 4 otvory pro mechanické usazení do ochranné krabičky. Výsledný návrh desky plošných spojů je na obr.7.1. Podle tohoto návrhu byla následně zhotovena finální deska plošných spojů, která je zachycena na obr.7.2.



Obr. 7.1: Výsledný návrh desky plošných spojů



Obr. 7.2: Výsledná deska plošných spojů

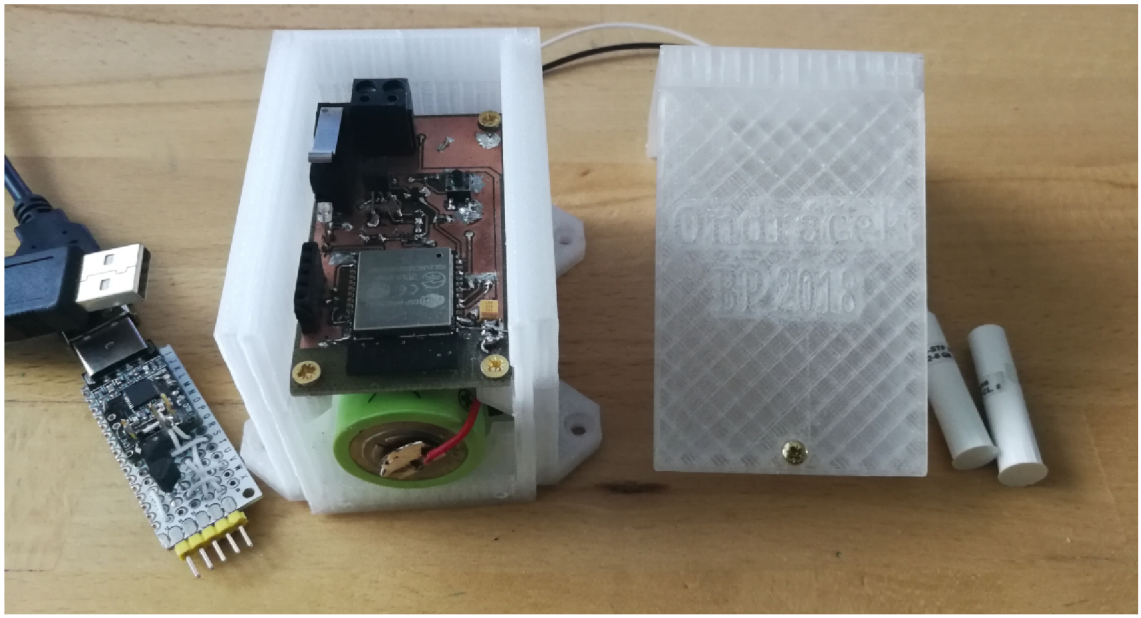
7.2 Ochranná krabička

Ochranná krabička byla vytvořena pomocí 3D tiskárny podle návrhu, který byl vytvořen pomocí bezplatné internetové aplikaci Tinkercad. Požadavky na design krabičky byly takové aby do ní bylo možné výsledné zařízení i s baterií pevně usadit a krabičku uzavřít. Dále musí mít krabička takový tvar aby při uzavření došlo k mechanickému sepnutí PANIC kontaktu a dále musí být vytvořeny dva otvory, jeden pro vyvedení magnetického kontaktu pomocí dvoudrátu a druhý otvor pro indikační LED diodu, která musí být viditelná při běžném provozu pro detekci různých událostí (např. neúspěšný pokus o připojení k Wifi). Výsledná krabička je zachycena na obr.7.3 [18]

Obr. 7.3: Výsledná ochranná krabička

7.3 Výsledné zařízení

Výsledné zařízení je zachyceno na obr. 7.4



Obr. 7.4: Výsledné zařízení

8 PROGRAMOVÁNÍ ZAŘÍZENÍ

V této kapitole budou přímo popsány nejdůležitější části zdrojového kódu zařízení, který realizuje algoritmus popsaný v kapitole návrh řídícího programu.

8.1 Knihovny

V řídícím programu byly použity následující knihovny.

- **Arduino.h** - oficiální arduino framework ulehčující vývoj softwaru
- **WiFi.h** - knihovna pro veškerou práci s Wifi
- **preferences.h** - knihovna umožňující ukládání a čtení dat do zabudované Flash paměti čipu ESP32
- **drivers/adc.h** - knihovna umožňující konfiguraci a použití zabudovaných AD převodníků čipu ESP32
- **Mailer.h** - jednoduchá knihovna pro odesílání e-mailových zpráv pomocí protokolu SMTP. [22]
- **ThingerESP32.h** - knihovna pro realizaci přenosu dat z čipu ESP32 na cloudovou službu Thinger.io pomocí protokolu HTTP
- **OneWire.h** a **DallasTemperature.h** - knihovny umožňující jednoduché čtení dat z teplotního senzoru DS18b20 pomocí sběrnice OneWire

Většina těchto knihoven je volně ke stažení z databáze knihoven vývojového prostředí Platform.io.

8.2 Jádro programu

Program je psaný v Arduino frameworku, který se skládá ze dvou hlavních funkcí. Funkce „Setup()“, která je vykonávána ihned po startu nebo restartu zařízení. Následuje funkce „Loop()“, která je vykonávána stále dokola v nekonečné smyčce. Jelikož zařízení provádí jen několik jednoduchých instrukcí po kterých je uvedeno do režimu spánku, je použita pouze funkce Setup na jejímž konci bude zařízení uspáno. Po následovném probuzení zařízení je opět vykonáván program od začátku funkce Setup.

Zařízení si po každém probuzení nejdříve načte veškerá potřebná data z Flash paměti (pokud jsou k dispozici) a uloží je do dočasných globálních proměnných pro další použití. Do Flash paměti jsou data ukládána pro zachování konfiguračních dat i při případném odpojení napájení. Ukládány jsou pouze konfigurační data, u kterých

nedochází k častému přepisování (z důvodu konečného počtu zápisů do paměti). Ostatní data, jejichž hodnota se často mění jsou ukládána do tzv. RTC (real-time clock) proměnných, které se nachází v RTC paměti, která je napájena i v režimu spánku. Timto jsou data zachována i v režimu spánku.

Po načtení potřebných dat je zavolána funkce *OsetriUdalost*, která reaguje na příčinu probuzení z režimu spánku. Tato funkce bude podrobněji popsána v následující podkapitole.

Před opětovným uspáním čipu je ještě nutno zkontrolovat, zda napětí baterie nekleslo pod limitní hodnotu, která je nastavená parametrem *limitniNapeti* (doporučené nastavení pro čip ESP32 je 2,9 V). Jelikož aktuální napětí baterie kolísá, není porovnávána aktuální hodnota napětí, ale průměrná hodnota za posledních 10 měření. Tato hodnota je ukládána do proměnné s názvem *prumNapeti*. Pokud je hodnota nižší, začnou se uživateli odesílat varovné e-maily (1x denně).

Následuje konfigurace podmínek podle kterých se bude čip probouzet až bude znovu uspán. K těmto účelům je použita tzv. maska pinů (BUTTON_PIN_BITMASK) pomocí které se definují jednotlivé piny, kterými bude možné čip probouzet podle dané konfigurace. Při použití pinů 34 a 35 musí mít maska tvar 0xC00000000. V binární podobě odpovídá každému bitu masky jeden pin. Aby čip reagoval na logickou hodnotu přivedenou na některý z těchto pinů musí být v masce jeho bit nastaven na hodnotu 1, v opačném případě, jeli nastaven na hodnotu 0, je v režimu spánku pin ignorován.

Před ukončením cyklu je zkontrolováno zda došlo ke změně některých konfiguračních dat. Pokud ano tak pomocí funkce SaveChangedPreferencesVariables() jsou tyto změny uloženy do Flash paměti.

Posledním krokem je samotné uspání čipu pomoci funkce *esp_sleep_enable_timer_wakeup(time_in_us)*, kde parametr *time_in_us* představuje dobu v mikrosekundách, po které se má čip automaticky probudit (Timer).

Výpis 8.1: Struktura programu

```
void Setup(){

pin34 = digitalRead(34);       //vysvětleno v další kapitole
pin35 = digitalRead(35);       //vysvětleno v další kapitole
preferences.begin("ESP-preferences",false);
LoadPreferencesVariables();    //Načtení dat z flash paměti
OsetriUdalost();               //podrobněji v další kapitole
ZkontrolujBaterii();
if((pin34==LOW)&&(pin35==LOW)){}
  esp_sleep_enable_ext1_wakeup(BUTTON_PIN_BITMASK,
  ESP_EXT1_WAKEUP_ANY_HIGH);  // běžný provoz
else{esp_sleep_enable_ext1_wakeup(BUTTON_PIN_BITMASK,
  ESP_EXT1_WAKEUP_ALL_LOW}    // po poplachu -
                              // čekání na konec události
SaveChangedPreferencesVariables();  //Uložení dat na flash
preferences.end();
esp_sleep_enable_timer_wakeup(time_in_us);   //uspání
}
void Loop(){}                  //nevyužito
```

8.3 Ošetření událostí

K ošetření události slouží již zmiňovaná funkce *OsetriUdalost*. Pro zjištění příčiny probuzení je použita funkce *esp_sleep_get_wakeup_cause()*. Tato funkce vrací hodnotu v rozmezí 1 - 5 podle příčiny probuzení. Pro účely práce jsou podstatné hodnoty 2 (reprezentující probuzení GPIO pinem) a 3 (Timer). Pomocí těchto hodnot byl vytvořen rozcestník (funkce switch), pro umožnění různého chování systému při různých událostech.

Výpis 8.2: Ošetření událostí

```
void OsetriUdalost(){
  esp_sleep_wakeup_cause_t wakeup_reason;
  wakeup_reason = esp_sleep_get_wakeup_cause();
  switch(wakeup_reason) {
    ...
    case 2: {...} //GPIO
    case 3: {...} //Timer
    ...
    }
```

Nyní si tyto události popíšeme podrobněji.

Probuzení časovačem (Timer)
K této události dochází v okamžik, kdy od uspání čipu uběhně doba definovaná parametrem *time_in_us*. V tomto případě je po programu vyžadováno aby získal aktuální napětí baterie, teplotu a odeslal tyto informace na cloudovou službu. Před samotným měřením je nutné přivést logickou jedničku na GPIO pin 12 aby bylo přivedeno napájení k teplotnímu senzoru a odporovému děliči pro měření napětí (podrobněji již bylo rozebráno v kapitole 4.2.3). Následuje měření teploty pomocí funkcí z knihoven OneWire a DallasTemperature. Dále při měření napětí je nutno nakonfigurovat AD převodník pinu 33 na rozsah 1,1V (útlum 0dB) a přesnost 12 bitů (4096 úrovní mezi 0 a 1,1 Voltů). Výslednou hodnotu napětí získáme přepočtem hodnoty získané z AD převodníku. Následuje průměrování hodnoty se staršími záznamy pro zmenšení vlivu kolísání napětí baterie. Nyní jsou již nachystány data k odeslání do cloudů v podobě proměnných *teplota* a *prumNapeti*. Následuje připojení zařízení k lokální siti pomocí funkce *připojenikWifi()*, jejíž návratová hodnota je true pokud se podařilo k síti připojit. V opačném případě je návratová hodnota false a tato skutečnost je indikována třemi rychlými záblesky indikační led diody. Následuje samotné odesílání dat. Metody odesílání na jednotlivé cloudové služby se od sebe liší a proto jsou podrobněji popsány v dalších kapitolách. Po ukončení těchto činností jsou pomocí pinu 12 opět odpojeny teď již nevyužívané periférie zařízení od napájení a funkce *OsetriUdalost* je ukončena.

Výpis 8.3: Probuzení časovačem (Timer)

```
case 3: {
  digitalWrite(12,HIGH);
  sensors.begin();
  sensors.requestTemperatures();
  teplota = sensors.getTempCByIndex(0); //načtení teploty
  adc1_config_width(ADC_WIDTH_12Bit);
  adc1_congig_channel_atten(ADC1_CHANNEL_5,ADC_ATTEN_0dB);
                        // channel 5 = GPIO pin 33
  double aktualNapeti = adc1_get_voltage(ADC1_CHANNEL_5);
  for(int i =9;i>0;i--){
    zaznamNapeti[i]=zaznamNapeti[i-1];
                       }
  zaznamNapeti[0]=(aktualNapeti/(1202)); //přepočet na volty
  double pomocnyPrumer0;
  for(int i = 0;i<10;i++){
    pomocnyPrumer+=zaznamNapeti[i]}
  prumNapeti=pomocnyPrumer/10;  //výsledné průměrné napětí

  if(pripojenikWifi()==true){
  ... //nasleduje odeslání teplota a prumNapeti na cloud
  }
  break;}
```

Probuzeni pomoci GPIO pinu

Druhý využívaný způsob probouzeni čipu z režimu spánku je pomocí přivedení předdefinované logické hodnoty na předdefinovaný GPIO pin (tzv. external wakeup). Čip ESP32 umožňuje probouzení pomocí různých GPIO pinů zároveň. Tyto piny jsou definovány již zmiňovanou pinovou maskou (BUTTON_PIN_BITMASK). Tento způsob probouzení je využit při rozpojení magnetického kontaktu (přivedení logické jedničky na pin 34) a při rozpojení PANIC kontaktu jakmile dojde k otevření ochranné krabičky (opět logická jednička na pin 35). Jelikož je nutné tyto dvě událost od sebe rozlišit lze použít funkci *esp_sleep_get_ext1_wakeup_status()*, která by měla vracet unikátní hodnotu při probouzení různými piny. Tato funkce se ovšem při testovací fázi prokázala jako nekonzistentní a tím pádem nevyužitelná. Proto byl zvolen jiný způsob. Při startu programu (ihned po rozpojení některého z kontaktů) jsou detekovány a uloženy logické úrovně pinů 34 a 35 pomocí funkce digitalRead() do proměnných *pin34* a *pin35*, které jsou použity při určení příčiny probuzení. Tímto jsou již jednoznačně rozlišeny veškeré možné události, které budou při provozu zařízení nastávat.

Výpis 8.4: Probuzení pomocí GPIO pinu

```
case 2:{ if(pin34==HIGH){...} // Magnetický kontakt
         if(pin35==HIGH){...} // PANIC kontakt
}
```

Nyní budou popsány činnosti, které program při vzniku těchto událostí vykoná.

Rozpojení magnetického kontaktu

Tato událost je indikována pomocí jednoho sekundového záblesku indikační led diody, dále dojde k připojení k lokální síti, odeslání poplachové zprávy na uživatelovu e-mailovou schránku pomocí funkce mail.send() a také odeslání zprávy na cloud. Konkrétní způsoby odeslání zprávy na různé cloudy jsou popsány v dalších kapitolách. Po odeslání poplachových zpráv je funkce *OsetriUdalost* ukončena.

Výpis 8.5: Rozpojeni magnetického kontaktu

```
if(pin34==HIGH){
  digitalWrite(15,HIGH);
  delay(1000);
  digitalWrite(15,LOW);
  if(pripojenikWifi()==true){
    mail.send(to_address, "ESP32 POPLACH", "Bylo detekovano
    rozpojeni magnetického kontaktu (GPIO 34)");
    ... //odeslání zprávy na cloud
                              }
              }
```

Rozpojení PANIC kontaktu

Tato událost je indikována pomocí dvou sekundových záblesků indikační led diody, dále dojde k připojení k lokální síti, odeslání poplachové zprávy na email a cloudové služby.

Výpis 8.6: Rozpojeni magnetického kontaktu

```
if(pin34==HIGH){
  digitalWrite(15,HIGH);
  delay(1000);
  digitalWrite(15,LOW);
  delay(1000);
  digitalWrite(15,HIGH);
  delay(1000);
  digitalWrite(15,LOW);
  if(pripojenikWifi()==true){
    mail.send(to_address, "ESP32 POPLACH", "Bylo detekovano
    rozpojeni PANIC kontaktu (GPIO 35)");
    ... //odeslání zprávy na cloud
                              }
  menu();       }
```

Rozpojení PANIC kontaktu znamená že byla buď poškozena krabička (sabotáž) nebo byla krabička otevřena uživatelem pro ruční konfiguraci některých parametrů přes sériovou linku. Pro tyto účely byla vytvořena funkce menu() která je aktivována po odeslání poplachových hlášení o rozpojení kontaktu. V tuto chvíli je předpokládano, že uživatel připojí adaptér s USB/UART převodníkem a začne zobrazovat komunikaci pomocí seriového monitoru. První informaci, kterou uživatel uvidí je výzva k zadání hesla a zařízení dále čeká na vstupní informace od uživatele. Vstupní informace, které jsou poslány sériovou linkou od uživatele k zařízení jsou pomocí funkce Serial.readString() postupně ukládána dokud nedojde k stlačení klávesy enter (znak „\n“). Pokud je výsledný řetězec shodný s řetězcem uloženým v proměnné *menu-heslo*, uživatel získá přístup ke konfiguračnímu menu, pokud ne, je uživatel požádán o opětovné zadání hesla. V průběhu čekání na zadávání hesla je kontrolován čas, který uběhl od otevření krabičky nebo od posledního zadání hesla a pokud tento čas překročí nastavenou hodnotu (zde nastavena 1 minuta), dojde k automatickému uspání zařízení. Toto opatření neslouží pouze pro zmenšení spotřeby ale řeší i situaci kdy dojde pouze k poškození krabičky po které nenásleduje komunikace po seriové lince. V konfiguračním menu se uživatel pohybuje pomocí zadávání číselných hodnot z možností, které mu jsou právě nabídnuty. Konfigurace jednotlivých parametrů probíhá stejně jako při zadávání hesla pomocí funkce Serial.readString(). Zdrojový kód funkce menu() obsahuje téměř 200 řádků kódu a jedná se pouze o kombinace funkce switch, výpisů textu na seriový monitor a zadávání vstupních informací uživatelem. Proto je níže zobrazena pouze hierarchie celého menu a zdrojový kód funkce je společně s celým zdrojovým kódem uložen na CD/DVD které je součástí práce.

Hierarchie konfiguračního menu

```
Menu
├── 1.Nastavit parametry wifi
│   ├── 1.Nastavit SSID
│   ├── 2.Nastavit Heslo
│   ├── 3.Nastavit Hostname zařízení
│   └── 4.Zpět
├── 2.Nastavit parametry emailu
│   ├── 1.Nastavit emailovou adresu ze které bude zařízení odesílat hlášení
│   │   (např.: jmeno@gmail.com)
│   ├── 2.Nastavit uživatelské jméno (typicky odpovídá emailové adrese)
│   ├── 3.Nastavit přihlašovací heslo
│   ├── 4.Nastavit hostname serveru (např.: pro gmail = smtp.gmail.com)
│   ├── 5.Nastavit smtp port (typicky 25, 465 nebo 587)
│   ├── 6.Nastavit adresu na kterou budou ohlašovány události
│   │   (např.: jmeno@gmail.com)
│   └── 7.Zpět
├── 3.Nastavit parametry thingspeaku
│   ├── 1.Nastavit writeAPIKey
│   └── 2.Zpět
├── 4.Nastavit parametry thingerIO
│   ├── 1.Nastavit USERNAME
│   ├── 2.Nastavit DEVICE_ID
│   ├── 3.Nastavit DEVICE_CREDENTIAL
│   └── 4.Zpět
├── 5.Nastavit parametry Ubidots
│   ├── 1.Nastavit Authentication Token
│   ├── 2.Nastavit Variable ID - Napeti
│   ├── 3.Nastavit Variable ID - Teplota
│   ├── 4.Nastavit Variable ID - Magneticky kontakt
│   ├── 5.Nastavit Variable ID - PANIC kontakt
│   └── 6.Zpět
├── 6.Nastavit interval probuzení
├── 7.Nastavit dolni hranici napětí baterie
├── 8.Změnit heslo pro vstup do menu
├── 9.Výpis všech nastavitelných parametrů
└── 10.Konec
```

Pro komunikaci se zařízením byly úspěšně vyzkoušeny tři různé „inteligentní“ sériové monitory - sériový monitor Platformio, sériový monitor Arduino.ide a PuTTY. Proto pro práci se zařízením doporučuji použít jeden z nich z důvodu uživatelsky přívětivých funkcí jako echo nebo line editing.

8.4 Implementace cloudových služeb

V následujících podkapitolách je popsán postup registrace a konfigurace jednotlivých cloudových služeb a jejich implementace do zdrojového kódu zařízení.

8.4.1 Připojení zařízení k Thingspeaku

Připojení IoT zařízení (v našem případě čipu ESP32) ke službě Thingspeak probíhá v několika jednoduchých krocích. Nejprve je nutné se bezplatně zaregistrovat na webových stránkách Thingspeaku. Po založení účtu můžeme již vytvořit nový kanál, kde se budou naše data ukládat a zobrazovat. Kanálu je po jeho vytvoření přiřazeno ID kanálu (channelID) a dva API klíče, jeden pro zápis (Write API Key), druhý pro čtení (Read API Key). Tyto parametry kanálu jsou použity v řídícím programu zařízení. Těmito kroky je základní nastavení cloudu pro funkci projektu kompletní (kromě vedlejších nastavení, jako je nastavení jednotek a popisů os grafů či množství zobrazovaných hodnot).

Dále je nutné upravit řídící program tak aby naměřená data při správné konfiguraci odeslal na cloudový kanál. V principu je nutné se pomocí HTTP protokolu připojit k serveru „api.thingspeak.com“ a odeslat pomocí API klíče řetězec ve tvaru „field1=data1&field2=data2&field3=....“ (kde „field“ určuje do kterého grafu bude hodnota uložena a „data“ představující odesílanou hodnotu, například naměřenou teplotu). Pro tyto účely byla vytvořena funkce *zapisDoThingSpeaku*, jejíž vstupní parametr je zmiňovaný řetězec.[19]

Výpis 8.7: Ukázka zdrojového kódu pro implementaci Thingspeaku

```
void zapisDoThingSpeaku(String body){
 if(client.connect("api.thingspeak.com",80){
  client.print("POST /update HTTP/1.1\n");
  client.print("Host: api.thingspeak.com\n");
  client.print("Connection: close\n");
  client.print("X-THINGSPEAKAPIKEY: "
              + String(writeAPIKey)+"\n");
  client.print("Content-Type:
              application/x-www-form-urlencoded\n");
  client.print("Content-Length: ");
  client.print(body.lenght());
  client.print("\n\n");
  client.print("body");
  client.print("\n\n");
   }
 client.stop();
 }
...
zapisDoThingSpeaku("field1="+String(prumNapeti)+"&field2="
                +String(teplota));  // příklad volání funkce
...
```

8.4.2 Připojení zařízení k Thinger.io

Připojení zařízení ke cloudové službě Thinger.io je také velice jednoduché. Je zapotřebí se zaregistrovat na webových stránkách Thinger.io. Po připojení je první krok vytvoření nového zařízení v záložce Devices, kde je zařízení přiřazen unikátní řetězec nazvaný „device credentials“ a uživatelem zvolený identifikátor nazvaný „device ID“. Tento řetězec slouží k jednoznačné identifikaci daného zařízení a je využíván společně s identifikátorem a přihlašovacím jménem (username) v řídícím programu. Pro ukládání odeslaných dat slouží tzv. „Data bucket“, z kterého lze data exportovat nebo přímo na stránkách vizualizovat pomocí tzv. „Dashboardu“.

Implementace do řídícího programu je velice jednoduchá díky knihovně přímo určené pro ESP32 s názvem ThingerESP32.h. Knihovna je volně dostupná ke stažení přímo ve vývojovém prostředí Platform.io. Po přidání knihovny stačí jen specifikovat username, device ID a device credentials a pomocí několika jednoduchých příkazů specifikovat odesílané hodnoty, databucket do kterého budou hodnoty ukládány a nakonec samotná data odeslat.[20]

Výpis 8.8: Ukázka zdrojového kódu pro implementaci Thinger.io

```
#include <ThingerESP32.h>
char USERNAME[100];
char DEVICE_ID[100];
char DEVICE_CREDENTIAL[100];
ThingerESP32 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);
...
void OsetriUdalost{
...
thing["Magnet"] >> outputValue("Probuzeno pinem 34 -
                                Magneticky kontakt");
thing.handle();
thing.write_bucket("MagnetKontakt","Magnet");
...}
```

8.4.3 Připojení zařízení k Ubidots

Připojení k Ubidots je obdobné jako u již zmíněných cloudových služeb. Nejdříve je nutná registrace na stránkách, bohužel Ubidots je služba pouze placená. V práci je využita neplacená verze Ubidots for Education, která má oproti placené verzi jistá omezení. Uživateli je příděleno zdarma 5000 kreditů, což je měna používaná pro rezervaci prostředků služby Ubidots. Po registraci stačí pouze vytvořit nové zařízení (záložka „Devices“ a v něm nadefinovat potřebná uložiště (záložka Variables) do kterých budou ukládány odeslaná data. Každé uložiště má své unikátní „Variable ID“, které je použito v řídícím programu společně s přihlašovacím tokenem (Authentication Token). Pro vizualizaci dat lze vytvořit v záložce Dashboard grafické komponenty (tzv. Widgety), který lze libovolně editovat podle uživatelových potřeb.

Implementace do řídícího programu zařízení je velmi podobné službě thingspeak. Jde o připojení na server „things.ubidots.com“ pomocí protokolu HTTP a odeslání řetězce ve tvaru „value:data, context:referencePopisu:popis“ pomocí přihlašovacího tokenu a Variable ID . Položka „data“ představuje odesílanou hodnotu což je v našem případě například hodnota napětí baterie a položka „popis“ je libovolný text, který je s odeslanou hodnotou svázaný (referencePopisu je řetězec, kterým se v grafických komponentech odkazujeme na samotný popis). Ubidots bohužel nepodporuje text jako platnou odesílanou hodnotu, proto v případě hlášení o rozpojení např. magnetického kontaktu (což je reprezentováno textem nikoliv číselnou hodnotou) je zapotřebí tuto zprávu zakomponovat do popisu nějaké proměnné (například napevno nadefinované hodnotě 1) a v grafické komponentě zobrazit pouze popis dané proměnné bez samotné odesílané hodnoty. Pro tyto účely byla vytvořena funkce

zapisDoUbidots jejíž vstupní parametry jsou již zmiňované Variable ID, odesílaná hodnota a její popis.[21]

Výpis 8.9: Ukázka zdrojového kódu pro implementaci Ubidots

```
void zapisDoUbidots(String var_id,String value,String context)
{
String var = "{\"value\": " + value +",\"context\":
                    {\"popis\": \""+context+"\"}"+ "}";
String length = String(var.leght());
if(client.connect("things.ubidots.com),80)
  {
  client.println("POST /api/v1.6/variables/"+var_id+"/values
                                                HTTP/1.1");
  client.println("Content-Type: application/json");
  client.println("Content-Length: " + lenght);
  client.println("X-Auth-Token: "+String(tokenUbidots));
  client.println("Host: things.ubidots.com\n");
  client.print(var);
  }
  client.stop();
}
...
zapisDoUbidots(String(variable_id1), String(prumNapeti),
              "Napeti baterie"); // příklad volání funkce
...
```

8.5 Emailová schránka - bezpečnostní riziko

Knihovna Mailer.h vyžaduje pro svoji funkčnost přihlašovací údaje k emailové schránce (ukládáno do proměnných *smtp_username*, *smtp_password* a *smtp_from_address*) ze které zprávy odesílá. Tyto informace nejsou nijak šifrované a vzniká tak bezpečnostní riziko ztráty osobních dat (útočníka dělí od přihlašovacích údajů pouze přihlašovací heslo pro vstup do menu).

V práci byla úspěšně vyzkoušena nejčastěji používaná emailová služba Gmail od firmy Google. Při použití Gmailu je ovšem nutné v zabezpečení google účtu povolit méně bezpečné aplikace. Bez tohoto nastavení se nebylo zařízení schopno přihlásit do emailové schránky. Tento problém byl zřejmě způsoben nezašifrovanou komunikací zařízení, kterou služba Gmail běžně nepodporuje. Tímto nastavením se stává emailová schránka zranitelná a proto při běžném provozu doporučuji vytvořit schránku výhradně pro tyto účely aby nedošlo ke zbytečnému ohrožení jiných citlivých dat.[22]

8.6 Porovnání parametrů vybraných cloudových služeb

V práci byly vyzkoušeny celkem tři různé cloudové služby. Každá má své výhody a nevýhody. Následující tabulka obsahuje výpis některých důležitých parametrů a omezení, které by uživatel měl při jejich použití znát. [23, 24, 25]

	Thingspeak	Thinger.io	Ubidots
Maximální počet připojených zařízení	Teoreticky neomezený	2	doporučeno max 5 (+omezeno kredity)
Maximální rychlost posílání zpráv	1 zpráva/15 sec	Pro každý data bucket 1 zpráva/60 sec	Dohromady ze všech zařízení 60 zpráv / 1 min
Maximální doba uchování dat	10 mil. zpráv (3+ roky)	1 rok	3 měsíce
Možnosti formátu exportovaných dat	JSON, XML, CSV	JSON, ARFF, CSV	CSV
Možnost specifikace exportovaných dat	Pouze 100 posledních záznamů	Všechny záznamy, možnost specifikace časového intervalu s přesností na **minuty**	Všechny záznamy, možnost specifikace časového intervalu s přesností na **dny**
Vizualizace dat na webových stránkách	Pouze grafy, text lze uložit ale není možné ho na stránce vizualizovat	Nejrůznější předdefinované widgety (zobrazeni hodnot, textu, grafy atd.)	Různé widgety s možností vlastní úpravy (např. zobrazeni časového razítka poplašné zprávy)

Tab. 8.1: Různé parametry cloudových služeb

Z vyzkoušených cloudů měla pro účely práce nejlepší parametry služba Thinger.io. Je to především díky možnosti přesné specifikace a výběru formátu exportovaných dat a vizualizaci dat. Dále jako jediná má svojí knihovnu, což bylo při programování zařízení velmi přívětivé. Nevýhodou je relativně malá maximální rychlost posílání zpráv, což ale v této práci není omezující, neboť zařízení tak často odesílat zprávy nebude.

Nejhorší parametry měla služba Thingspeak. Především se jedná o velice omezené možnosti exportu dat a omezené možnosti vizualizace dat na webových stránkách.

Jediná znatelná výhoda je maximální doba uchování dat, která při maximální rychlosti posílání zpráv umožňuje zobrazit data přibližně 3 roky staré, při měnší rychlosti i starší.

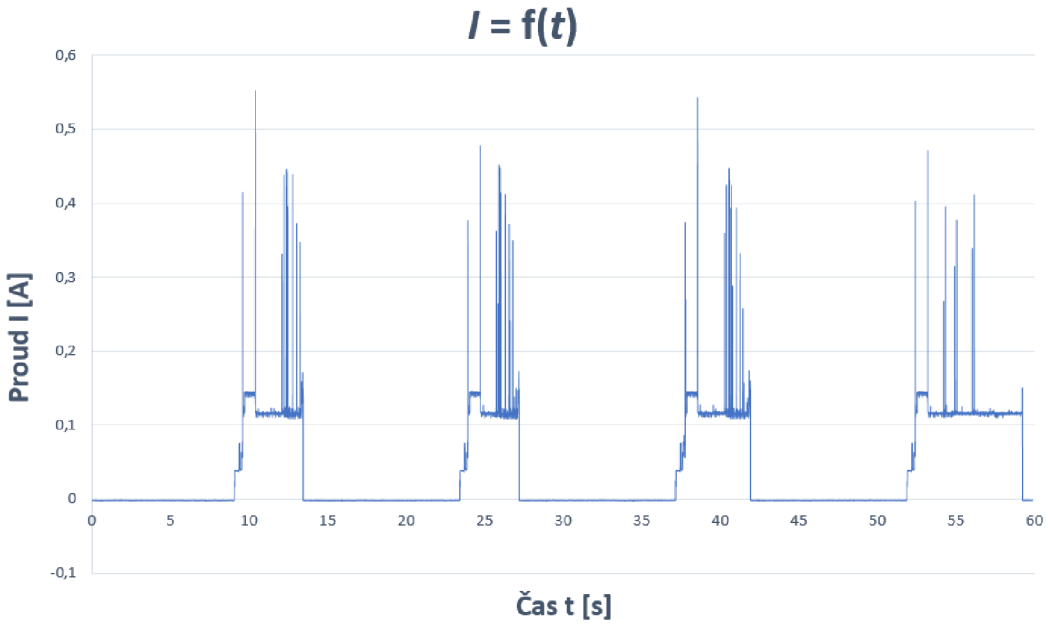
Placené verze služeb většinu těchto omezení nemají. Bližší specifikace jsou k nalezení na příslušných webových stránkách.

9 MĚŘENÍ SPOTŘEBY ZAŘÍZENÍ

Pro měření byl použit stejnosměrný analyzátor elektrické energie N6705B od firmy Agilent Technologies a digitální multimetr 34401A rovněž od firmy Agilent.

Při měření vzniklo několik problémů. Prvním problémem byl fakt, že spotřeba zařízení je závislá především na délce intervalu probouzení což je parametr, který je v souladu se zadáním nastavitelný. Výsledkem této kapitoly tudíž není jedna hodnota ale graf závislosti maximální provozní doby na nastavení tohoto parametru.

Druhým problémem byla skutečnost, že se zařízení bude při běžném nastavení probouzet jednou za dobu v řádu jednotek až desítek minut. Měření by tedy bylo zdlouhavé a vnitřní paměť analyzátoru by byla pro potřebný vzorkovací kmitočet nedostačující. V reálném provozu je spotřeba v režimu spánku téměř konstantní. Proto byla naměřena digitálním multimetrem 34401A, a špičky (aktivní režim) se složitějšími průběhy byly naměřeny pomocí analyzátoru N6705B (s nastaveným krátkým intervalem probouzení). S touto konfigurací bylo možné naměřit více špiček najednou pro porovnání. Naměřené hodnoty byly z analyzátoru exportovány do souboru .csv a následně upraveny do podoby grafu 9.1.



Obr. 9.1: Průběh odebíraného proudu v čase

Z grafu je již patrný další problém, který zavádí největší chybu měření. Je jím nekonzistentní doba, kterou zařízení stráví v aktivním režimu. Při měření se tato doba pohybovala okolo 4,5 sekundy což přibližně odpovídá 0,145 mAh spotřebovaného

náboje za jednu špičku. Tato hodnota náboje jedné špičky byla určena pomocí rovnice 6.7 z kapitoly 6.2 (v principu jde o výpočet plochy pod křivkou pomocí metody obdélníků). Ovšem v ukázkovém grafu se „podařilo“ zachytit i interval, kde dosáhla doba v aktivním režimu hodnoty 7,35 sekundy, což odpovídá přibližně hodnotě 0,235 mAh. K těmto větším výkyvům dochází náhodně a ne příliš často, nicméně musí být zahrnuty do následujících úvah. Všechny tyto výkyvy jsou způsobeny především kolísáním doby připojování k lokální síti (k nejbližšímu přístupovému bodu = access point AP). Toto kolísání je způsobeno různými faktory, například umístěním zařízení vzhledem k přístupovému bodu (AP), rušením od jiných stanic, aktuálním vytížením AP a jiné. Většina těchto rušivých vlivů je bohužel pouze částečně odstranitelná nebo neodstranitelná. Z těchto důvodu je do následující výpočtů zavedena velká chyba měření a proto se bude dále jednat jen o orientační výsledky pro nastínění přibližné spotřeby a provozní doby zařízení.

Z naměřených hodnot špiček budeme uvažovat jako průměrnou dobu zařízení v aktivním režimu 5 sekund (zaokrouhleno na vyšší hodnotu z důvodu nekonzistentnosti naměřených hodnot). Takové špičce odpovídá spotřeba 0,165 mAh. Dále je nutné znát spotřebu zařízení v režimu spánku. Pomocí digitálního multimetru byl naměřen stabilní odběr proudu 5,1 μA. Výpočet průměrného odběru proudu (v mA) za jeden úplný cyklus, což představuje interval ve kterém je zařízení bez přerušení v režimu spánku a jedné špičky, je dán vztahem

$$\begin{aligned} I_{\text{prum[mA]}} &= \frac{Q}{t} = \frac{Q_{\text{SLEEP}} + Q_{\text{ACTIVE}}}{t_{\text{SLEEP}} + t_{\text{ACTIVE}}} = \\ &= \frac{I_{\text{SLEEP}} \cdot t_{\text{SLEEP}} + \sum_{n=1}^{P} I_n \cdot t_{vz[h]}}{t_{\text{SLEEP}} + t_{\text{ACTIVE}}} = \\ &= \frac{\frac{5{,}1}{1000} \cdot \frac{x}{3600} + 0{,}165}{\frac{x+5}{3600}}, \end{aligned}$$

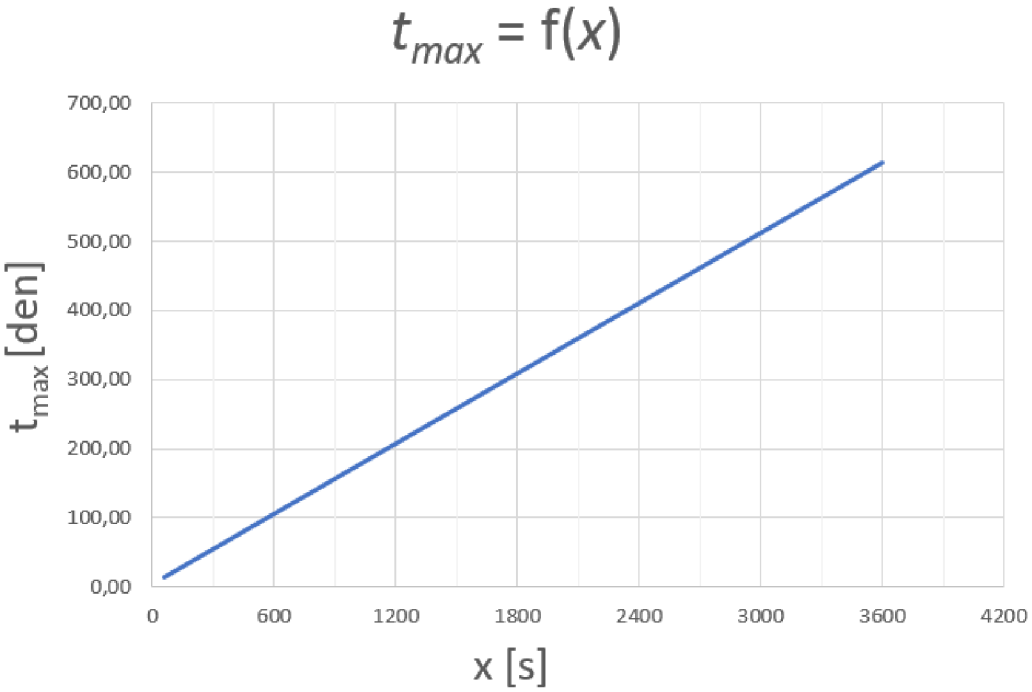
kde x je hodnota nastavitelného parametru *interval probouzení* v sekundách.

Teoretická maximální provozní doba (s použití baterie ANR26650 s kapacitou 2500 mAh) je potom dána vztahem

$$t_{max} = \frac{C_{\text{bat[mA]}}}{I_{\text{prum[mA]}}} = \frac{2500}{I_{\text{prum[mA]}}} \tag{9.1}$$

Vzniklá závislost teoretické maximální provozní doby t_{max} na parametru *interval probouzení* x je zobrazen v grafu. 9.2

Tabulka několika hodnot intervalu probouzení a jejich odpovídající provozní doba



Obr. 9.2: Závislost provozní doby na parametru *interval probouzení*

x	t_{max}
1 min	11,4 dnů
5 min	53,4 dnů
10 min	105,6 dnů
20 min	209,2 dnů
1 h	613,2 dnů

Tab. 9.1: Typické hodnoty *intervalu probouzení* a odpovídající provozní doba

9.1 Shrnutí výsledků měření

Z výsledků měření můžeme prohlásit, že úsporný návrh zařízení byl správný, jelikož spotřeba v obou režimech (aktivní a spánek) odpovídá tabulkovým hodnotám v datasheetu čipu ESP32. Nedochází tedy ke zbytečnému odběru proudu z důsledku špatného designu.

Dominantní část spotřeby celého zařízení je dána odezvou vnější sítě na požadavek o připojení a tudíž je v rámci práce neovlivnitelná. V konkrétních případech by se dala situace ovlivnit způsobem, že by se zařízení nepřipojovalo pomocí DHCP protokolu ale bylo do sítě připojeno pomocí statické adresy, čímž by se celý proces připojování k síti urychlil a mohlo by dojít ke zvýšení provozní doby zařízení.

Výsledná závislost provozní doby na intervalu probouzení je pouze orientační, reálná provozní doba se od vypočtené hodnoty bude lišit. Rekapitulace většiny příčin, které měly vliv na nepřesnost měření je uveden níže.

- Proměnlivá délka setrvání v aktivním režimu způsobené
  - Kolísání doby připojování k lokální síti
  - Rychlost připojení k serverům cloudových služeb je také proměnlivá (ovšem vzniklé zpoždění se svojí délkou ani z daleka neblíží době čekání na připojení k lokální síti)
- Samovolné vybíjení baterie - baterie není dokonalá a v průběhu času postupně ztrácí svůj naakumulovaný náboj
- U reálné baterie nelze využít celou kapacitu baterie (tento problém by se dal teoreticky omezit zakomponováním měniče napětí, který by umožnil využít větší část kapacity baterie, ovšem měnič samotný má vlastní spotřebu, což by se z úsporného hlediska zařízení nemuselo ve výsledku vyplatit)
- Události - v kapitole bylo počítáno pouze s probouzením zařízení po uplynutí nastavené doby na časovači (Timer). Ovšem zařízení se bude probouzet také při rozpojování magnetického kontaktu a PANIC kontaktu. Zde je charakter probouzení závislý na povaze místa usazení zařízení. Tudíž nelze nijak do výpočtů spotřeby zahrnout.
- Zaokrouhlování ve výpočtech, z důvodu nekonzistentnosti naměřených hodnot.

10 ZÁVĚR

Cílem bakalářské práce bylo vytvořit jednoduchý detektor otevření dveří, oken či libovolného prostoru pomocí magnetického kontaktu a WiFi modulu ESP32. Zařízení je napájeno bateriově a bylo navrženo tak aby mělo dlouhou provozní dobu bez nutného dobíjení baterie. Tohoto bylo docíleno v hardwarové části například připojováním napájecího napětí k periferiím zařízení pomocí MOSFET tranzistorů pouze v okamžiku jejich využití nebo použitím LiFePo4 baterie poskytující napětí 3,2 V díky čemuž nebylo nutné zakomponovat regulátor napětí. Softwarová část ovlivnila úspornost zařízení pomocí efektivního přepínání mezi aktivním režimem a režimem spánku při zachování požadované funkčnosti zařízení. Výsledné zařízení je možno konfigurovat přes sériovou linku po připojení USB/UART převodníku do konektoru zařízení, který je přístupný pouze při otevření krytu zařízení. Po konfiguraci těchto parametrů je zařízení schopno připojení k jednoduché domácí WiFi síti, přes kterou odesílá hlášení o rozpojení magnetického kontaktu a mikrospínače hlídajícího otevření ochranného krytu zařízení na uživatelovu emailovou adresu a dále tato hlášení a pravidelné záznamy o teplotě a napětí baterie posílá do různých cloudových úložišť, kde jsou tyto informace vizualizovány a také je lze dále exportovat do souborů různého formátů.

Práce popisuje celý proces realizace zařízení. První část práce je teoreticky zaměřená. Je zde rozebrána problematika Internetu věcí, dále jsou popsány parametry a funkce konkrétního softwaru a součástek, které jsou v rámci práce použity. Druhá část práce je zaměřena na celkový návrh zařízení. Je zde navrhnuto schéma zapojení, blokové schéma zachycující algoritmus použitý pro funkci řídícího programu zařízení a také jsou zde uvedeny způsoby, kterými lze měřit spotřebu zařízení. Třetí část je již zaměřena na samotnou realizaci zařízení, což zahrnuje návrh a výrobu desky plošných spojů a ochranné krabičky, ve které je zařízení usazeno. Dále je zde podrobněji popsáno samotné programování čipu, implementace cloudových služeb a jejich konfigurace. V rámci práce byly vyzkoušeny a naimplementovány tři různé cloudové služby - Thingspeak, Thinger.io a Ubidots. Byly použity neplacené verze těchto služeb, které mají různé vlastnosti a omezení. Tyto rozdíly byly v práci porovnány a jako nejoptimálnější byl pro účely této práce vybrán cloud Thinger.io. Na závěr práce byla naměřena orientační spotřeba zařízení, teoretická maximální provozní doba a byly popsány vlivy, které zapříčinily nepřesnost tohoto měření. Výsledné zařízení je připraveno k instalaci a uvedení do běžného provozu.

LITERATURA

[1] Portál I2oT, *Internet věcí* [online]. 2015 [cit. 27. 11. 2017]. Dostupné z URL: <http://i2ot.eu/internet-of-things/>.

[2] Portál idnes.cz, *Firmy bojují o ovládnutí internetu věcí* [online]. poslední aktualizace 17. 3. 2015 [cit. 27. 11. 2017]. Dostupné z URL: <https://technet.idnes.cz/souboj-internetu-veci-iot-097-/hardware.aspx?c=A150316_161724_hardware_vse>.

[3] Portál Network Computing, *The Internet of Things and IP address needs* [online]. poslední aktualizace 16. 4. 2015 [cit. 27. 11. 2017]. Dostupné z URL: <https://www.networkcomputing.com/networking/internet-things-ip-address-needs/1170065007>.

[4] IoT portál, *Využití IoT* [online]. 2016 [cit. 27. 11. 2017]. Dostupné z URL: <https://www.iot-portal.cz/vyuziti/>.

[5] Portál Espressif, *ESP32 overview* [online]. [cit. 27. 11. 2017]. Dostupné z URL: <http://espressif.com/en/products/hardware/esp32/overview>.

[6] Portál root.cz, *ESP32 je tu. Co přinese nástupce ESP8266?* [online]. poslední aktualizace 1. 9. 2016 [cit. 27. 11. 2017]. Dostupné z URL: <https://www.root.cz/clanky/esp32-je-tu-co-prinese-nastupce-esp8266/>.

[7] Portál Espressif, *ESP32 datasheet* [online]. poslední aktualizace 26. 10. 2017 [cit. 27. 11. 2017]. Dostupné z URL: <http://espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>.

[8] Portál Espressif, *ESP32 Modules and Boards* [online]. [cit. 27. 11. 2017]. Dostupné z URL: <https://dl.espressif.com/doc/esp-idf/latest/hw-reference/modules-and-boards.html>.

[9] Portál jlelektronik.sk, *jazýčkové spínače* [online]. poslední aktualizace 8. 11. 2017 [cit. 27. 11. 2017]. Dostupné z URL: <http://www.jlelektronik.sk/produkty-databaza/5.Elektromechanické%20súčiastky/jazyckove_spinace.pdf>.

[10] Portál Maxim Integrated, *DS18B20 datasheet* [online]. [cit. 27. 11. 2017]. Dostupné z URL: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>.

[11] Portál Arduino Návody, *Teplotní senzor DS18B20* [online]. poslední aktualizace 28. 6. 2016 [cit. 27. 11. 2017]. Dostupné z URL: <http://navody.arduino-shop.cz/navody-k-produktum/teplotni-senzor-ds18b20.html>.

[12] Portál PlatformIO, *What is PlatformIO?* [online]. [cit. 27. 11. 2017]. Dostupné z URL: <http://docs.platformio.org/en/latest/what-is-platformio.html>.

[13] Portál instaluj.cz, *Eagle* [online]. [cit. 27. 11. 2017]. Dostupné z URL: <https://www.instaluj.cz/eagle>.

[14] Portál MetaCentrum, *Definice cloudu* [online]. [cit. 27. 11. 2017]. Dostupné z URL: <https://wiki.metacentrum.cz/wiki/Definice_cloudu>.

[15] Portál mathworks.com, *ThingSpeak* [online]. [cit. 27. 11. 2017]. Dostupné z URL: <https://www.mathworks.com/help/thingspeak/?requestedDomain=www.mathworks.com>.

[16] Portál thinger.io, *thinger.io platform* [online]. [cit. 15. 5. 2018]. Dostupné z URL: <https://thinger.io/>.

[17] Portál ubidots.com, *Cloud tools to build your business* [online]. [cit. 15. 5. 2018]. Dostupné z URL: <https://ubidots.com/platform/>.

[18] Portál tinkercad.com, *autodesk tinkercad* [online]. [cit. 15. 5. 2018]. Dostupné z URL: <https://www.tinkercad.com/>.

[19] Portál nothans.com, *Measure Wi-Fi Signal Levels with the ESP8266 and ThingSpeak* [online]. poslední aktualizace 1. 2. 2017 [cit. 27. 11. 2017]. Dostupné z URL: <http://nothans.com/measure-wi-fi-signal-levels-with-the-esp8266-and-thingspeak>.

[20] Portál Github.com, *thinger-io / Arduino-Library* [online]. poslední aktualizace 15. 3. 2018 [cit. 16. 5. 2018]. Dostupné z URL: <https://github.com/thinger-io/Arduino-Library/blob/master/examples/ESP32/ESP32.ino>.

[21] Portál ubidots.com, *Connect the ESP32-DevKitC to Ubidots in under 30 minutes* [online]. [cit. 16. 5. 2018]. Dostupné z URL: <http://help.ubidots.com/connect-your-devices/connect-the-esp32-devkitc-to-ubidots-in-under-30-minutes>.

[22] Portál kerikeri.top, *Jak poslat email z ESP32* [online]. poslední aktualizace 8. 4. 2017 [cit. 16. 5. 2018]. Dostupné z URL: <`https://kerikeri.top/posts/2017-04-08-esp32-mail/`>.

[23] Portál thingspeak.com, *ThingSpeak Licensing FAQ* [online]. [cit. 16. 5. 2018]. Dostupné z URL: <`https://thingspeak.com/pages/license_faq`>.

[24] Portál thinger.io, *Flexible Pricing* [online]. [cit. 16. 5. 2018]. Dostupné z URL: <`https://thinger.io/pricing/`>.

[25] Portál ubidots.com, *How Ubidots for Education Credits Work* [online]. [cit. 16. 5. 2018]. Dostupné z URL: <`http://help.ubidots.com/faqs-and-troubleshooting/how-ubidots-for-education-credits-work`>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

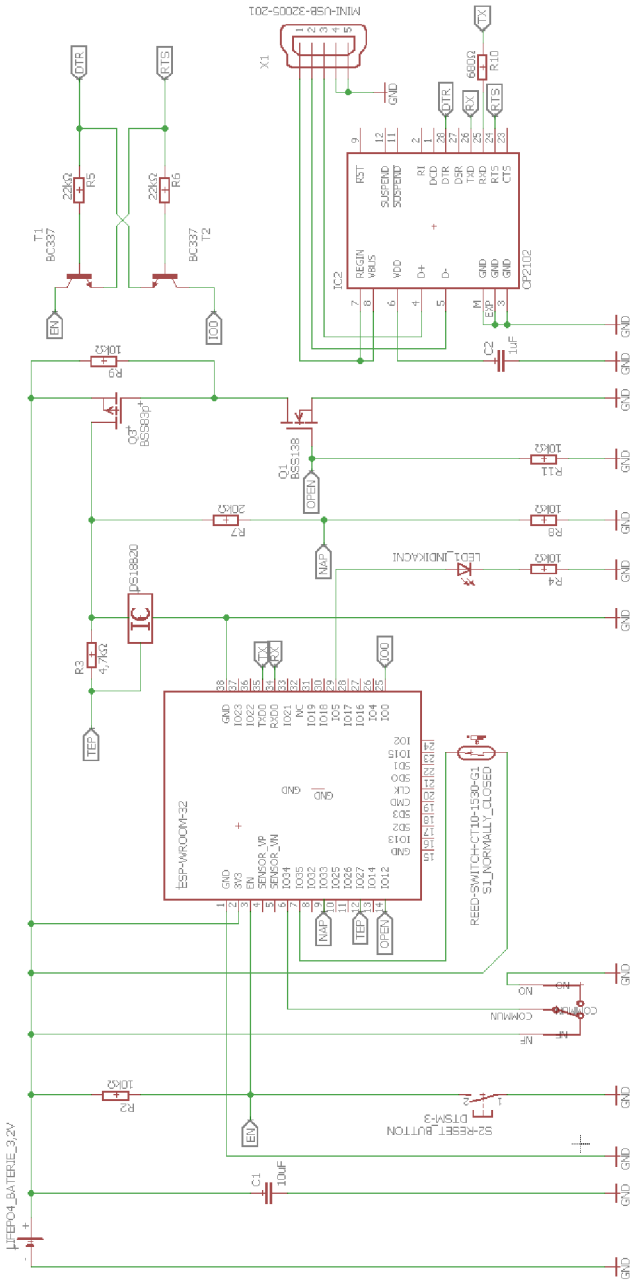
IoT	Internet of Things, internet věcí
IP	Internet protocol, internetový protokol
RS232	Recommended standard 232, doporučený standard 232
RS485	Recommended standard 485, doporučený standard 485
I2C	Inter Integrated Circuit, interní sériová sběrnice
SPI	Serial Peripheral Interface, sériové periférní rozhraní
IIOT	Industrial Internet of Things, průmyslový internet věcí
SoC	System on Chip, systém na čipu
SRAM	Static Random Access Memory, statická paměť s náhodným přístupem
CPU	Central Processing Unit, centrální procesorová jednotka
IEEE 802.11	Standard pro Wifi vytvořený institutem pro elektrotechnické a elektronické inženýrství
WEP	Wired Equivalent Privacy, soukromí ekvivalentní drátovým sítím
WPA,WPA2	Wifi Protected Accesss, chráněný přístup k Wifi
PSK	Phase Shift Keying, klíčování fázovým posunem
BLE	Bluetooth Low Energy, Bluetooth s nízkou spotřebou energie
PCB	Printed Circuit Board, deska plošných spojů
IPEX	Konektor vytvořený firmou IPEX
GPIO	General-Purpose Input/Output, programovatelné vývody
UART	Universal asynchronous receiver-transmitter, univerzální asynchronní přijímač/vysílač
A/D převodník	Převodník analogové hodnoty na digitální
USB	Universal Serial Bus, univerzální sériová sběrnice
NC	Normally Closed, normalně zavřený
NO	Normally Open, normalně otevřený
LiFePo4	Lithium-železo-fosfát
SaaS	Software as a Service, Software jako služba
PaaS	Platform as a Service, Platforma jako služba
IaaS	Infrastructure as a Service, Infrastruktura jako služba
API	Application Programming Interface, rozhraní pro programování aplikací
HTTP	Hypertext Transfer Protocol, protokol určený pro výměnu hypertextových dokumentů
VLAN	Virtual Local Area Network, virtuální lokální síť
VPN	Virtual Private Network, virtuální privátní síť
RTC	Real Time Clock, hodiny reálného času

MOSFET	Metal Oxide Semiconductor Field Effect Transistor, polem řízený tranzistor se strukturou kov(M)-oxid(O)-polovodič(S)
DTR	Data Terminal Ready, datový terminál připraven (pin)
RTS	Ready To Send, připraven k odesílání (pin)
SSID	Service Set Identifier, název Wifi sítě
LED	Light-Emitting Diode, dioda emitující světlo
PCB	Printed Circuit Board, deska plošných spojů
TCP	Transmission Control Protocol, spojově orientovaný protokol pro spolehlivý přenos dat
UDP	User Datagram Protocol, nespojově orientovaný protokol pro nespolehlivý/bez záruky doručení přenášených dat
MQTT	Message Queuing Telemetry Transport, protokol pro předání zpráv mezi klienty pomocí centrálního bodu
CSV	Comma Seperated Values, Čárkou oddělené hodnoty
JSON	JavaScript Object Notation, Javascriptová objektová notace
XML	eXtensible Markup Language, rozšířitelný značkovací jazyk
ARFF	Attribute-Relation File Format - Souborový formát pro atributy se souvislostmi

SEZNAM PŘÍLOH

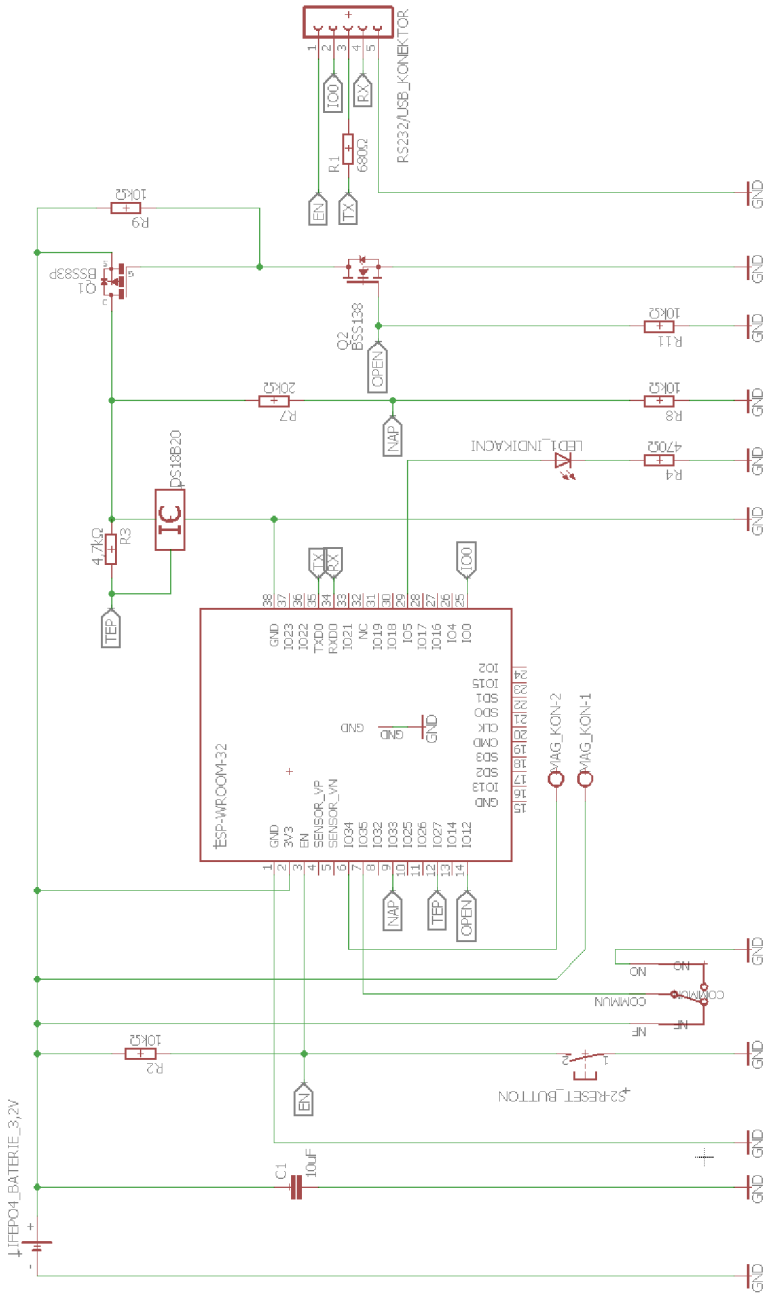
A SCHÉMA ZAPOJENÍ

A.1 Celkové schéma zapojení



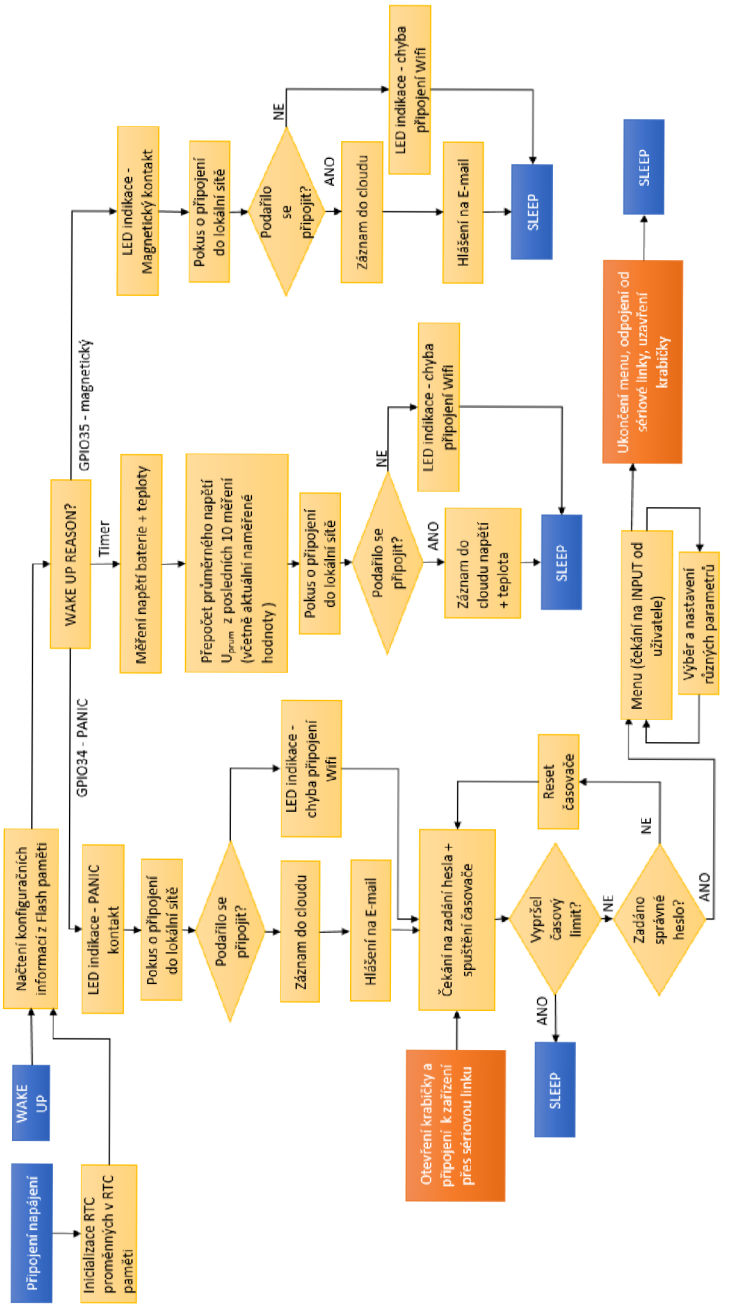
Obr. A.1: Celkové schéma zapojení

A.2 Upravené schéma pro desku plošných spojů



Obr. A.2: Upravené schéma pro desku plošných spojů

B BLOKOVÝ DIAGRAM ŘÍDÍCÍHO PROGRAMU



Obr. B.1: Blokový diagram řídícího programu

C OBSAH PŘILOŽENÉHO CD

Na přiloženém CD se nachází zdrojový kód ve formě .txt souboru a také jako .zip složka, kterou lze importovat přímo do vývojového prostředí Atom.io s Platform.io pluginem. Software byl testován na verzi atom.io 1.26.1 x64 a platform.io na verzi 3.5.3b5. Dále jsou přiloženy soubory pro editor Eagle obsahující upravené schéma pro desku plošných spojů a samotný návrh desky plošných spojů. Byla použita verze Eagle 7.7.0 Standard. A poslední dva soubory obsahují návrh ochranné krabičky ve formátu .stl. Tyto soubory lze importovat do internetové aplikace TinkerCAD a návrh zobrazit. [18]

```
/ ..... kořenový adresář přiloženého CD
├── 1.Bakalářská_práce
│   └── Bakalářská Práce - Petr Ondráček.pdf
├── 2.Zdrojový_kód
│   ├── Atom.io+platform.io Řídící program.zip
│   └── Zdrojový kód.txt
├── 3.Eagle
│   ├── Eagle - Deska plošných spojů DPS.brd
│   └── Eagle - Upravené schéma pro DPS.sch
└── 4.TinkerCAD
    ├── TinkerCAD - Druhý a třetí díl.stl
    └── TinkerCAD - První díl.stl
```