UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2020 Filip Prakesch

Univerzita Pardubice
fakulta Elektrotechniky a Informatiky

Záznamník stavu prostředí

Bakalářská práce

2020 Filip Prakesch

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze 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, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 1. 5. 2020

Filip Prakesch

PODĚKOVÁNÍ

Zde chci poděkovat všem, kteří mi pomohli, poradili a podpořili mě vytváření této práce. Děkuji vedoucímu práce Pavlu Rozsívalovi za jeho rady a trpělivost, dále svým kamarádům Filipu Tererovi, Filipu Svobodovi a Petru Skálovi.

ANOTACE

Tato bakalářská práce se zabývá návrhem a realizací záznamníku stavu prostředí pro vnitřní prostory a ukládání dat, které záznamník produkuje. Teoretická část se zaměřuje na popis internetu věcí (IoT), teorie k rozhraní I2C a dále na popis čidel a výběru desky pro dané řešení. V praktické části je popsán kód.

KLÍČOVÁ SLOVA

Internet věcí, I2C, Záznamník stavu prostředí, Adafruit IO, MQTT

TITLE

Environmental data logger

ANNOTATION

This bachelor thesis focuses on design and realization of environmental data logger for indoor use. The theoretical section focuses on Internet of Things (IoT), theory of I2C bus, description of sensors and choice of a board for this implementation. Practical section of the thesis describes the code.

KEYWORDS

Internet of Things, I2C, Enviromental data logger, Adafruit IO, MQTT

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

ACK	Acknowledgement
AWS	Amazon Web Services
DDR	Double data rate
ESP32	Espressif Systems procesors
GPIO	General-purpose input/output
I2C	Inter-Integrated Circuit
I2S	Inter-Integrated Circuit Sound
IDE	Integrated Development Environment
IO	Input/output
IoT	Internet of Things
IP	Internet Protocol
M2M	Machine to machine
MIT	Massachusetts Institute of Technology
MQTT	Message Queuing Telemetry Transport
PPB	Parts per billion
PPM	Parts per million
PWM	Pulse-width modulation
QoS	Quality of service
RAM	Random-access memory
RS232	Recommended Standard 232
SCL	Seriál clock
SD	Secure Digital
SDA	Seriál data
SPI	Serial Peripheral Interface
SRAM	Static random-access memory
UART	Universal Asynchronous Receiver/Transmitter
WIFI	Wireless networking technology

ÚVOD

V domácnostech, továrnách či ve vzdělávacích institucích se můžeme setkat s přístroji, které monitorují tlak, vlhkost a teplotu. Jedná se o takzvané záznamníky stavu prostředí. Tyto přístroje však mají většinou právě jen tyto tři senzory. Přitom pro oblasti, kde se vyskytuje více lidí je zajímavé sledovat i hodnoty oxidu uhličitého, jiných plynů a složek ovzduší. Ty indikují kvalitu ovzduší a jejich regulovatelnost. Tyto hodnoty jsou podkladem pro zdravější prostředí.

Cílem této bakalářské práce je navrhnout a realizovat záznamník stavu prostředí pro vnitřní prostor, který bude monitorovat tyto parametry, bude je ukládat na záznamové médium a na internet. Data bude tedy možné sledovat odkudkoli z přehledných grafů, nebo ze záznamového média. Tato práce by v budoucnu mohla být implementovatelná do většího systému, například pro řízení větrání.

První kapitola teoretické práce je zaměřena na koncept internetu věcí (IoT). Je zde uveden popis a využití této služby.

Druhá kapitola je zaměřena na popis sběrnice I2C, která je v této práci využívána pro přenos dat od senzorů k mikroprocesoru. Je zde popsána její funkce a parametry.

Třetí kapitola se zaměřuje na výběr platformy, kolem které je řešení vybudováno. Jsou zde stručně popsány parametry vybraných platforem.

Čtvrtá kapitola se zaměřuje na popis měřených veličin, včetně rešerše vybraných senzorů.

Pátá kapitola je zaměřena na serverovou službu Adafruit IO. Je tu zmíněn protokol MQTT, který je využíván pro posílání dat z platformy na server. Dále je zde uveden podrobný návod na instalaci a nastavení této služby, do podoby, kdy je možné sledovat přijaté hodnoty.

Šestá kapitola je zaměřena na návrh záznamníku. Jsou zde ukázány jednotlivé fyzické komponenty řešení a popis funkce celého návrhu.

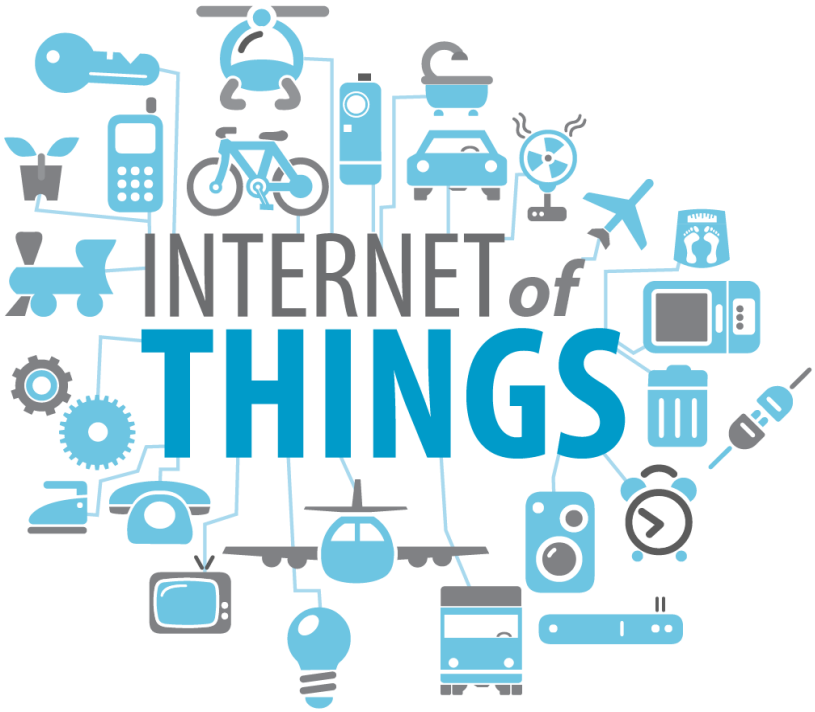
Další kapitoly se věnují části praktické.

Sedmá kapitola je zaměřena na kód. Jsou zde vysvětleny všechny části, ze kterých se kód skládá.

Osmá kapitola obsahuje naměřené hodnoty v rozmezí jednoho dne.

1 IoT

IoT, tedy internet of things (internet věcí), je souhrn různých zařízení, která bez zásahu lidí, dokáží posílat data přes internet k zajištění zjednodušení každodenního života. První zmínka o IoT je z roku 1999, kdy ji použil spoluzakladatel společnosti Auto-ID na MIT Kevin Ashon na prezentaci Procter & Gamble. Samotná myšlenka je však starší a sahá do 70. let minulého století a první aplikace přišla v 80. letech minulého století, kdy v kampusu Carnegie Mellon University byl instalován automat na colu, kdy pomocí internetu šlo zjistit, zda jsou nápoje v něm vychlazené, či ne. IoT se vyvinulo z M2M komunikace (komunikace od stroje ke stroji), kde stroje pomocí sítě komunikovali se serverem, který je řídil a sbíral data. [1]



Obrázek 1: Ilustrace vzhledu IoT

[2]

1.1 Funkce

IoT se skládá ze zařízení schopných připojit se na internet. Tato zařízení jsou často spojena do větších celků, kdy každá část zastává svou úlohu, jako například sběr dat. Tato data jsou dále využita jinými částmi, například pro regulaci či zabezpečení. [1]

Příkladem funkce IoT může být například již zmíněný záznamník stavu prostředí. Ten ve větším celku může fungovat tak, že získá ze senzorů data, ty upraví do vyžadované formy a pošle je do další části, například k termostatu, který tato data vyhodnotí a na jejich základě sníží nebo zvýší teplotu v místnosti, aby byla dosažena požadovaná hodnota. To vše bez nutnosti zásahu člověka. [1]

1.1.1 Výhody

Mezi výhody IoT patří dostupnost dat ze zařízení kdykoli a kdekoli, kde je přístup k internetu. Dokáže zlepšit komunikaci mezi připojenými zařízeními. Využití IoT v průmyslu dokáže pomoci s optimalizací výrobních procesů a tím snížit náklady na výrobu. Automatizace způsobená IoT, zvyšuje kvalitu služeb a redukuje nutnost zásahů člověka do výrobního procesu, tím dále redukuje možnost vzniku chyb způsobených lidskou činností. [1]

1.1.2 Nevýhody

Kvůli neustálému nárůstu zařízení připojených na internet, vzniká problém s ubývajícím počtem unikátních adres protokolu IPv4[1]. Tento problém se v dnešní době řeší postupným převáděním zařízení na protokol IPv6[2]. Vysoké počty zařízení zvyšují riziko ztráty zabezpečení a produkují velké množství dat, které negativně ovlivňují datové infrastruktury. Protože IoT nemá pevně stanovený standard je komunikace mezi zařízeními od jiných výrobců obtížná. [1]

1.2 Příklady použití

IoT pomáhá v domácnostech, průmyslu či v obchodu, k automatizaci a zjednodušení každodenního života.

1.2.1 Chytré domácnosti

Pod tímto pojmem se dá představit velká řada věcí. Od chytrého zámku na vchodových dveřích, až po systém celkové kontroly elektroniky v domě, který zahrnuje například celkové zabezpečení domu, regulace teploty, automatické žaluzie, osvětlení, sledování a regulace spotřeby.

Hlavní důraz je kladen na ulehčení života a zefektivnění spotřeby elektrické energie, vody, plynu, a dalších komodit, kdy jejich regulace nám může pomoci výrazně snížit náklady na provoz domácnosti. [1]

[1] https://www.techopedia.com/definition/5367/internet-protocol-version-4-ipv4
[2] https://searchnetworking.techtarget.com/definition/IPv6-Internet-Protocol-Version-6

1.2.2 Zdravotnictví

Ve zdravotnictví pomáhá IoT, při podrobnějším monitorování pacienta, za pomoci analýzy nashromážděných dat. V nemocnicích se IoT používá k zefektivnění managementu farmaceutických a lékařských nástrojů. [1]

1.2.3 Chytrá města

Zde se hlavně uplatňuje IoT v dopravě, kdy sleduje a řídí dopravu a tím předchází například zácpám, či pomáhá korigovat dopravu pro ulehčení průjezdu zásahových vozidel. Dále monitoruje energetickou a enviromentální stopu města. [1]

1.2.4 Průmysl

Největší využití pro IoT je v průmyslu. Kontrola kvality výroby se dnes velmi spoléhá na data z měřících senzorů. Řízení kvality ovzduší může mít pozitivní dopad na produktivitu pracovníků. Postupná automatizace výrobních procesů velmi čerpá z přínosů IoT. [1]

2 I2C

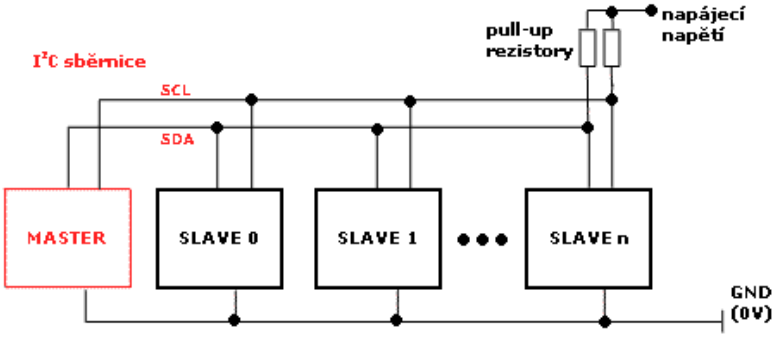
I2C je zkratka pro Internal – Integrated – Circuit Bus, tedy interní, integrovaná sběrnice. Byla vyvinuta v počátku osmdesátých let minulého století pro komunikaci mezi komponenty na jedné desce. [3]

Je řízena hodinovými signály podobně jako sběrnice SPI[3]. Data jsou díky tomu přenášeny synchronně. Na rozdíl od sběrnice SPI, I2C funguje pouze poloduplexně. V jeden okamžik tedy může vysílat pouze jedno zařízení. Přijímat však mohou všechny ostatní, které jsou připojeny na stejné sběrnici. [3]

Mezi hlavní výhody oproti ostatním sběrnicím je jednoduchost, jsou zapotřebí pouze dva signálové vodiče, to snižuje nároky na počet vstupně-výstupních pinů a zjednodušuje zapojení. První vodič je SDA, tedy serial data, který slouží pro obousměrný přenos dat. Druhý vodič je SCL, tedy serial clock, přes který zařízení určené jako master posílá hodinový signál, tento signál není striktně omezený jako například u RS232[4]. Oba vodiče dále musí být připojeny přes takzvané pull up rezistory k napájení. Ty slouží ke zvednutí napětí obou vodičů na úroveň logické jedničky, což je pro sběrnici normou stanovený klidový stav, ve kterém může setrvat po neomezenou dobou. [3]

[3] https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html#
[4] http://www.bb-elec.com/Learning-Center/All-White-Papers/Serial/What-Is-RS-232.aspx

I2C používá pro komunikaci jednoduché rozhraní Master – Slave. Kdy každé Slave zařízení má svou vlastní unikátní adresu. I2C podporuje Multi-Master mód a má dva základní mody adresace, 7bitovou a 10bitovou. [3]

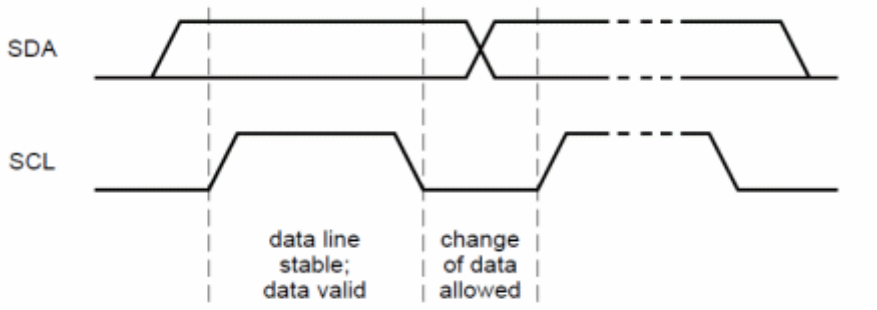


Obrázek 2: Sběrnice I2C

[4]

2.1 Funkce

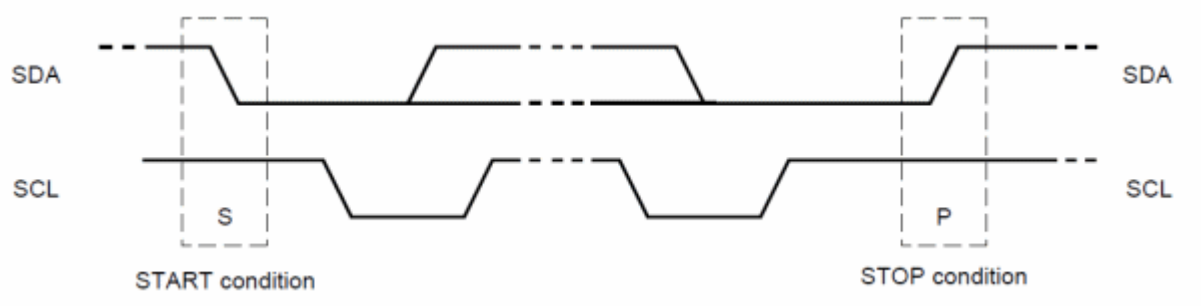
Funkce sběrnice I2C spočívá v tom, že v každém hodinovém impulzu se vyšle jeden bit. Signál z SDA se může změnit pouze pokud je SCL v logické nule. Pokud je SCL v logické jedničce signál je stálý. [5]



Obrázek 3: Příklad vysílání na sběrnici I2C

[5]

Každá komunikace vyvolána zařízením nastaveným jako master obsahuje START a STOP stav. Kdy po vyslání START stavu, který je identifikovatelný jako kombinace, kdy SCL je v logické jedničce a SDA se přepíná z logické jedničky na logickou nulu, je sběrnice brána jako používaná a nejde ji použít jiným zařízením do doby, dokud se nevyšle STOP stav, který je identifikovatelný jako kombinace, kdy SCL je v logické jedničce a SDA je přepíná z logické nuly na logickou jedničku. [5]



Obrázek 4: Příklad vzhledu začátku a konce vysíláni na sběrnici I2C

[5]

2.1.1 Standardní rychlosti

Sběrnice I2C stanovuje několik standardizovaných rychlostí.

Označení	Přenosová rychlost
Low speed mode	10 kbps
Standard mode	100 kbps
Fast mode	400 kbps
Fast mode	1 Mbps
High speed mode	3,4 Mbps

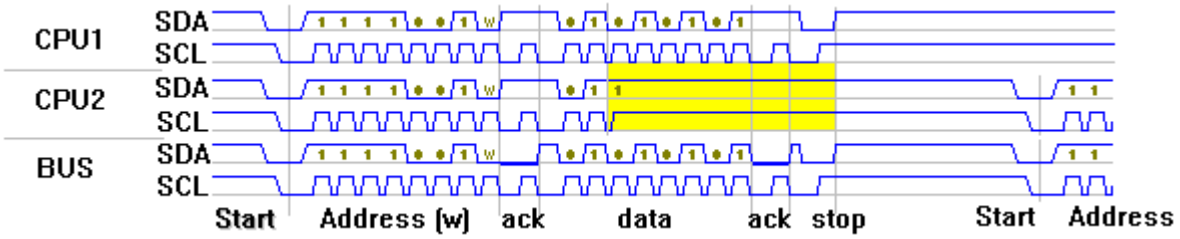
Tabulka 1: Standardní rychlosti sběrnice I2C

[3]

2.2 Multi – Master mód

Sběrnice I2C může fungovat v modu Multi – Master módu, na sběrnici je možné připojit více zařízení, které mají stejnou roli Master. Tyto zařízení se však musí řídit arbitrážní logikou, aby nedošlo ke kolizím signálů. [6]

Arbitrážní logika u sběrnice I2C funguje tak, že zařízení, které jsou v módu Master mohou sledovat probíhající komunikaci pro sběrnici. Jmenovitě sledují START a STOP stav. Dále zařízení testují, zda sběrnice je zrovna v jedničce či nule. Pokud zařízení změní hodnotu na jedna musí zůstat v jedničce, což také zkontroluje. Pokud však zjistí že sběrnice je na nule ztratí arbitráž do té doby, než může znovu zkusit vysílat. [6]



Obrázek 5: Příklad vysílání v Multi – Master módu

[7]

Na obrázku můžeme pozorovat příklad vysílání v Multi – Master módu. Obě zařízení chtějí vysílat na stejnou adresu. Jakmile začnou vysílat a začnou se ve zprávě lišit, zařízení, co chce vysílat jedna ztratí arbitráž a vysílá pouze druhé zařízení. [7]

2.3 7-bit adresace

Každé zařízení na sběrnici I2C má přidělenou unikátní adresu, pro jednoznačné rozpoznání. Při menším počtu připojených zařízení se používá 7bitová adresace, která se přenáší v jednom bajtu, kdy poslední bit indikuje, zda půjde o zápis či čtení dat. Při této adresaci je teoreticky možné adresovat až 128 unikátních zařízení, prakticky se jedná o menší číslo z důvodu rezervace vybraných adres. [5]

Po zahájení komunikace se vyšle adresa na všechna zařízení nastavena do módu Slave, po přenosu všech osmi bitů, si každé zařízení porovná vyslanou adresu se svojí, poté vybrané zařízený pošle potvrzující bit ACK. Po obdržení tohoto bitu zařízení nastavené jako Master začne komunikovat. [5]

2.4 10-bit adresace

Při této adresaci můžeme využít teoreticky až 1024 adres. Opět použitelných je o něco méně. Při adresování deseti bity musíme vysílat adresu ve dvou bajtech. To způsobuje neefektivitu. [5]

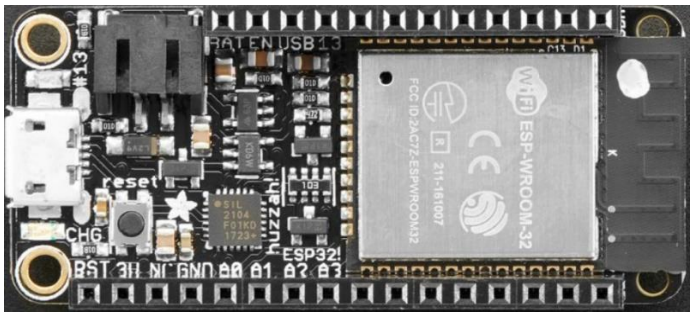
V prvním bajtu jsou uloženy dva nejvyšší bity adresy ve tvaru 11110xx0, kde x znázorňuje pozice nejvyšších bitů. V druhém bajtu je zbytek adresy. Tento postup funguje pro komunikace z Master do Slave, pokud bychom chtěli, aby Slave poslal data, museli bychom použít tzv. opakovaný start, kdy se vyslání adresy opakuje vysláním prvního bajtu, kdy poslední bit je negovaný a druhý bajt je přiřazen informaci od Slave. [5]

3 Výběr platformy

V dnešní době můžeme na trhu najít nepřeberné množství platforem vhodných pro aplikaci, jako je právě záznamník stavu prostředí. Při výběru platformy bylo nahlíženo na dvě důležité vlastnosti. Platforma musí podporovat standardy IEEE 802.11[5] pro bezdrátovou komunikaci a platforma má podporu, nebo již u sebe má rozhraní na SD kartu.

3.1 HUZZAH32

Tato platforma využívá procesoru z architektury ESP32. Má integrovanou 520KB SRAM. Wifi čip podporující standardy 802.11 b/ g/ n a Bluetooth mód. Má třikrát UART rozhraní, třikrát SPI rozhraní a dvakrát I2C rozhraní. Má podporu SD karty. Dá se programovat pomocí jazyku Micropython, nebo jazyku Wiring. [8]



Obrázek 6: Deska HUZZAH32

[8]

3.2 Linklt Smart 7688

Tato platforma využívá systém na bázi Linux OpenWrt. Má 128 MB DDR2 RAM a 32 MB Flash paměť. Wifi čip podporující standardy 802.11 b/ g/ n. 22 pinů podporujících GPIO, I2C, I2S, SPI, UART, PWM a Ethernet. Na spodní straně má slot na SD kartu. Dá se programovat pomocí jazyku Wiring, Java, nebo C. [9]

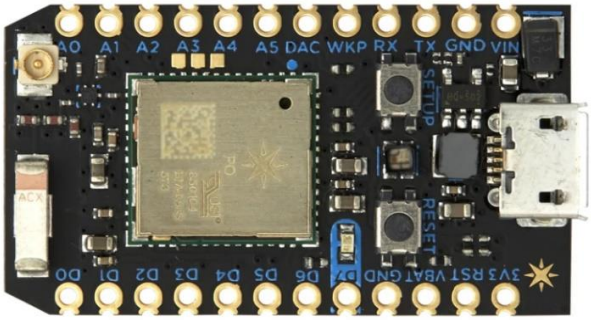
Obrázek 7: Deska Linklt Smart 7688

[9]

[5] https://www.networkworld.com/article/3238664/80211-wi-fi-standards-and-speeds-explained.html

3.3 Particle PHOTON

Tato platforma využívá procesoru z architektury ARM Cortex M3. Má 128 KB RAM a 1 MB Flash paměť. Wifi čip podporující standardy 802.11 b/ g/ n. Operační systém pracující v reálném čase FreeRTOS. 18 pinů, kdy dva jsou přiřazeny SPI, jeden I2S, jeden I2C, devět PWM. Má přístup ke cloud uložišti, které obsahuje funkce jako management zařízení, Firmware update na dálku, nebo nástroje pro vývojáře. Dá se programovat v jazyce Wiring, nebo Micropython. [10]



Obrázek 8: Deska PHOTON WIFI

[10]

3.4 Linklt 7697

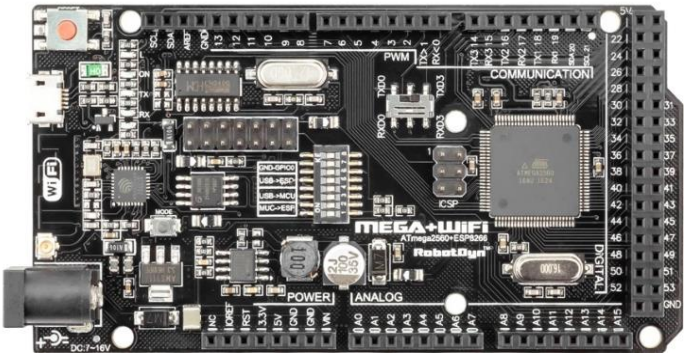
Tato deska využívá procesoru z architektury ARM Cortex M4. Má 352 KB RAM a 4 MB Flash paměť. Wifi čip podporující standardy 802.11 b/ g/ n a Bluetooth mód. Má dvakrát UART rozhraní, jedenkrát SPI rozhraní, jedenkrát I2C rozhraní a osmnáctkrát PWM rozhraní. Dá se programovat pomocí jazyku Wiring, Java, nebo C. [11]

Obrázek 9: Deska Linklt 7697

[11]

3.5 MEGA + Wifi R3 ATmega2560 + ESP8266

Tato deska využívá procesoru z architektury ATmega. Má 256 KB RAM a 32 MB Flash paměť. Wifi čip podporující standardy 802.11 b/ g/ n. Má 54 digitálních a 16 analogových pinů. Dá se programovat pomocí jazyku C nebo C++. [12]



Obrázek 10: Deska MEGA R3

[12]

3.6 Omega2+

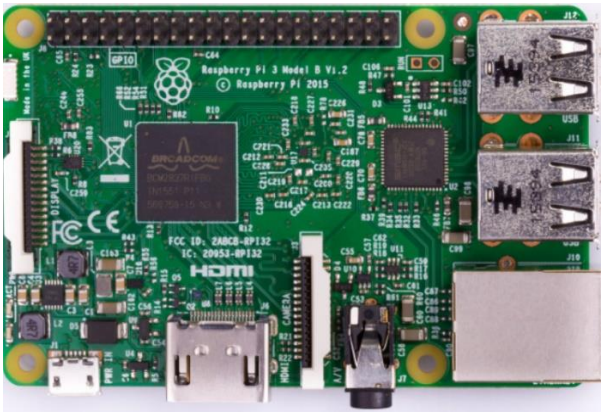
Tato deska využívá systém na bázi Linux. Má 128 MB RAM a 32 MB Flash paměť. Wifi čip podporující standardy 802.11 b/ g/ n a podporuje ethernet. Má dvakrát UART rozhraní, jedenkrát SPI rozhraní, jedenkrát I2C rozhraní, jedenkrát I2S rozhraní a dvakrát PWM rozhraní. Na spodní straně má slot na SD kartu. Dá se programovat například pomocí jazyku C, C++, PHP, nebo Python. [13]

Obrázek 11: Deska Omega2+

[13]

3.7 Raspberry Pi 3

Tato deska využívá architektury ARMv8. Má 1 GB SDRAM. Integrovaný wifi čip podporující standardy 802.11n, 10/100 Mbit/s Ethernet a Bluetooth 4.1. Má 17 GPIO pinů. Má slot na SD kartu, kterou využívá jako interní paměť. [14]



Obrázek 12: Deska Raspberry PI 3

[14]

3.8 WiPy 3.0

Tato deska využívá procesoru z architektury ESP32. Má 4 MB RAM a 8 MB Flash paměť. Wifi čip podporující standardy 802.11 b/ g/ n. Má dvakrát UART rozhraní, dvakrát SPI rozhraní, I2C rozhraní a I2S rozhraní. Má podporu SD karty. Dá se programovat pomocí jazyku Micropython. [15]

Obrázek 13: Deska WiPy 3.0

[15]

WiPy 3.0 byla vybrána pro řešení návrhu z důvodu, že splňuje kritéria, která byla stanovena na začátku této kapitoly. Obsahuje wifi čip podporující standardy 802.11 b/ g/ n. Protože nemá na sobě žádný konektor na napájení, musí se k této platformě dokoupit, nebo vyrobit, rozšiřující deska. Deska zkonstruovaná přímo k této platformě má však již v základu na sobě slot na SD kartu, což velmi zjednodušuje návrh. Cena se pohybuje v rozmezí ostatních

platforem. Hlavní výhodou WiPy oproti výše zmíněným platformám je velmi dobře zpracovaná dokumentace, která obsahuje příklady použití různých nástrojů WiPy v kódu.

4 Senzory

Senzor je zařízení, které reaguje na vnější vliv, měřenou velečinu, stav prostředí. Převádí tuto veličinu na výstupní veličinu, informaci. Senzory byly vybírány na základě předpokladu, že mohou komunikovat po sběrnici I2C.

4.1 Měřené veličiny

Záznamník měří základní veličiny, které můžeme běžně měřit v uzavřených prostorech, teplotu, tlak, vlhkost. Dále však měří veličiny, které indikují kvalitu ovzduší, oxid uhličitý a přítomnost těkavých organických sloučenin.

4.2 Teplota

Teplota měří vnitřní energii systému. Můžeme ji měřit pomocí teploměru, nebo pomocí kalorimetru[6]. Základní jednotky teploty je stupeň Celsia (značeno °C), stupeň Fahrenheita (značeno °F) a stupeň Kelvina (značeno K). [16]

4.2.1 MAX31850TATB+

Senzor MAX31850TATB+, měří teplotu. Jedná se o 1-Wire zařízení, kdy se senzor dokáže napájet přes datovou sběrnici a tedy využít pouze dva vodiče, jeden pro zem a druhý společný pro data a napájení. Měří ve velkém rozsahu, ale je velmi nepřesný.

Označení	MAX31850TATB+
Napájecí napětí	3.0 až 3.7 V
Rozhraní	1-Wire
Rozsah	-270 až 400 °C
Přesnost	+/- 4 °C

Tabulka 2: Vybrané hodnoty senzoru MAX31850TATB+

[17]

[6] https://www.greelane.com/sk/science-tech-math/veda/definition-of-calorimeter-in-chemistry-604397/

4.2.4 ADT7420UCPZ-R2

Senzor ADT7420UCPZ-R2, měří teplotu.

Označení	ADT7420UCPZ-R2
Napájecí napětí	2.7 až 5.5 V
Rozhraní	I2C
Rozsah	-20 až 105 °C
Přesnost	+/- 0.25 °C

Tabulka 3: Vybrané hodnoty senzoru ADT7420UCPZ-R2

[18]

4.2.4 SMT172-220

Senzor SMT172-220, měří teplotu.

Označení	SMT172-220
Napájecí napětí	2.7 až 5.5 V
Rozhraní	PWM
Rozsah	-45 až 130 °C
Přesnost	+/- 0.25 °C

Tabulka 4: Vybrané hodnoty senzoru SMT172-220

[19]

4.2.4 DS18S20

Senzor DS18S20, měří teplotu je to 1-Wire zařízení. Tento senzor byl vybrán pro návrh.

Označení	DS18S20
Napájecí napětí	3.0 až 5.5 V
Rozhraní	1-Wire
Rozsah	-55 až 125 °C
Přesnost	+/- 0.5 °C

Tabulka 5: Vybrané hodnoty senzoru DS18S20

[20]

4.3 Tlak

Tlak definuje sílu vykonanou na jednotku plochy. Jednotkou tlaku je Pascal (značeno Pa). Senzory na měření tlaku, měří většinou pouze absolutní tlak. Tento tlak obsahuje v sobě

atmosférický tlak, který je způsoben tíhou sloupce vzduchu nad plochou. Atmosférický tlak se mění s nadmořskou výškou, proto se mění i absolutní tlak. [21]

4.3.1 DPS422XTSA1

Senzor DPS422XTSA1, měří barometrický tlak.

Označení	DPS422XTSA1
Napájecí napětí	1.7 až 3.6 V
Typ	Barometrický
Rozhraní	I2C
Rozsah	30 až 120 kPa
Přesnost	+/- 300 Pa

Tabulka 6: Vybrané hodnoty senzoru DPS422XTSA1

[22]

4.3.2 2SMPB-02E

Senzor 2SMPB-02E, měří absolutní tlak.

Označení	2SMPB-02E
Napájecí napětí	1.71 až 3.6 V
Typ	Absolutní
Rozhraní	I2C
Rozsah	30 až 110 kPa
Přesnost	+/- 50 Pa

Tabulka 7: Vybrané hodnoty senzoru 2SMPB-02E

[23]

4.3.3 2SMPB-01-01

Senzor 2SMPB-01-01, měří absolutní tlak.

Označení	2SMPB-01-01
Napájecí napětí	2.25 až 3.6 V
Typ	Absolutní
Rozhraní	I2C
Rozsah	30 až 110 kPa
Přesnost	+/- 750 Pa

Tabulka 8: Vybrané hodnoty senzoru 2SMPB-01-01

[24]

4.3.4 MPL115A2

Senzor MPL115A2, měří absolutní tlak. Může měřit i teplotu. Tento senzor byl vybrán pro návrh.

Označení	MPL115A2
Napájecí napětí	2.375 až 5.5 V
Typ	Absolutní
Rozhraní	I2C
Rozsah	50 až 115 kPa
Přesnost	+/- 1 kPa

Tabulka 9: Vybrané hodnoty senzoru MPL115A2

[25]

4.4 Vlhkost

Vlhkost udává množství vodních par ve vzduchu. Většinou se vlhkost uvádí jako relativní. Je to procentuální vyjádření par ve vzduchu vztažené k maximální hodnotě nasycení vzduchu vodními parami při dané teplotě. Při 100% relativní vlhkosti, vzduch již nedokáže přijímat žádné vodní páry. [26]

4.4.1 HPP845E034R5

Senzor HPP845E034R5, měří vlhkost a teplotu.

Označení	HPP845E034R5
Napájecí napětí	1.5 až 3.6 V
Rozhraní	I2C
měření	vlhkost
Rozsah	0 až 100 %
Přesnost	+/- 2 %
Doba odezvy	5 s
měření	teplota
Rozsah	-40 až 125 °C
Přesnost	+/- 0.3 °C
Doba odezvy	10 s

Tabulka 10: Vybrané hodnoty senzoru HPP845E034R5

[27]

4.4.2 HDC1080DMBT

Senzor HDC1080DMBT, měří vlhkost a teplotu.

Označení	HDC1080DMBT
Napájecí napětí	2.7 až 5.5 V
Rozhraní	I2C
měření	vlhkost
Rozsah	0 až 100 %
Přesnost	+/- 2 %
Doba odezvy	15 s
měření	teplota
Rozsah	-40 až 125 °C
Přesnost	+/- 0.2 °C
Doba odezvy	-

Tabulka 11: Vybrané hodnoty senzoru HDC1080DMBT

[28]

4.4.3 SHT25

Senzor SHT25, měří vlhkost a teplotu.

Označení	SHT25
Napájecí napětí	2.1 až 3.6 V
Rozhraní	I2C
měření	vlhkost
Rozsah	0 až 100 %
Přesnost	+/- 1.8 %
Doba odezvy	8 s
měření	teplota
Rozsah	-40 až 125 °C
Přesnost	+/- 0.2 °C
Doba odezvy	30 s

Tabulka 12: Vybrané hodnoty senzoru SHT25

[29]

4.4.4 SHT31-DIS-B

Senzor SHT31-DIS-B, měří vlhkost a teplotu. Dovoluje využití rychlosti komunikace přes I2C až 1MHz. Je továrně zkalibrovány. Tento senzor byl vybrán pro návrh.

Označení	SHT31-DIS-B
Napájecí napětí	2.15 až 5.5 V
Rozhraní	I2C
měření	vlhkost
Rozsah	0 až 100 %
Přesnost	+/- 2 %
Doba odezvy	8 s
měření	teplota
Rozsah	-40 až 125 °C
Přesnost	+/- 0.2 °C
Doba odezvy	2 s

Tabulka 13: Vybrané hodnoty senzoru SHT31-DIS-B

[30]

4.5 Oxid uhličitý

Oxid uhličitý je plyn bez barvy a zápachu. Je to vedlejší produkt dýchání a spalování fosilních paliv. Je minoritní složkou atmosféry. Zelené rostliny využívají tento plyn k fotosyntéze. Při zvýšené koncentraci může mít vliv na produktivitu člověka. Oxid uhličitý se většinou vyjadřuje pomocí ppm (parts per milion), neboli v částicích na milion. Ve vnitřních prostorech je optimální mít hodnoty oxidu uhličitého pod 1000 ppm. Při překročení hodnoty 2000 ppm, již mohou nastat problémy s dýcháním. [31]

4.5.1 SCD30

Senzor SCD30, měří hodnoty oxidu uhličitého, dále měří teplotu a vlhkost.

Označení	SCD30
Napájecí napětí	3.3 až 5.5 V
Rozhraní	I2C, UART, PWM
Rozsah	0 až 10000 ppm
Přesnost	+/- 30 ppm +/- 3 %
měření	vlhkost
Rozsah	0 až 100 %
Přesnost	+/- 3 %
Doba odezvy	8 s
měření	teplota
Rozsah	-40 až 70 °C
Přesnost	+/- 0.4 °C
Doba odezvy	10 s

Tabulka 14: Vybrané hodnoty senzoru SCD30

[32]

4.5.2 T6613-5KC

Senzor T6613-5KC, měří hodnoty oxidu uhličitého.

Označení	T6613-5KC
Napájecí napětí	5 V
Rozhraní	I2C, UART
Rozsah	0 až 5000 ppm
Přesnost	+/- 30 ppm +/- 5 %

Tabulka 15: Vybrané hodnoty senzoru T6613-5KC

[33]

4.5.3 T6713

Senzor T6713, měří hodnoty oxidu uhličitého. Tento senzor byl vybrán pro návrh.

Označení	T6713
Napájecí napětí	4.5 až 5.5 V
Rozhraní	I2C, UART
Rozsah	0 až 2000 ppm
Přesnost	+/- 25 ppm +/- 3 %

Tabulka 16: Vybrané hodnoty senzoru T6713

[34]

4.6 Těkavé organické sloučeniny

Těkavé organické sloučeniny, jsou plyny z různých zdrojů, které zhoršují kvalitu ovzduší a mohou způsobit zdravotní potíže. Ve vnitřních prostorech je jejich koncentrace až pětkrát větší než ve venkovních. Mezi zdroje těchto sloučenin patří barvy, laky, čistící prostředky, pesticidy, uskladněné palivo a jiné automobilové produkty, náplně do tiskáren, lepící prostředky. [35]

4.6.1 SGP30

Senzor SGP30, měří hodnoty těkavých organických sloučenin. Měří hodnoty oxidu uhličitého, dokáže měřit hodnoty Vodíku a Ethanolu.

Označení	SGP30
Napájecí napětí	1.62 až 1.98 V
Rozhraní	I2C
Rozlišení TVOC	0 až 60000 ppb

Tabulka 17: Vybrané hodnoty senzoru SGP30

[36]

4.6.2 CCS811

Senzor CCS811, měří hodnoty těkavých organických sloučenin. Dokáže přepočítávat prvotní data na ekvivalentní celkové hodnoty těkavých organických sloučenin a oxidu uhličitého. Tento senzor byl vybrán pro návrh.

Označení	CCS811
Napájecí napětí	1.8 až 3.6 V
Rozhraní	I2C
Rozlišení TVOC	0 až 32768 ppb

Tabulka 18: Vybrané hodnoty senzoru CCS811

[37]

5 Cloudová služba Adafruit IO

Adafruit.IO je cloud služba od společnosti Adafruit. Je zaměřena na ukládání a správu dat z přístrojů, které spadají do kategorie IoT. Tato služba dokáže promítat data v reálném čase, poskytnout vzdálené řízení a propojit projekty s více službami, jako například propojení Adafruit IO s Gmail, kdy data, nebo upozornění, jsou odeslána přímo na email. V základu se jedná o službu zdarma, kdy je k dispozici 30denní uložení dat, 30 data pointů za minutu a až 5 feeds. Placená verze funguje na měsíčním předplatném, kdy ukládá data na 60 dní a je zde 60 data pointů za minutu. [38]

Další podobné služby jako Adafruit IO jsou například ThinkSpeak[7], Cayenne[8], nebo IoT GURU[9]. ThinkSpeak je služba propojena s programem Matlab[10], dovoluje tedy složitější analýzu dat, je zdarma a největší omezení oproti placené verzi je v počtu přenesených zpráv za rok, kdy v neplacené verzi můžeme odeslat 3 miliony zpráv. Cayenne funguje na systému drag and drop widgetů. IoT GURU, se primárně zaměřuje na mobilní platformu. Další služby jsou zaměřeny na velké společnosti, nabízí velké množství možností, ale jsou velmi drahé. Google cloud IoT[11], Microsoft Azure IoT[12], AWS IoT[13], nabízejí velkou ochranu před ztrátou dat, využívají umělou inteligenci a úzce spolupracují s open source externími programy pro analýzu a zpracování dat.

[7] https://thingspeak.com/
[8] https://developers.mydevices.com/cayenne/features/
[9] https://iotguru.live/
[10] https://www.mathworks.com/products/matlab.html
[11] https://cloud.google.com/solutions/iot
[12] https://azure.microsoft.com/en-us/
[13] https://aws.amazon.com/iot/

5.1 MQTT

MQTT (Message Queuing Telemetry Transport) je jednoduchý protokol, zaměřený na komunikaci mezi klienty přes centrální bod (Broker). Jeho jednoduchost a nenáročnost, jsou důvody proč se používá na zařízeních zařazených do skupiny IoT. Pracuje na principu publish-subscribe (vydej-odebírej). Publisher odešle zprávu do centrálního bodu (Broker). Ten je uloží a pošle dál. Zprávy jsou v brokeru ukládány do tzv. topic (témat). Zprávy z jednotlivých témat jsou poslány na klienta, který dané téma odebírá. V různých tématech může být jeden klient subscriber i Publisher. [39]

Obsah zpráv není pevně daný. Jsou to data v binární podobě a jejich velikost je omezena na 256 MB. MQTT kvůli minimalizaci přidává k datům pouze minimum zabezpečujících dat. Zpráva se skládá ze dvou hlaviček (jedna povinná, druhá volitelná), vlastních dat, a QoS (Quality of Service). [39]

5.1.1 QoS úrovně

Tyto tři úrovně udávají, jakým způsobem se bude příjem zpráv potvrzovat.

1) Nejnižší úroveň udává odeslání zprávy bez potvrzení, při této úrovni není zaručeno, že se zpráva doručí, nebo bude doručena v pořádku. [39]

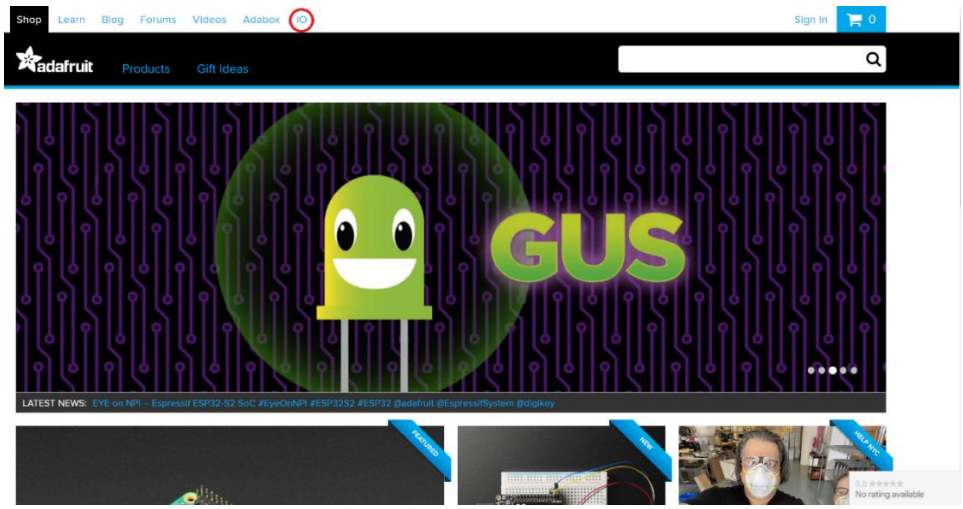
2) Prostřední úroveň udává odeslání zprávy s jedním potvrzením, při této úrovni příjemce odešle potvrzení o přijmutí zprávy. [39]

3) Nejvyšší úroveň udává, že se zpráva odešle právě jen jednou, při této úrovni je zaručeno pomocí dvou páru paketů, že se zpráva odešle pouze jednou. [39]

V návrhu záznamníku je použita knihovna zajišťující MQTT protokol, pro odesílání dat z platformy do služby Adafruit IO.

5.2 Instalace

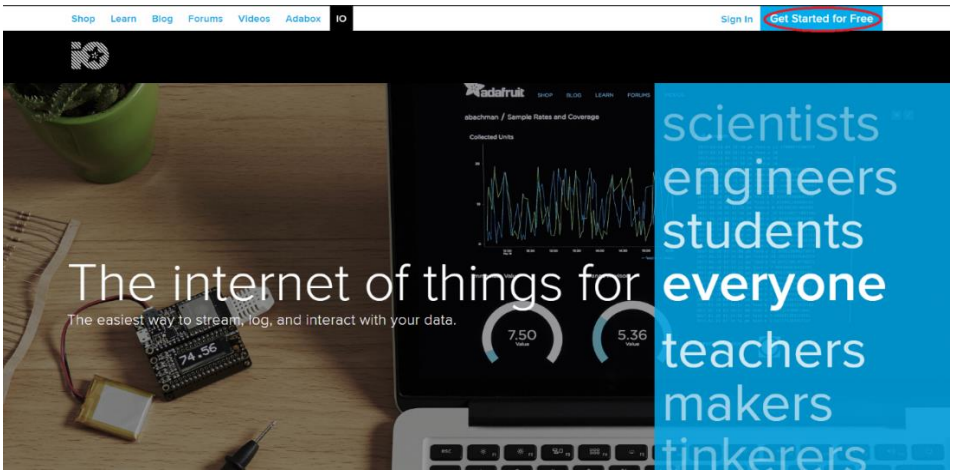
V internetovém prohlížeči přejdeme na webovou stránku https://www.adafruit.com/. Zde klikneme na poslední položku menu nahoře – „IO“.



Obrázek 14: Instalace cloudové služby: hlavní strana

[38]

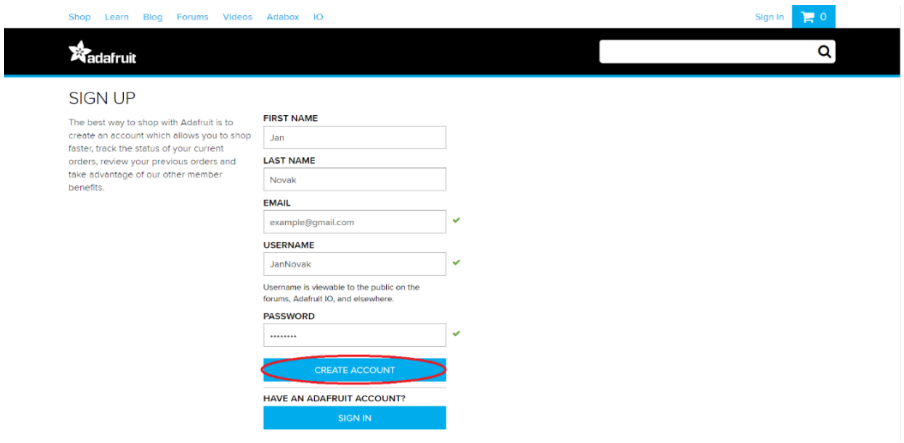
Zde klikneme na tlačítko „Get Started for Free“, které se nachází v pravém horním rohu stránky.



Obrázek 15: Instalace cloudové služby: hlavní strana IO

[38]

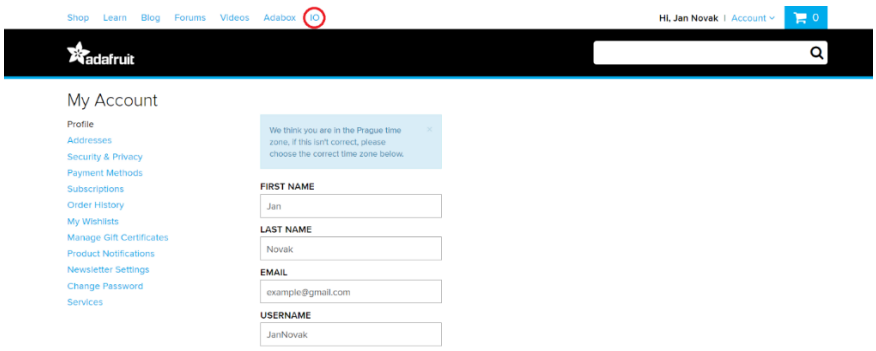
Na registrační stránce vyplníme všechny potřebné údaje, k vytvoření účtu a poté klikneme na tlačítko- „Create Account“.



Obrázek 16: Instalace cloudové služby: Vytvoření účtu

[38]

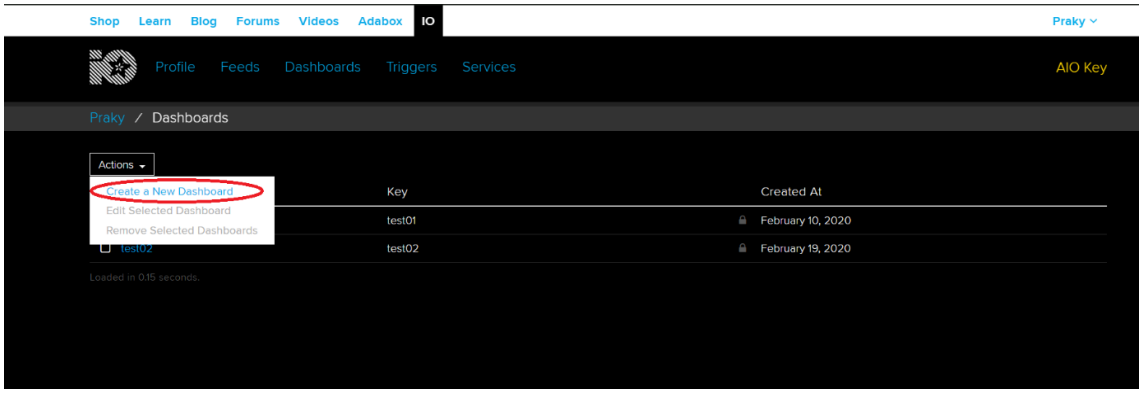
Po vytvoření účtu klikneme opět na tlačítko- „IO“, kdy budeme přesměrováni na hlavní stranu naší cloudové služby.



Obrázek 17: Instalace cloudové služby: přihlášení

[38]

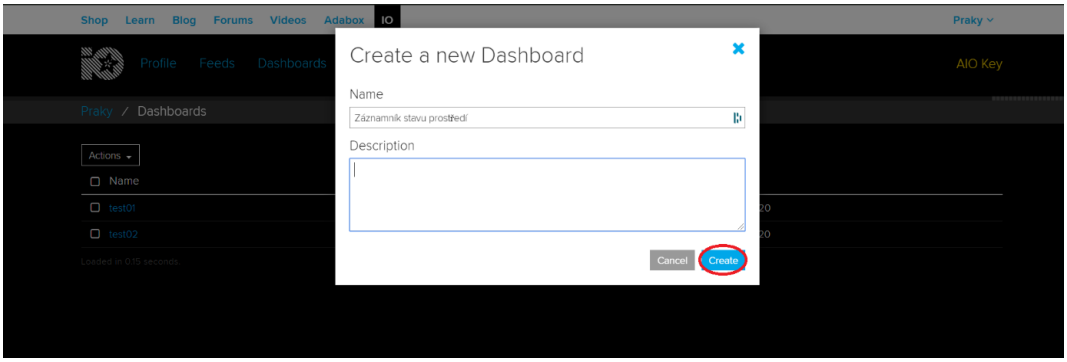
Na této stránce nalezneme všechny námi vytvořené projekty. Ve službě Adafruit IO jsou pojmenovány jako „Dashboards“. K vytvoření nového projektu klikneme na tlačítko „Actions“ a ze zobrazené nabídky si vybereme „Create a New Dashboard“.



Obrázek 18: Instalace cloudové služby: vytvoření projektu

[38]

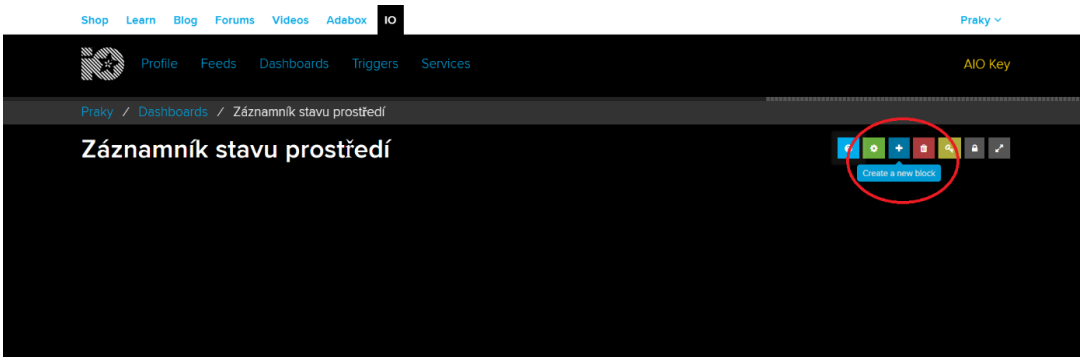
Projekt si pojmenujeme. Můžeme přidat popisek a v pravém dolním rohu okna zmáčkneme tlačítko - „Create“.



Obrázek 19: Instalace cloudové služby: vytvoření projektu 2

[38]

Ve vytvořeném projektu se nám na pravé straně objeví menu. Pomocí kterého můžeme přidávat, odebírat, nebo upravovat bloky, které využijeme při zobrazování naměřených hodnot.



Obrázek 20: Instalace cloudové služby: menu pro úpravu projektu

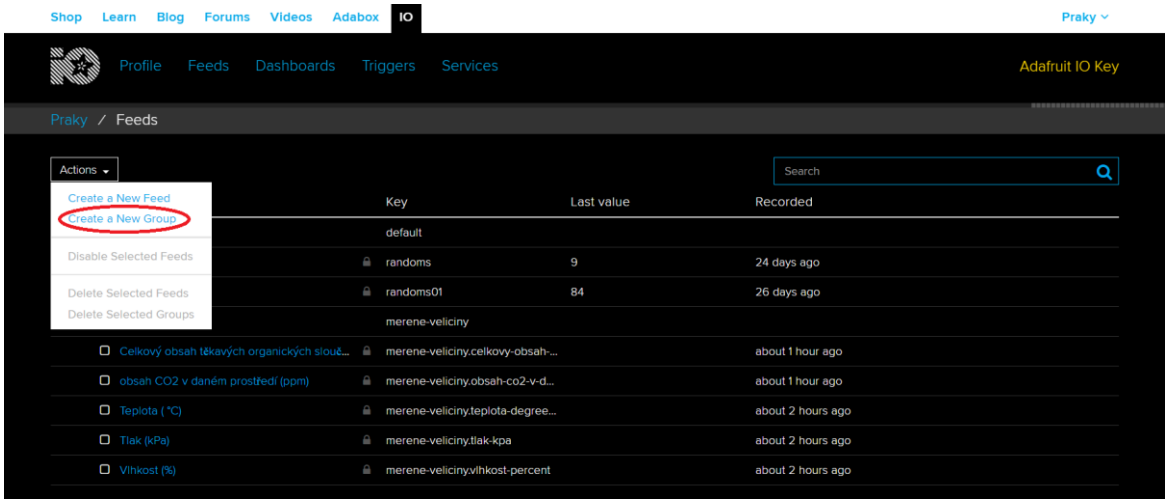
[38]

Před vytvořením bloku si musíme vytvořit složku do které budeme zasílat naměřená data. V službě Adafruit IO se tyto složky nazývají „Feeds“. Klikneme tedy v horní straně stránky na záložku Feeds. Otevře se nám okno, zde zaklikneme – „view all“.

Obrázek 21: Instalace cloudové služby: záložka pro Feeds

[38]

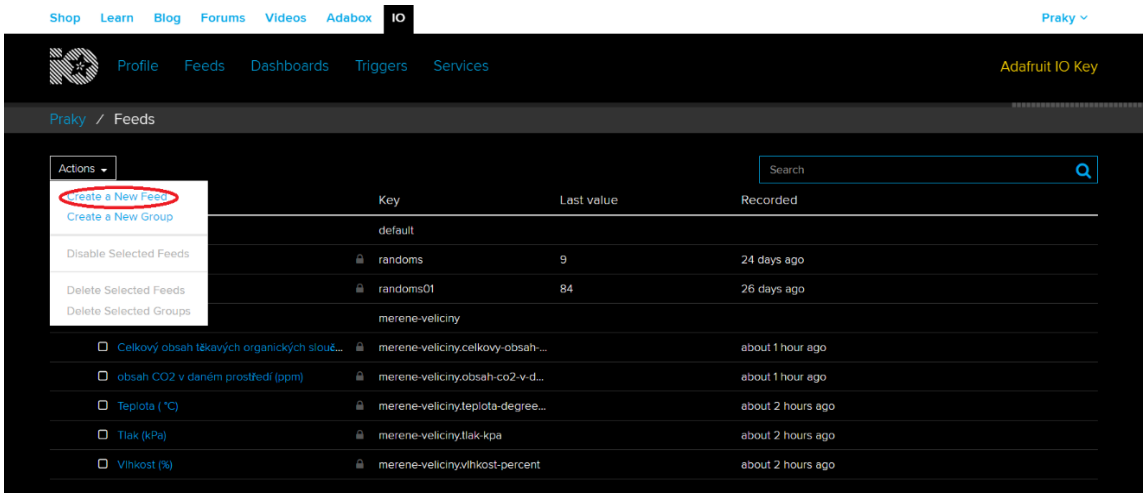
Zde uvidíme seznam všech námi vytvořených složek. Můžeme je řadit do skupin pro lepší přehlednost. Pro vytvoření nové skupiny klikneme na tlačítko „Actions“ a ze zobrazené nabídky vybereme – „Create New Group“.



Obrázek 22: Instalace cloudové služby: vytvoření skupiny

[38]

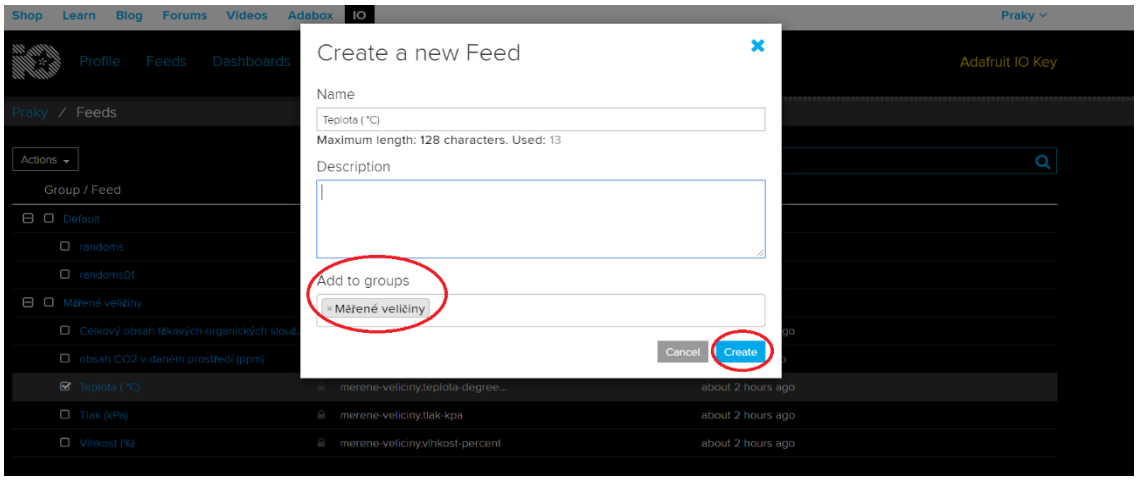
Tu si stejně jako při vytváření projektu pojmenujeme a dáme „Create“. Poté vytvoříme složku. Klikneme na tlačítko „Action“ a ze zobrazené nabídky vybereme - „Create New Feed“.



Obrázek 23: Instalace cloudové služby: vytvoření Feed

[38]

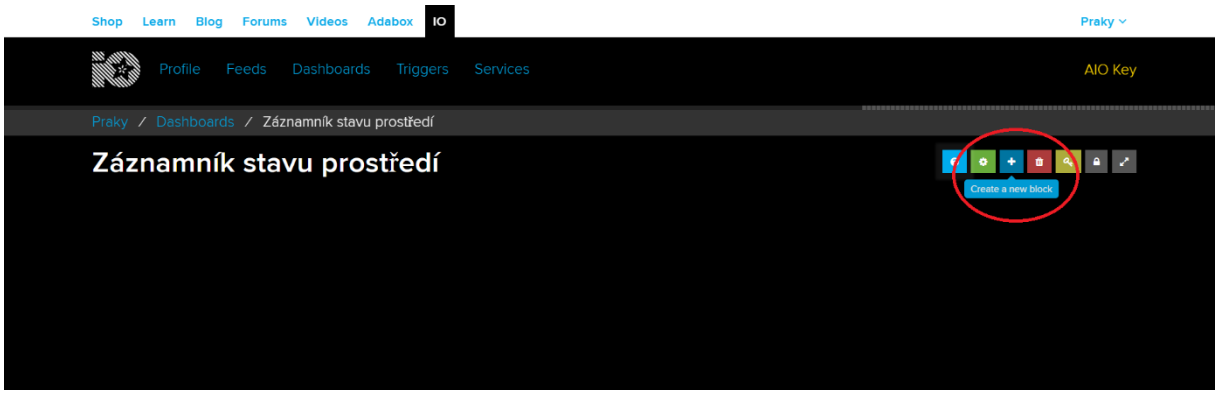
Zde si složku pojmenujeme, přiřadíme ji do námi vytvořené skupiny a dáme „Create“.



Obrázek 24: Instalace cloudové služby: vytvoření Feed 2

[38]

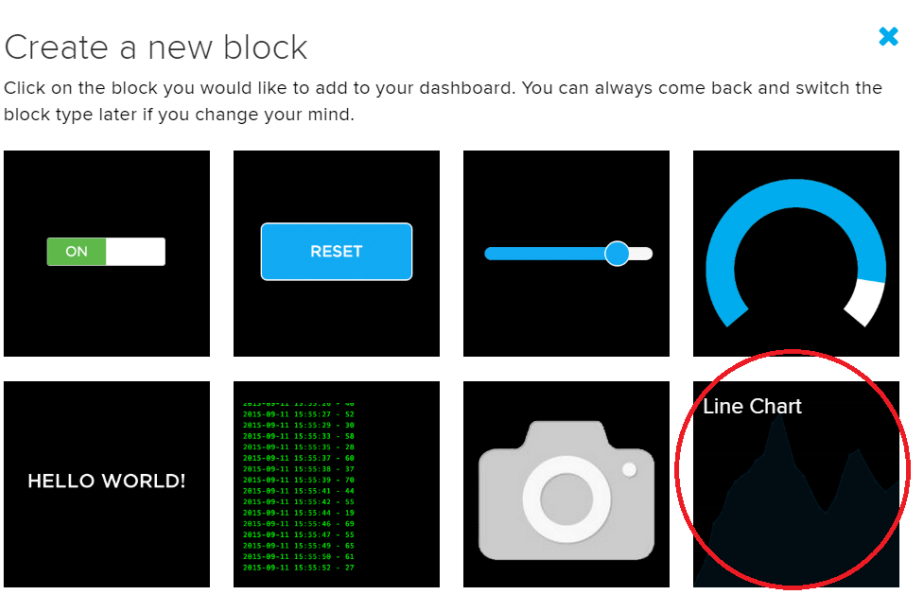
Když máme vytvořenou složku, kde budou nahrávána naše data, klikneme na záložku „Dashboards“, vybereme a klikneme na námi vytvořený projekt a ve pravém menu klikneme na tlačítko „+“.



Obrázek 25: Instalace cloudové služby: přidání objektu do projektu

[38]

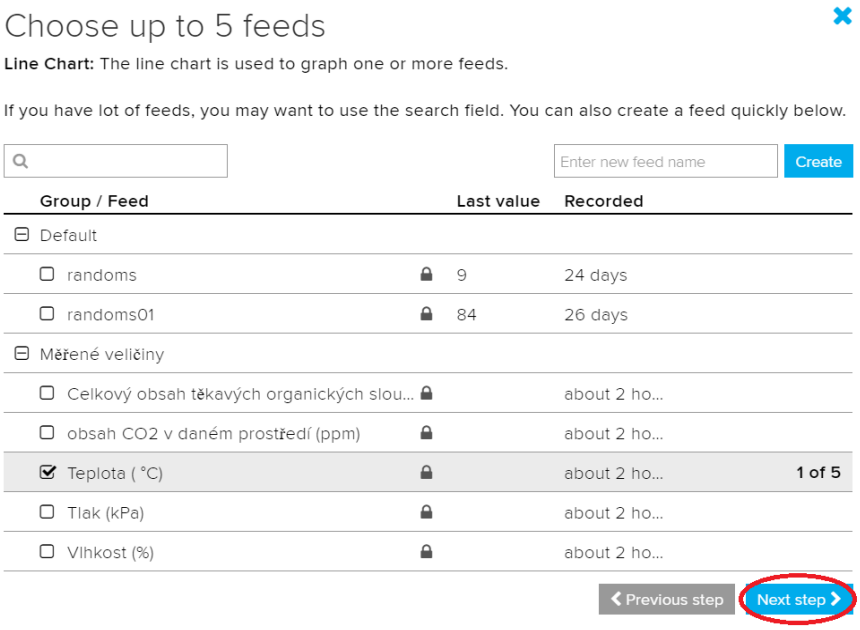
Otevře se nám okno s nabídkou bloků, které můžeme do projektu použít. Pro měření teploty se jeví jako nejlepší možnost graf, proto vybereme blok s názvem „Line Chart“.



Obrázek 26: Instalace cloudové služby: vybrání objektu

[38]

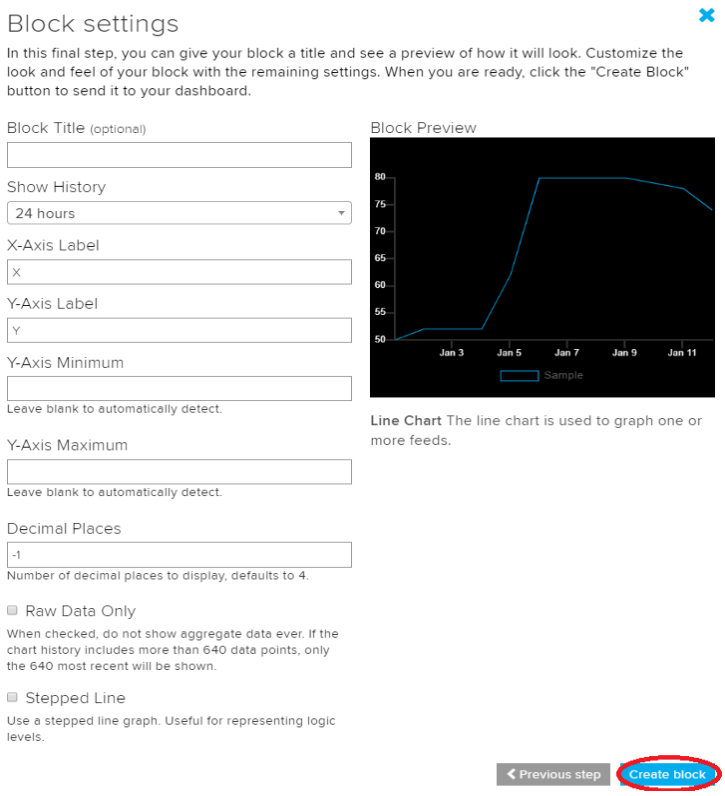
Zde vybereme, jaké složky s daty chceme na grafu zobrazit, do jednoho bloku, pokud používáme verzi zdarma můžeme nechat zobrazit až 5 různých složek dat v jednom grafu. Když máme vybrány složky, co chceme zobrazit dáme tlačítko - „Next Step“.



Obrázek 27: Instalace cloudové služby: spojení Feeds k objektu

[38]

Objeví se nám okno, kde můžeme graf upravit, dát mu název nebo nastavit jaké časové období se má zobrazovat. Když máme nastaveno, dáme tlačítko – „Create Block“.



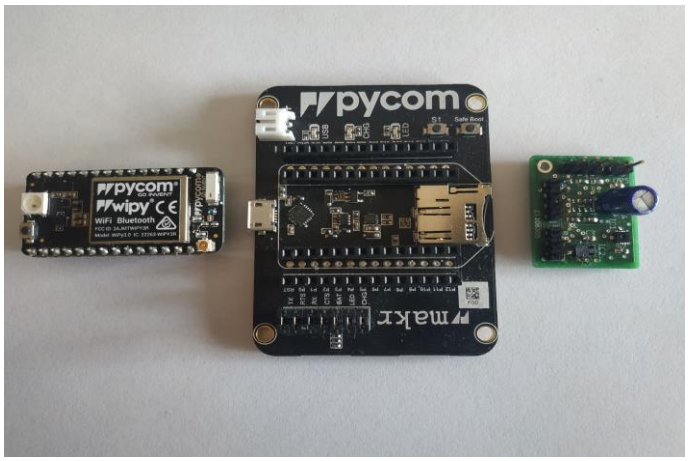
Obrázek 28: Instalace cloudové služby: úprava a vytvoření objektu

[38]

Po naměření a odeslání hodnot ze záznamníku na server Adafruit IO, se v grafu začnou objevovat hodnoty.

6 Návrh

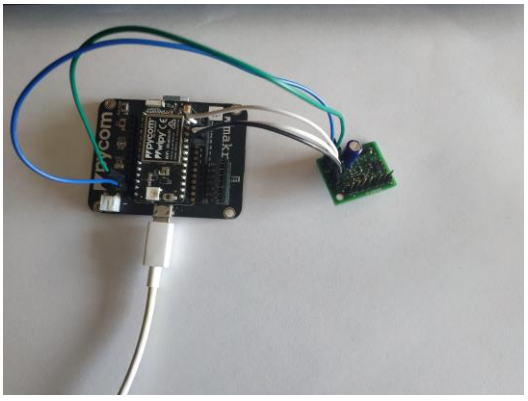
Záznamník stavu prostředí byl navržen s cílem dosáhnout optimálního a jednoduchého řešení. Celý systém se skládá ze tří komponent. Z platformy WiPy 3.0, rozšiřovací desky pro tuto platformu a z navržené desky na které se nacházejí všechny měřící senzory, a to ve stavu, kdy buď jsou na desce přímo umístěné, nebo jsou k desce připojené.



Obrázek 29: Komponenty, ze kterých se skládá návrh

6.1 Zapojení

Platforma WiPy je napájená z rozvodné sítě. Z rozšiřovací desky dále vystupují dva vodiče na napájení a tři vodiče použité pro data, kdy dva z nich jsou určeny pro I2C komunikaci a jeden pro 1-Wire komunikaci. Celý systém funguje na principu, kdy při dodání elektrické energie do platformy, se platforma připojí k bezdrátové internetové síti, dále v přesně stanoveném čase získá data ze senzorů, tyto data poté uloží na SD kartu a pošle na cloudovou službu Adafruit IO.



Obrázek 30: Zapojení návrhu

7 Program

Program pro platformu WiPy 3.0, byl napsán v jazyce Micropython, za pomoci programu Atom IDE. Pro implementaci na platformu bylo použito rozšíření pymakr, Program zajišťuje získání dat ze senzorů a jejich úpravu a následné ukládání dat na SD kartu a posílání dat na server Adafruit IO, pomocí protokolu MQTT. Určité části kódu byly přejaty a upraveny z jiných, již existujících programů.

7.1 Třída data senzorů

Tato třída v sobě obsahuje funkce, které získávají data ze senzorů. Každý senzor využívá svou knihovnu[14], která zajišťuje správnou úpravu a načtení dat. Dále obsahuje funkci, která přiřazuje naměřená data ke konkrétním feeds na serveru Adafruit IO.

7.1.1 Získání dat ze senzorů

Tyto jednotlivé funkce zajišťují získání dat ze senzorů za pomoci knihoven. Funkce _init_ je konstruktor, který pomáhá inicializovat vybrané proměnné, je zde i nastavení pinů pro komunikaci, kdy pro 1-Wire je nastaven pin 11 a pro komunikaci I2C jsou nastaveny piny 9 a 10. Pin 9 je nastaven pro SDA a pin 10 pro SCL.

```
class Senzory_data():

    def __init__(self):
        self.onetemp = DS18X20(OneWire(Pin('P11')))
        self.i2c = I2C(0, pins=('P9','P10')) #(P09=SDA, P10=SCL)
        self.i2c.init(I2C.MASTER, baudrate=20000)
        self.ccs811 = CCS811.CCS811(i2c=self.i2c, addr=90)
        self.sht31sensor = sht31d_2.SHT31(self.i2c)

    def onewire_temperature(self):
        self.onetemp.start_conversion()
        time.sleep(2)
        one_temp_real = self.onetemp.read_temp_async()
        time.sleep(3)
        return one_temp_real + 24.75
```

Obrázek 31: Blok kódu pro zjištění dat ze senzoru 1-Wire

[14] Pro 1-Wire byla využita knihovna: https://docs.pycom.io/tutorials/all/owd/
Pro CCS811 byla využita knihovna: https://github.com/Notthemarsian/CCS811/blob/master/CCS811.py
Pro SHT31d byla využita knihovna: https://github.com/kfricke/micropython-sht31/blob/master/sht31.py

```
def CCS811_CO2(self):
    while not self.ccs811.data_ready():
        time.sleep(1)
    return self.ccs811.eCO2

def CCS811_TVOC(self):
    while not self.ccs811.data_ready():
        time.sleep(1)
    return self.ccs811.tVOC

def sht31d_temp(self):
    sht31_t, _ = self.sht31sensor.get_temp_humi()
    return sht31_t

def sht31d_hum(self):
    _, sht31_h = self.sht31sensor.get_temp_humi()
    return sht31_h
```

Obrázek 32: Blok kódu pro zjištění dat ze senzorů

7.1.2 Přiřazení dat

Tato funkce přiřazuje naměřeným hodnotám jména, která jsou použita na serveru Adafruit IO a slouží k identifikaci, kam přesně se mají data na server uložit.

```
def measure_all(self):
    return {'celkovy-obsah-tekavych-organickych-sloucenin-v-danem-prostredi':self.CCS811_TVOC(),
            'obsah-co2-v-danem-prostredi-ppm':self.CCS811_CO2(),
            'teplota-degrees-c':self.sht31d_temp(), #:self.onewire_temperature()
            'teplota-onewire':self.onewire_temperature(),
            'vlhkost-percent':self.sht31d_hum()}
```

Obrázek 33: Blok kódu pro přiřazení dat

7.2 Třída odesílání a zápis dat

Tato třída v sobě obsahuje funkce potřebné na připojení k bezdrátové síti, připojení se na server Adafruit IO, odesílání dat na server a ukládání dat na SD kartu. Obsahuje také funkci na časovou synchronizaci čipu, pro zajištění aktuálního času.

7.2.1 Údaje pro MQTT

Tento blok kódu zajišťuje údaje potřebné k přihlášení na server Adafruit IO pomocí protokolu MQTT. MQTT_CLIENT je slovník a je deklarovaný ze stažené knihovny umqtt.py[15]. WIFI_LIST je také slovník, který v sobě zahrnuje údaje o připojení k bezdrátový sítím a vyhledává se v něm pomocí parametru ssid, kdy při skenování dostupných sítí se vezme první, která se najde a pomocí její ssid se začne hledat ve slovníku, zda o ní jsou údaje, pokud

[15] Kód použitý v knihovně umqtt.py je dostupný na stránce:
https://raw.githubusercontent.com/micropython/micropython-lib/master/umqtt.simple/umqtt/simple.py

ano pokusí se připojit, pokud ne vezme se další. Základ pro tuto část kódu jsem čerpal z webových stránek[16]. [40]

```
class Odesilani_zapis_dat():
    # Adafruit IO (AIO) configurace:
    MQTT_CLIENT=dict(server = 'io.adafruit.com',
                     port = 1883,
                     user = 'Praky',
                     key = 'aio_URaf15aY42CMgE9UTw7qDAWHrYeN',
                     client_id = ubinascii.hexlify(machine.unique_id()),
                     data_path = 'Praky/feeds/merene-veliciny.')
    WIFI_LIST={i.get('ssid'): i for i in [
    dict(ssid = 'PraWifi', password = '@$9atp7!}y^SEU9=Sslp', encryption = WLAN.WPA2),
    dict(ssid = 'FIFA-WIFI', password = '%54905Lz', encryption = WLAN.WPA2)
    ]}
```

Obrázek 34: Blok kódu s údaji pro server Adafruit IO

7.2.2 Připojení k wifi

Tento blok kódu zajišťuje skenování bezdrátových sítí v dosahu čipu a následné připojení. Funkce wifi_connect nejprve provede sken okolí. Najde všechny bezdrátové sítě v dosahu a pomocí for cyklu vybírá jednotlivé sítě. Hledá ve slovníku, zda má vybraná sít údaj. Pokud záznam má, pokusí se připojit. Pokud vyprší nastavený čas, nešlo se připojit, vezme další sít a postup se opakuje. Pokud vybraná sít záznam nemá, bude ignorována a vybere se další. Základ pro tuto část kódu jsem čerpal z webových stránek[17].

```
def __init__(self):
    self.sd_card_reconnect()
    self.wlan = WLAN(mode=WLAN.STA)
    self.rtc = RTC()

def wifi_connect(self):
    print('scan start')
    nets = self.wlan.scan()
    print('scan stop')
    for net in nets:
        desire_net = self.WIFI_LIST.get(net.ssid)
        if desire_net is not None:
            print(desire_net)
            start=time.ticks_ms()
            try:
                self.wlan.connect(desire_net.get('ssid'), auth=(net.sec, desire_net.get('password')), timeout=5000)
                while not self.wlan.isconnected() and time.ticks_ms() < start + 6000:
                    machine.idle()
            except socket.timeout:
                print('Cannot be connect')
            if self.wlan.isconnected():
                print('WLAN connection succeeded!')
                return True
    print('Cannot find wifi')
    return False
```

Obrázek 35: Blok kódu pro připojení k WIFI

[16] https://core-electronics.com.au/tutorials/internet-of-things-with-pycom-and-adafruit-io.html
[17] https://docs.pycom.io/tutorials/all/wlan/

7.2.3 Připojení se k Adafruit IO

Tento blok kódu zajišťuje synchronizaci vnitřních hodin procesoru se serverem ntp.org, pro získání aktuálního času. Dále načte údaje potřebné k připojení se k serveru Adafruit IO přes protokol MQTT a připojí se. Základ pro tuto část kódu jsem čerpal z webových stránek[18].

```
def time_synchronize(self):
    self.rtc.ntp_sync('1.cz.pool.ntp.org',update_period=3600)

def connect_to_aio_server_mqtt(self):
    self.client = MQTTClient(self.MQTT_CLIENT.get('client_id'),
                             self.MQTT_CLIENT.get('server'),
                             self.MQTT_CLIENT.get('port'),
                             self.MQTT_CLIENT.get('user'),
                             self.MQTT_CLIENT.get('key'))
    self.client.connect()
```

Obrázek 36: Blok kódu s připojením se na server pomocí MQTT

7.2.4 Zápis dat na SD kartu

Tento blok kódu zajišťuje zápis naměřených hodnot na SD kartu. Prvně si zjistí čas pomocí funkce rtc.now (), dále si připojí SD kartu a zapíše data, struktura zápisu dat je ve tvaru, kdy, pokud není vytvořená složka pro data, tak ji vytvoří, dále vytvoří podsložku s aktuálním rokem, zde vytvoří podsložku s aktuálním měsícem a zde vytvoří textový dokument, ve kterém zapíše naměřenou hodnotu a čas, kdy byla naměřena.

```
def write_data_to_sd_card(self, measure_data):

    year, month, day, hour, minute, second = self.rtc.now()[:6]
    os.mount(self.sd, '/sd')

    for variable, value in measure_data.items():
        if variable not in os.listdir('/sd'):
            os.mkdir('/sd/%s' %variable)
        if str(year) not in os.listdir('/sd/%s' %variable):
            os.mkdir('/sd/%s/%d' %(variable, year))
        if str(month) not in os.listdir('/sd/%s/%d' %(variable, year)):
            os.mkdir('/sd/%s/%d/%d' %(variable, year, month))

        with open('/sd/%s/%d/%d/%d.txt' %(variable, year, month, day), 'a+') as f:
            f.write('time = %d:%02d:%02d\tvalue = %d\n' %(hour, minute, second, value))

    os.umount('/sd')
```

Obrázek 37: Blok kódu pro uložení dat na SD kartu

[18] https://core-electronics.com.au/tutorials/internet-of-things-with-pycom-and-adafruit-io.html

7.2.5 Znovu připojení SD karty

Tento blok kódu se snaží znovu připojit SD kartu, pokud byla za běhu programu odpojena.

```
def sd_card_reconnect(self):
    try:
        self.sd = SD()
        print('SD card mounted')
    except Exception as e:
        self.sd = None
        print('SD card unmounted')
```

Obrázek 38: Blok kódu pro znovu připojení SD karty

7.2.6 Posílání dat na server

Tento blok kódu zajišťuje publikaci naměřených dat na server. Základ pro tuto část kódu jsem čerpal z webových stránek[19].

```
def send_data_to_server(self ,measure_data):
    for variable, value in measure_data.items():
        print('Publishing: %s to %s ... ' %(value, variable), end='')
        try:
            self.client.publish(topic=self.MQTT_CLIENT.get('data_path')+variable, msg=str(value))
            print('DONE')
        except Exception as e:
            print('FAILED')
            self.client.connect()
```

Obrázek 39: Blok kódu s posíláním dat na server

7.3 Třída pro hlavní blok programu

Tato třída v sobě obsahuje hlavní blok programu. V konstruktoru je inicializován časový interval, který udává, jak často se budou získávat data. Dále je tu funkce zajištující že se tento interval dodržuje a funkce main, která bude zavolána ihned po spuštění programu.

7.3.1 Funkce pro zajištění intervalu

Tento blok kódu porovnává aktuální procesorový čas programu s procesorovým časem programu v době posledního měření a ověřuje, zda mezi nimi proběhl dostatečný časový interval, z důvodu nezatěžování serveru příliš rychlým odesíláním.

```
class Callcentrum():
    def __init__(self):
        pycom.heartbeat(False)
        self.senzors = Senzory_data()
        self.sendin_data = Odesilani_zapis_dat()
        self.last_send = 0
        self.interval = 900000
```

Obrázek 40: Blok kódu s nastavením délky intervalu posílání

[19] https://core-electronics.com.au/tutorials/internet-of-things-with-pycom-and-adafruit-io.html

```
def interval_checker(self):
    if time.ticks_ms() - self.last_send >= self.interval:
        self.last_send = time.ticks_ms()
        return True
    return False # Too soon since last one sent.
```

Obrázek 41: Blok kódu se zajištěním intervalu posílání

7.3.2 Hlavní blok programu

Tento blok kódu představuje hlavní blok programu. Zavolá metodu wifi_connect, která vrátí hodnotu, zda se podařilo připojit na internet. Pokud ano, zavolají se metody time_synchronize a connect_to_aio_server_mqtt. Poté se zavolá metoda interval_checker, pokud uběhl zadaný interval, naměří se data, která se odešlou na server za podmínky, že je aktivní spojení. Poté se data zapíší na SD kartu. Pokud nastane chyba, přeruší se spojení se serverem a vypne se připojení k internetu.

```
def main(self):
    if self.sendin_data.wifi_connect():
        self.sendin_data.time_synchronize()
        self.sendin_data.connect_to_aio_server_mqtt()

    try:
        while True:
            if self.interval_checker():
                measure_data = self.senzors.measure_all()
                if self.sendin_data.wlan.isconnected():
                    self.sendin_data.send_data_to_server(measure_data)
                try:
                    self.sendin_data.write_data_to_sd_card(measure_data)
                    print('SD card is conected')
                except Exception as e:
                    print(e)
                    print('SD card is disconected')
                    self.sendin_data.sd_card_reconnect()

    finally:
        if self.sendin_data.wlan.isconnected():
            self.sendin_data.client.disconnect()
            self.sendin_data.wlan.disconnect()
        self.sendin_data.client = None
        self.sendin_data.wlan = None
        print('Disconnected from Adafruit IO.')
        pycom.rgbled(False)
```

Obrázek 42: Blok kódu s hlavním kódem programu

7.4 Kontrola volání

Tento blok kódu zjišťuje, zda tento program byl zavolán přímo, nebo zda byl importovaný.

```
if __name__ == '__main__':  # kontorla spusteni tohodle file
    callcentrum = Callcentrum()
    callcentrum.main()
```

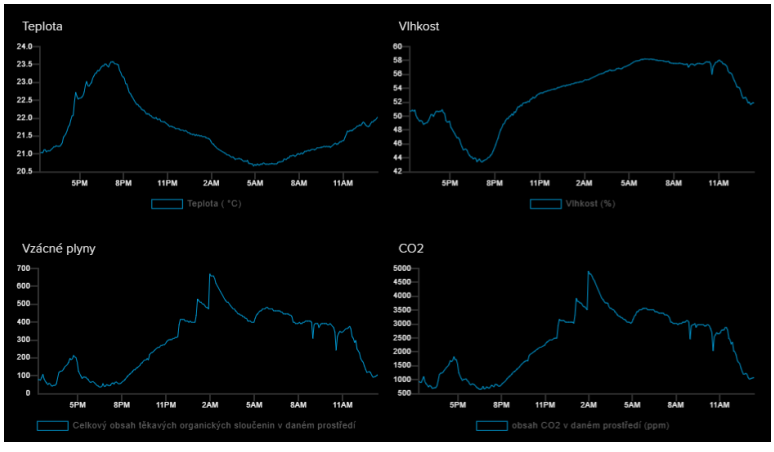
Obrázek 43: Blok kódu zajištující kontrolu hlavního bloku programu

8 Naměřené hodnoty

V tabulce jsou zaznamenány zaokrouhlené hodnoty měření, za jeden den. Z naměřených hodnot CO2, lze pozorovat, kdy se v měřeném prostředí větralo a hodnoty byly výrazně nižší, než když se nevětralo. V grafech jsou zaznamenány hodnoty v měřícím intervalu 5 minut.

čas	teplota [°C]	teplota 1-Wire [°C]	Vlhkost [%]	CO2 [ppm]	TVOC [ppb]
0:00	20	19	53	1323	140
1:00	19	19	54	1095	104
2:00	19	19	55	1055	99
3:00	19	19	55	1829	217
4:00	19	19	56	1829	214
5:00	19	19	56	2432	309
6:00	19	19	56	2296	317
7:00	20	20	56	1899	288
8:00	20	20	55	1553	172
9:00	20	20	54	1634	187
10:00	20	20	47	414	75
11:00	20	20	48	932	78
12:00	21	20	51	1058	98
13:00	21	21	49	795	58
14:00	21	21	50	1339	139
15:00	22	22	50	1245	126
16:00	23	22	48	840	67
17:00	23	23	45	714	47
18:00	23	22	43	862	68
19:00	22	21	45	1939	146
20:00	22	21	50	2318	292
21:00	21	21	51	3125	439
22:00	21	21	53	3023	482
23:00	21	21	54	3330	670

Tabulka 19: Tabulka naměřených hodnot za jeden den



Obrázek 44: Průběhy měřených hodnot na Serveru Adafruit IO

ZÁVĚR

Cílem bakalářské práce bylo navrhnout a realizovat záznamník stavu prostředí, který by sledoval kvalitu ovzduší v daném prostředí. Základ byl postaven na platformě WiPy. Senzory byly vybrány na základě oblíbenosti, kdy velmi používané senzory, mají lépe zdokumentovány možnosti použití a jejich správná činnost v systémech je prověřena.

Návrh se podařilo splnit, záznamník měří teplotu ze dvou senzorů pro lepší názornost a porovnání, dále měří vlhkost, hodnoty CO2 a hodnoty těkavých organických sloučenin. Systém je připraven na další rozšíření ať už jde o přidání nových senzorů, nebo pro implementaci do složitějších systémů.

Dalším cílem bylo naměřená data uložit na záznamové médium a zasílat je na internet. Data jsou tedy po naměření uložena na SD kartu a jsou také odeslána na serverovou službu Adafruit IO, kde jsou zobrazena v přehledných grafech.

Při návrhu byly zjištěny nedostatky a problémy, které nejsou v aktuální verzi opraveny a budou implementovány v budoucnu. To zahrnuje například znovu připojení SD karty při nechtěném odpojení za běhu programu, restart a obnova programu při zjištění chyby, nebo přesnější kalibrace a nastavení senzorů.

POUŽITÁ LITERATURA

Bibliografie

[1] ROUSE, Margaret. Internet of things (IoT). *IoTAgenda* [online]. Newton: TechTarget, 2020 [cit. 2020-03-25]. Dostupné z: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[2] Connected Devices Accelerate The Need For IPv6 In The Internet Of Things. In: *Teamarin* [online]. Centreville (Maryland): American Registry for Internet Numbers, c1997-2020 [cit. 2020-04-06]. Dostupné z: https://teamarin.net/2013/12/27/connected-devices-accelerate-the-need-for-ipv6-in-the-internet-of-things/

[3] TIŠNOVSKÝ, Pavel. Komunikace po sériové sběrnici I2C. *ROOT.cz* [online]. Praha: Internet Info, c1998-2020 [cit. 2020-03-25]. Dostupné z: https://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/

[4] Sběrnice I2C. In: *Vyvoj.hw.cz* [online]. Praha: HW server, c1997-2014 [cit. 2020-04-06]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/strucny-popis-sbernice-i2c-a-jeji-prakticke-vyuziti-k-pripojeni-externi-eeprom-24lc256

[5] *I2C Info – I2C Bus, Interface and Protocol* [online]. Carlsbad (California): Ezoic, 2020 [cit. 2020-03-27]. Dostupné z: https://i2c.info/

[6] *I2C BUS* [online]. Hamburg: Telos, 2019 [cit. 2020-03-27]. Dostupné z: i2c-bus.org

[7] Bus Arbitration. In: *EmSA* [online]. Barsinghausen (Německo): Embedded Systems Academy, c1999-2017 [cit. 2020-04-06]. Dostupné z: http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/general-introduction/bus-arbitration.html

[8] Adafruit HUZZAH32. *Adafruit* [online]. New York: Adafruit Industries, 2015 [cit. 2020-03-28]. Dostupné z: http://www.adafruit.com/product/3619

[9] Linklt Smart 7688. *Seeed Studio* [online]. Shenzen (China): Seeed Technology, c2008-2020 [cit. 2020-03-28]. Dostupné z: https://www.seeedstudio.com/Linklt-Smart-7688.html

[10] PHOTON WIFI DEVELOPMENT BOARD. *Particle Store* [online]. San Francisco (California): Particle, 2020 [cit. 2020-03-28]. Dostupné z: https://store.particle.io/collections/wifi/products/photon?variant=10284195907

[11] LinkIt 7697. *Seeed Studio* [online]. Shenzen (China): Seeed Technology, c2008-2020 [cit. 2020-03-28]. Dostupné z: https://www.seeedstudio.com/LinkIt-7697-p-2818.html

[12] MEGA+WiFi R3 ATmega2560+ESP8266. *RobotDyn* [online]. Zhuhai (China): RobotDyn, 2019 [cit. 2020-03-28]. Dostupné z: https://robotdyn.com/mega-wifi-r3-

atmega2560-esp8266-flash-32mb-usb-ttl-ch340g-micro-usb.html

[13] Omega2+. *Onion* [online]. Boston (Massachusetts): Onion Corporation, 2020 [cit. 2020-03-28]. Dostupné z: https://onion.io/store/omega2p/

[14] Raspberry Pi 3 Model B 64-bit 1GB RAM. *RPishop.cz* [online]. České Budějovice: rpishop.cz, 2020 [cit. 2020-03-28]. Dostupné z: https://rpishop.cz/raspberry-pi-3b/283-raspberry-pi-3-model-b-64-bit-5060214370028.html?SubmitCurrency=1&id_currency=2

[15] WiPy 3.0. *Pycom* [online]. Cranleigh (United Kingdom): Pycom, 2018 [cit. 2020-03-28]. Dostupné z: https://pycom.io/product/wipy-3-0/4/

[16] ZIMMERMAN JONES, Andrew. Temperature Definition in Science. *ThoughtCo.* [online]. New York: Dotdash, 2019 [cit. 2020-04-06]. Dostupné z: https://www.thoughtco.com/temperature-definition-in-science-2699014

[17] MAX31850TATB+. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-15]. Dostupné z: https://cz.farnell.com/maxim-integrated-products/max31850tatb/temperature-sensor-4deg-c-tdfn/dp/2515620

[18] ADT7420UCPZ-R2. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-15]. Dostupné z: https://cz.farnell.com/analog-devices/adt7420ucpz-r2/temp-sensor-16bit-0-25c-16lfcsp/dp/2306581

[19] SMT172-220. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-15]. Dostupné z: https://cz.farnell.com/smartec/smt172-220/temperature-sensor-1deg-c-to-220/dp/2543396

[20] DS18S20+. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-15]. Dostupné z: https://cz.farnell.com/maxim-integrated-products/ds18s20/digital-thermometer-18s20-3to/dp/2519401?st=DS18S20

[21] Density and Pressure. *Khan Academy* [online]. San Jose (California): Khan Academy, 2020 [cit. 2020-04-06]. Dostupné z: https://www.khanacademy.org/science/physics/fluids/density-and-pressure/a/pressure-article

[22] DPS422XTSA1. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/infineon/dps422xtsa1/pressure-temp-sensor-barometric/dp/2986299RL

[23] 2SMPB-02E. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/omron-electronic-components/2smpb-02e/pressure-sensor-lga-9/dp/2830096

[24] 2SMPB-01-01. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/omron-electronic-components/2smpb-01-01/pressure-sensor-30-110kpa-qfn/dp/2449220

[25] MPL115A2. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/nxp/mpl115a2/mini-i2c-digital-barometer-50/dp/2102577

[26] Humidity. *National Geographic Society* [online]. Washington (Maryland): National Geographic Society, c1996-2020 [cit. 2020-04-06]. Dostupné z: https://www.nationalgeographic.org/encyclopedia/humidity/

[27] HPP845E034R5. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/sensor-solutions-te-connectivity/hpp845e034r5/humidity-temp-sensor-digital-dfn/dp/2771896

[28] HDC1080DMBT. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/texas-instruments/hdc1080dmbt/humidity-temp-sensor-digital-wson/dp/3009094RL

[29] SHT25. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/sensirion/sht25/humidity-temp-sensor-digital-dfn/dp/2126337

[30] SHT31-DIS-B. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/sensirion/sht31-dis-b/humidity-temp-sensor-digital-dfn/dp/2474218

[31] What is CO2?. *Airthings* [online]. Oslo: Airthings, 2020 [cit. 2020-04-06]. Dostupné z: https://www.airthings.com/en/what-is-carbon-dioxide

[32] SCD30. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/sensirion/scd30-co2-sensor-module/gas-sensor-co2-0-075a-5-5v-40000ppm/dp/2887812

[33] T6613-5KC. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/amphenol-advanced-sensors/t6613-5kc/sensor-co2-sensor-0-5000ppm/dp/2472286

[34] T6713. *Farnell* [online]. Leeds (England): Avnet, 2019 [cit. 2020-04-14]. Dostupné z: https://cz.farnell.com/amphenol-advanced-sensors/t6713/sensor-co2-sensor-0-2000ppm/dp/2472298?st=T6713#anchorTechnicalDOCS

[35] Volatile Organic Compounds' Impact on Indoor Air Quality. *United States Environmental Protection Agency* [online]. Washington (Maryland): Environmental Protection Agency, 2017 [cit. 2020-04-06]. Dostupné z: https://www.epa.gov/indoor-air-quality-iaq/volatile-organic-compounds-impact-indoor-air-quality

[36] SGP30. *SparkFun Electronic* [online]. Niwot (Colorado): SparkFun Electronics, 2003 [cit. 2020-04-14]. Dostupné z: https://www.sparkfun.com/products/14813

[37] CCS811. *SparkFun Electronic* [online]. Niwot (Colorado): SparkFun Electronics, 2003 [cit. 2020-04-14]. Dostupné z: https://www.sparkfun.com/products/14193

[38] *Adafruit IO* [online]. New York: Adafruit Industries, 2015 [cit. 2020-04-06]. Dostupné z:

https://io.adafruit.com/

[39] MALÝ, Martin. Protokol MQTT: komunikační standard pro IoT. *ROOT.cz* [online]. Praha: Internet Info, c1998-2020 [cit. 2020-04-08]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/

[40] Internet of Things with Pycom and Adafruit IO - From Zero to Hero. *Core Electronics* [online]. Newcastle (Australia): CORE ELECTRONICS, 2013 [cit. 2020-04-10]. Dostupné z: https://core-electronics.com.au/tutorials/internet-of-things-with-pycom-and-adafruit-io.html

PŘÍLOHY

Příloha A – Schéma zapojení

PŘÍLOHA A – Schéma zapojení