UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Monitoring v odlehlých lokalitách

Martin Skřebský

Bakalářská práce
2021

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Skřebský**
Osobní číslo:	**I18056**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Komunikační a mikroprocesorová technika**
Téma práce:	**Monitoring v odlehlých lokalitách**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování

Rozvoj technologií nastartovat období IoT i v nečekaných oblastech, kdy i toustovač může být připojen do sítě. Stále však zůstávají oblasti, kde technologie pronikají pomaleji, jelikož zde narážejí na technologické limity, jako jsou možnosti napájení a datového připojení. Cílem práce bude návrh a ověření možnosti systému pro monitoring, nebo řízení v lokalitách mimo elektrické napájení, případně dosah běžných sítí jako například v lesnictví, myslivectví a zemědělství. Praktická práce bude obsahovat návrh konkrétního zařízení pro vzdálený monitoring zvolené veličiny a ověření funkčnosti/spolehlivosti vybraného řešení. Práce bude obsahovat rozbor možnosti řešení přenosu dat ze vzdálené lokality, přehled by mě obsahovat jak metody přímého spojení, možnosti retranslace za využití více stanic až po využití existujících datových sítí jako gsm, nb-iot, sigfox, LoRa a podobně.

Rozsah pracovní zprávy: **30-60**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

[1] VÁŇA, V. Mikrokontroléry ATMEL AVR: popis procesoru a instrukční soubor. Praha: BEN technická literatura, 2003. 336 s. ISBN 978-80-7300-083-0.
[2] VÁŇA, V. Mikrokontroléry ATMEL AVR: programování v jazyce C. Praha: BEN technická literatura, 2003. 216 s. ISBN 978-80-7300-102-0.
[3] VLACH, J. Řízení a vizualizace technologických procesů. Praha: BEN technická literatura, 2002. 160 s. ISBN 978-80-86056-66-X.
[4] BRTNÍK, B. Základní elektronické obvody. Praha:BEN technická literatura, 2011. 156s. ISBN 978-80-7300-408-8
[5] RIPKA, P.; TIPEK, A. Master Book of Sensors. Praha : BEN, 2003. ISBN 0-12-752184

Vedoucí bakalářské práce: **Ing. Pavel Rozsíval**
Katedra elektrotechniky

Datum zadání bakalářské práce: **15. listopadu 2020**
Termín odevzdání bakalářské práce: **14. května 2021**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Pidanič, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 29. ledna 2021

Prohlášení autora

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., autorský zákon, 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. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 11. 08. 21 Martin Skřebský

Poděkování

Rád bych poděkoval své rodině, která mě bezmezně a trpělivě podporovala v po celou dobu studia a sepsání této práce. Dále bych chtěl poděkovat také svému vedoucímu práce Ing. Pavlovi Rozsívalovi za vzorné vedení a odborné konzultace k práci.

Anotace

Tato práce vznikla jako popis problematiky zařízení instalovaných v odlehlých oblastech a předvedení možností řešení jednoduchých příkladů. Práce se soustředí na již zavedené standardy sítí a k nim vzniklé moduly. V praktické části se zaměřuji na platformy s velkou podporou a přehledností dokumentace. Praktická část obsahuje zapojení a naprogramování tří různých zařízení umístěných v krmelci, včelíně a na posedu.

Klíčová slova

internet věcí, sigfox, lora, nb-iot, monitoring

Title

Remote location monitoring

Annotation

This work has been created as a description of the problems of equipment installed in remote areas and demonstration of possible solutions with simple examples. The work focuses on already established network standards and modules created for them. In the practical part I focus on platforms with good support and documentation. The practical part contains the connection and programming of three different devices located in the feeder, apiary and hunting stand.

Keywords

Internet of things, sigfox, lora, nb-iot, monitoring

Obsah

Seznam zkratek

GSM	Groupe Spécial Mobile
IoT	Internet věcí (Internet of Things)
LAN	lokální síť (Local Area Network)
PWM	pulzně šířková modulace (Pulse Width Moduation)
ADC	Analogově digitální převodník
DAC	Digitálně analogový převodník

Seznam obrázků

Úvod

I v dnešním světě, kde jsou technologie dostupné téměř všude, zde můžeme najít mnoho různých míst, které přístup pokročilým technologiím značně stěžují.

Taková místa nedisponují základními zdroji pro dokumentování různých jevů (teplota, tlak, pohyb, poloha) jako jsou například stálý přístup k elektrické síti či přímý přístup k internetu. Tyto zařízení tedy musejí používat jiné zdroje k dosažení cílů.

Nejvíce problémové faktory při řešení jsou napájení a v případě sledování veličin i připojení k internetu či jinému uživatelsky dostupnému systému. Největší část alternativních způsobů napájení zařízení jsou baterie či akumulátory. I když se při těchto řešeních snažíme dosáhnout co nejmenší spotřeby je nutné tyto zdroje energie vyměňovat či dobíjet jinými metodami, jako například solární energie.

Pro připojování podobných zařízení vzniklo celé odvětví propojování zařízení a to internet věcí. K této síti se lze připojit díky mnoha různým možnostem jako je wifi propojení, GSM, SigFox, LoRa apod. Každé z těchto připojení má své výhody i omezení, proto je při návrhu potřeba vzít v úvahu různé požadované nároky na polohu či množství přenášených dat.

1 Volba technologie připojení

Pro monitoring vzdálených lokalit je potřeba vytvořit nějaké datové připojení. Teoreticky nejpracnější variantou je tvorba vlastního rádiového pojítka (i když se v první fázi může jevit jako nejsnadnější), pokud bychom chtěli implementovat spojení s nějakou garancí spolehlivosti a bezpečnosti. Museli bychom nadefinovat nějaký protokol, kontrolu chyb, ověření případně další. Z tohoto pohledu vychází na první pohled (pohled programátora) složitější varianta, využít stávající technologií a standardů. Je sice nutná znalost API, ale řeší veškerou problematiku za nás. Nejvhodnější technologie pro moje řešení se jeví technologie používané v INTERNETU VECÍ (IOT).

Pod tímto souslovím si lze představit velkou spoustu možností využití. Název ovšem může být pro neznalé mírně zavádějící. Internetem věcí nazýváme síť vzájemně propojených zařízení, která ovšem nemusejí být nutně připojena k Internetu. Jde tedy o skupinu samostatných jednotek, které díky vzájemné komunikaci mohou plně spolupracovat na společném cíli.

Tento koncept vzniknul především za účelem zjednodušit lidskou práci na minimum. Toto se ovšem nedotýká pouze výrobních procesů. Plně automatizované továrny jsou zřejmě největším lákadlem a proto se IoT často spojuje s pojmem Průmysl 4.0. Chytré domácnosti jsou dalším velkým zástupcem v této oblasti. Majitelé domů si čím dál více žádají obsluhu svých domácností počínaje centrálním řízením regulovatelných světel až po přístup k ovládání libovolného zařízení z internetu pomocí svého chytrého telefonu.

Nemalou oblast ovšem tvoří daleko drobnější projekty. Ty mohou řešit pouze jednoduché úkoly jako změření jedné veličiny a odeslání dat. Zejména zde můžeme mluvit o dvou rozdílných variantách zpracování dat. První z možností je uzavřený systém kdy propojíme dvě nebo více zařízení a komunikace tedy probíhá pouze mezi nimi. Výhodou je bezpečnost a jednoduchost těchto spojení. Nevýhodu je ovšem relativně malý dosah pro komunikaci. Druhou variantou je připojení stanice do internetu. To lze přes síťový modul nebo pomocí některého z operátorů provozujících bezdrátové sítě v dané oblasti. Tento způsob nám umožní data poslat na server a mít k nim přístup z celého světa nebo je poskytnout veřejně.

Tento rozmach však bohužel také znamená více potenciálních cílů pro black-hat hackery. Útoky mohou vést ke zneužití či narušení diskrétní zóny uživatelů. Informace mohou být využity k vydírání, krádeži, neoprávněnému užívání těchto zařízení nebo jiné kriminální činnosti. Je tedy potřeba vždy důkladně zvážit jaké systémy a technologie vyváží pohodlí z ovládání či monitoringu a jaké by při možném zneužití narušovali náš osobní prostor.

1.1 Wifi

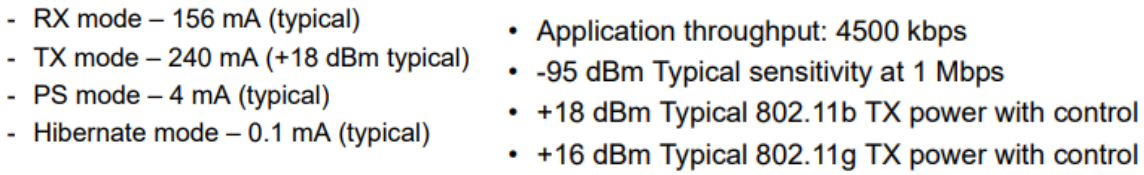
Tato technologie se využívá především k přenosu velkého objemu dat či připojení jednoduchého snímače do již zavedené domácí wifi sítě. V dnešním světě se využívá zavedeného standardu 2,4 a 5 GHz. Díky takto vysoké frekvenci je přenos velice rychlý, ale také se tím snižuje prostupnost různými překážkami například zdmi nebo dřevinami. Pokrytí sítě ovlivňují i další aspekty jako jsou třeba zarušení prostředí jinými vysílači, vyzařovaný výkon, typ antény, apod.

Pro maximální využití tohoto typu bezdrátové komunikace lze využít směrových antén. Vytvoří se tak spojení mezi dvěma pevnými body, na jejichž koncích lze vybudovat samostatné LAN sítě, které si mezi sebou mohou předávat data. Nejoptimálnějšího výsledku dosáhneme, když je mezi vysílačem a přijímačem přímá viditelnost. Ne vždy je ovšem taková možnost k dispozici a proto je důležité změřit jejich propustnost, aby spojení bylo stabilní.

Při téměř neustálém přenosu je nutné myslet i na to, jaký příkon je třeba dodávat a zda tyto požadavky můžeme naplnit. Tyto energetické nároky se z dlouhodobějšího hlediska často neslučují s možnostmi normálních baterií či akumulátorů. Ve větším množství aplikací se tedy musí volit možnost trvalého napájení a články ponechat pouze pro zálohu při krátkodobém výpadku.

Technologie je tedy uplatňována spíše na rozšíření domácí sítě do objektů, které disponují vlastním napájením a mají být datově spojeny s hlavní sítí. Příkladem využití wifi je zabezpečení garáže mimo pozemek LAN kamerovým systémem. Díky této topologii lze mít pouze jedno nahrávací zařízení pro obě budovy. Dojde tedy k úspoře peněz i zvýšení uživatelské přívětivosti.

Pro příklad energetické náročnosti se můžeme podívat na modem MRF24WG0MA, který patří mezi energeticky málo náročné wifi modemy

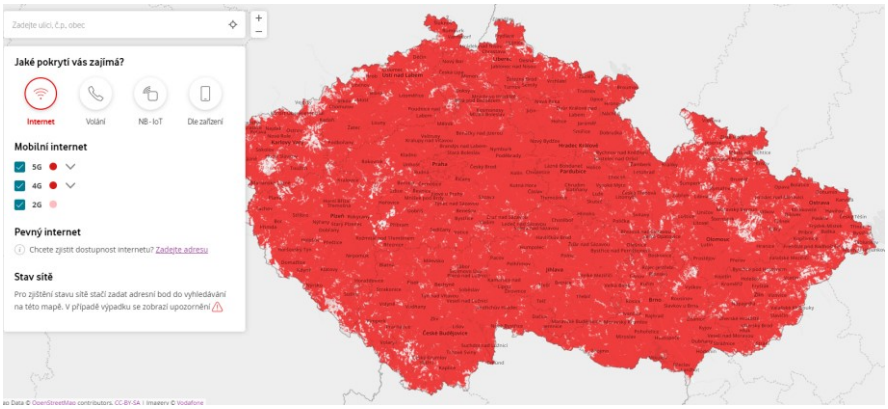


Obrázek 1: Parametry modemu MRF24WG0MA (výstřižek [1])

Na parametrech vidíme relativně nízkou spotřebu ve stovkách mA, s propustností max 4.5 Mb. Musíme si také uvědomit nutnost trvalého spojení, nebo poměrně dlouhou dobu pro navázání spojení, v případě že bychom modem vypínali.

1.2 GSM

GSM je buňková síť, což znamená, že zařízení komunikují se sítí prostřednictvím nejbližší buňky. GSM síť funguje na několika radiových frekvencích. V Česku jsou používány pro 2G sítě frekvence 900 a 1800 MHz, 3G sítě na 2100 MHz a LTE sítě na 800, 900, 1800 a 2100 MHz.



Obrázek 2: Mapa pokrytí Vodafone [2]

Využití GSM sítě je možné pomocí tzv. modemů a je možné pomocí SMS zpráv anebo pomocí datového přenosu některou z technologií který daný modem a poskytovatel připojení umožňuje. K dispozici je pak připojení přímo do sítě internet se všemi službami.

Výhodou oproti jiným možnostem je historicky husté pokrytí operátory nad naším územím a tudíž velká pravděpodobnost možnosti této sítě využít.

Největší nevýhodou tohoto řešení je vysoká energetická náročnost a relativní finanční náročnost spojená s platbou za přenesená data. Pro ukázku vezmeme modul Quectel EC25, jeho klidová spotřeba je kolem 30mA, pokud je uspaný tak 4mA.

Electrical Features	
Supply Voltage Range	3.0–3.6 V, 3.3 V Typ.
Power Consumption	3.9 mA @ Sleep, Typ. 30 mA @ Idle

	burst transmission	level on EGSM900.			
I_{VBAT}	Peak supply current (during transmission slot)	Maximum power control level on EGSM900.	1.8	2.0	A

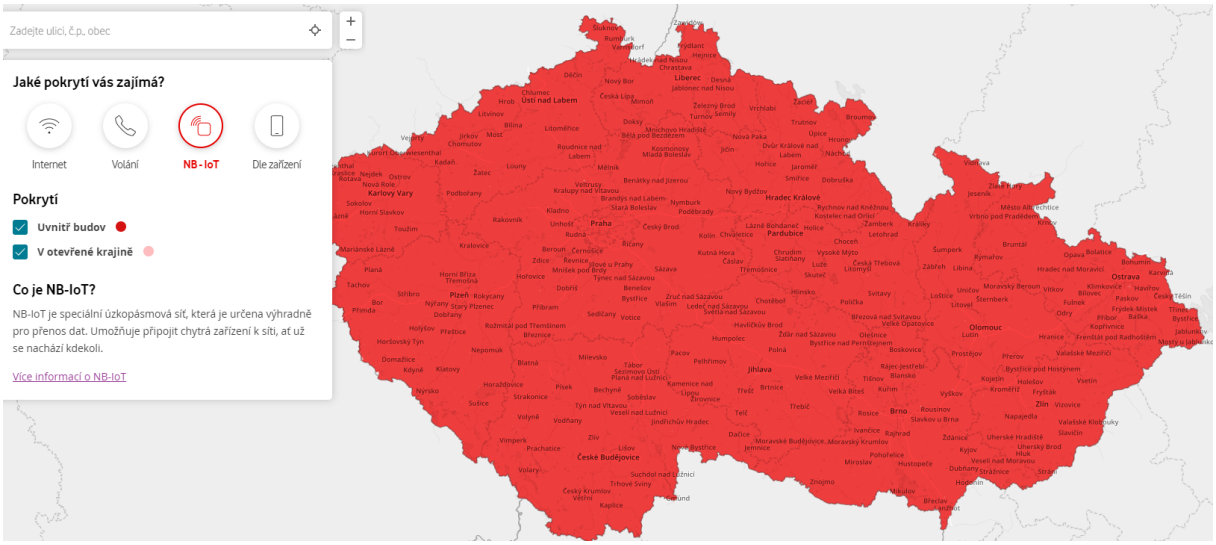
Obrázek 3: Spotřeba QUECTEL C25 [3]

V případě aktivního přenosu je spotřeba v jednotkách ampér při povoleném poklesu napájení 400mV. Z toho je vidět že připojení pomocí GSM sice nabízí velký dosah díky velkému pokrytí, ale na úkor energetické náročnosti a potřebě tvrdého zdroje napájení.

1.3 NB-IoT

NB-IoT, neboli narroband internet of things, je nízko energetická síť s širokým pokrytím. Je přímo určena pro použití se zařízeními s nízkou spotřebou a potřebou dobrého pokrytí. Technologie staví na standartu LTE, ale využívá pouze 200kHz šířku pásma.

Pokrytí touto technologií nabízí zatím pouze Vodafone a mapa pokrytí uvádí pokrytí lepší než je tomu u klasické mobilní sítě (uvádí 100%)



Obrázek 4: Pokrytí NB-IoTvodafone (mapa pokrytí) [2]

Drobnou nevýhodou technologie je její omezená podpora pro prototypování jak ze strany výrobců kitů, tak ze strany operátora, který by měl připojení umožnit.

1.4 LoRa

LoRa je síť pro internet věcí (IoT), která umožňuje svým zařízením obousměrnou komunikaci s omezeným počtem zpětných zpráv ve frekvenčním pásmu 868 MHz. Tato technologie umožňuje zařízením komunikovat, bezpečně a na velké vzdálenosti při zcela minimální spotřebě energie. LoRa dovoluje nasazení jednotlivých vysílacích stanic i v místních lokalitách, čímž dokáže posílit svůj signál a zajistit pokrytí tam, kde je potřeba.

Specifikace LoRaWAN je pak veřejně dostupná a definuje protokol přístupové vrstvy pro řízení komunikace mezi bránami LPWAN a koncovým zařízením.

Nevýhodou je, že sítě lora se zatím většinou používají v chytrých městech a podobných projektech a není celonárodní poskytovatel. Samozřejmou možností je vytvoření vlastní sítě, což vzhledem k velmi velkému dosahu je možnost pro některé lokality a potřeby optimální. Rádiový dosah se pak uvádí kolem 10 km s propustností ve stovkách bitů a desítkách kilobitů. Pro příklad parametry LoRa modemu RN2903

Specification	Description
Frequency Band	902.000 MHz to 928.000 MHz
Modulation Method	FSK, GFSK, and LoRa® Technology modulation
Maximum Over-the-Air Data Rate	300 kbps with FSK modulation; 12500 bps with LoRa Technology modulation
RF Connection	Board edge connection
Interface	UART
Operation Range	Up to 15 km coverage at suburban; up to 5 km coverage at urban area
Sensitivity at 1% PER	-146 dBm(1)
RF TX Power	Adjustable up to max. +18.5 dBm on 915 MHz band(2)
Generated Conductive Harmonics Level	Below -70 dBm
Temperature (operating)	-40°C to +85°C

Obrázek 5: RN2903 elektrické parametry - výstřižek [4]

Ze specifikací je vidět že nabízí spojení na 15km ve volné krajině a 12kb při použití LoRa.

Spotřeba elektrické energie je relativně nízká v porovnání s WiFi a GSM a vidíme, že záleží na vysílacím výkonu.

TX Power Setting	Output Power (dBm)	Typical Supply Current at 3.3V (mA)
2	3.0	42.6
3	4.0	44.8
4	5.0	47.3
5	6.0	49.6
6	7.0	52.0
7	8.0	55.0
8	9.0	57.7
9	10.0	61.0
10	11.0	64.8
11	12.0	73.1
12	13.0	78.0
14	14.7	83.0
15	15.5	88.0
16	16.3	95.8
17	17.0	103.6
20	18.5	124.4

Obrázek 6: RN2903 výkon/spotřeba (výstřižek [4])

1.5 SigFox

Sigfox využívá patentovanou technologii, která umožňuje komunikaci pomocí rozhlasového pásma ISM (industrial, scientific a medical), které se v Evropě šíří na frekvenci 868 MHz a 902 MHz v USA. Sigfox používá signál s dlouhým dosahem, který volně prochází skrz pevné objekty, tzv. "ultra narrowband", a na jehož vyzařování je třeba jen málo energie. Tato síť patří mezi "Low-Power Wide-Area Network" sítě (LPWAN). Síť je založena na hvězdicové topologii, kdy mezi každými dvěma stanicemi existuje vždy jen jedna cesta, a vyžaduje mobilního operátora, který zajišťuje vzniklý provoz. Signál může snadno pokrýt velkou plochu a dosáhne k podzemním objektům. [5]

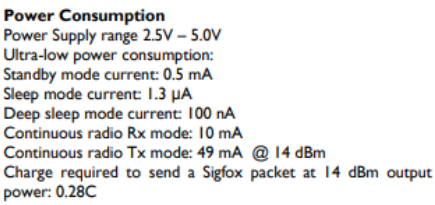
Je celosvětově rozšířená síť což otevírá velké množství využití. Tuto síť vytvořila stejnojmenná firma Sigfox. Poskytovatel tohoto připojení je v České republice společnost SimpleCell.

Pokrytí signálem je v radě případů lepší než u mobilních sítí (ČR). Pokud pokrytí chybí, je možné si pronajmout, nebo pořídit a zaregistrovat novou „Micro base station“, která danou lokalitu pokryje.

Obrovskou výhodou je podpora výrobců i provozovatele, kde většina kitů přichází s celosvětově platnou licencí pro připojení na 1 rok zdarma kdekoliv na světě, pokud použiju správné pásmo, pro danou zemi.

Drobnou nevýhodou DIY licence je omezení na přenosy o délce 12 byte a 100 zpráv denně. Komunikace je obousměrná a pro velkou část projektů venkovního monitorování plně dostačující.

Energetická náročnost obzvlášť v porovnání s WiFi, nebo GSM technologií je řádově nižší. Pro ukázku parametry modemu eRIC-SIGFOX, kde vidíme, že klidový odběr je v mikro ampérech a ve spánku padá řádově na 100 nA. Špičkový odběr v průběhu vysílání je 50 mA (v porovnání s 2 A u GSM)


Power Consumption
Power Supply range 2.5V – 5.0V
Ultra-low power consumption:
Standby mode current: 0.5 mA
Sleep mode current: 1.3 µA
Deep sleep mode current: 100 nA
Continuous radio Rx mode: 10 mA
Continuous radio Tx mode: 49 mA @ 14 dBm
Charge required to send a Sigfox packet at 14 dBm output power: 0.28C

Obrázek 7: eRIC-SIGFOX parametry (výstřižek [6])

2 Ukázková řešení

Z přehledu uvedeného v předchozí kapitole se jako nejsnadnější řešení jeví řešení postavené na technologii SigFox. Omezení na datech nás pro dané aplikace nijak neomezuje, nízká spotřeba a provozní poplatky, spojené s velmi dobrým pokrytím v rámci ČR a mnou vybraných lokalit jako optimální.

Pro praktickou ukázku jsem si připravil dva příklady. Každý z nich je postaven na zařízení od různých výrobců a je tedy třeba zvolit mírně odlišné způsoby práce s nimi. Nespornou výhodou vybraných produktů jsou bez pochyby přehledné a obsáhlé dokumentace, které pomohou i začátečníkům používat vybrané moduly.

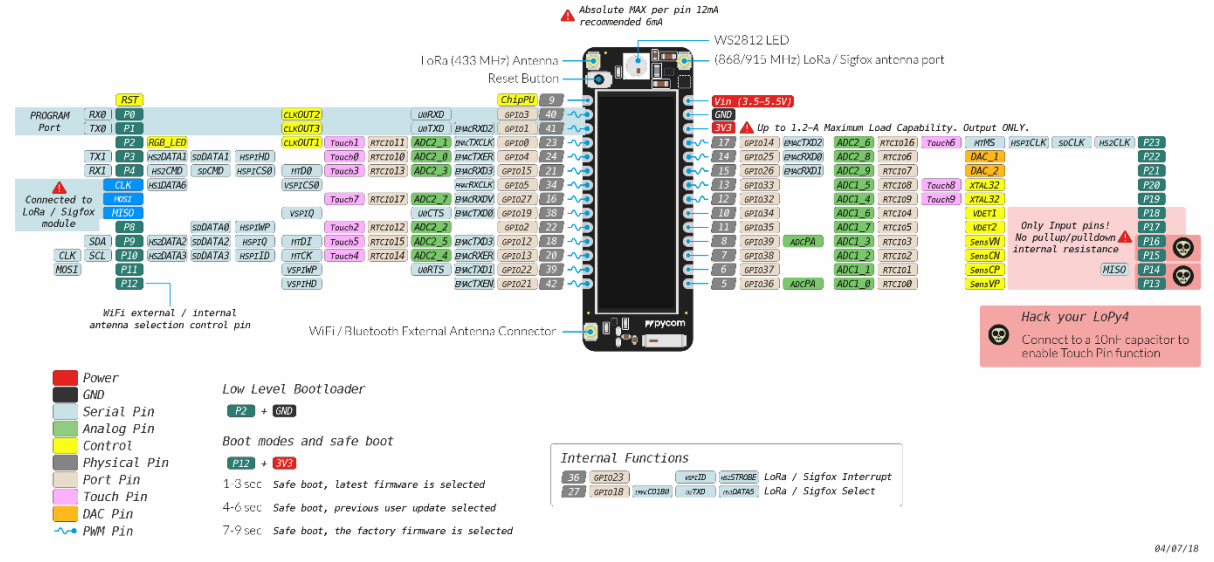
2.1 Pycom(Krmelec)

Jednou z typických představitelů, kde lze zapojit IoT sítě je myslivectví. Toto odvětví ještě zcela nevyužilo potenciál, který vyspělé technologie nabízejí. Jistě lze namítnout, že například fotopast je dnes výbavou téměř každého zapáleného myslivce, ale ne vždy je potřeba zpracovávat tak velké množství informací. Pro tuto ukázku jsem si tedy zvolil odlišný způsob jak technologicky podpořit tuto činnost.

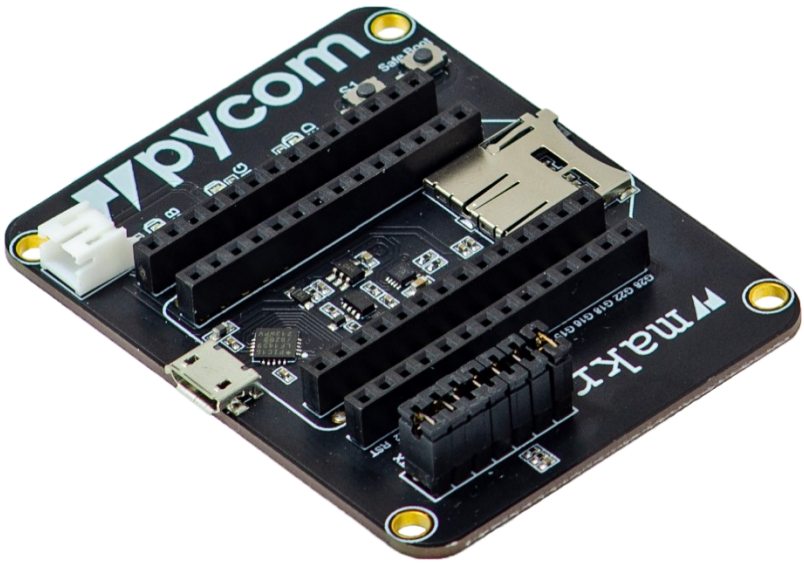
V zimních měsících je zapotřebí pomáhat zvěři a doplňovat do krmelců seno, sypká krmiva nebo solný liz. Živočichové ovšem nemají předem určené dávkování, a proto je doplňování zásobníků ošemetnou záležitostí. V příkladu se zaměříme na základní monitorování množství potravy a okolních meteorologických podmínek.

Ukázka je postavena na modulu od firmy Pycom a to konkrétně LoPy4, který je osazen výkonným dvoujádrovým 32-bitovým procesorem Xtensa LX6. Tato jednotka ovšem není jedinou zajímavou součástkou. Základní deska také obsahuje komponentu, která umožňuje připojení k bezdrátovým sítím Lora a Sigfox. Je zde umožněna i konektivita s více známými standardy jako Wifi a Bluetooth díky internímu vysílači. Všechna tato spojení lze provádět díky konektorům U.FL, sloužících k přidání externích antén. Jako poslední zde nalezneme 8MB flash a 4MB SRAM paměť, RGB diodu a tlačítko pro tvrdý reset.

Na zařízení nalezneme 23 vstupně výstupních pinů a každý z nich obsahuje množství alternativních funkcí jako např. I^2C a SPI sběrnice, PWM, ADC či DAC převodníky a podobně. K modulu jsem připojil Expansionboard v3.1, která umožní nejen snadnější komunikaci s počítačem přes USB nebo nahrávání programu, ale také podporuje vložení microSD karty a zásuvný konektor pro baterii.



Obrázek 8: Rozložení pinů [7]

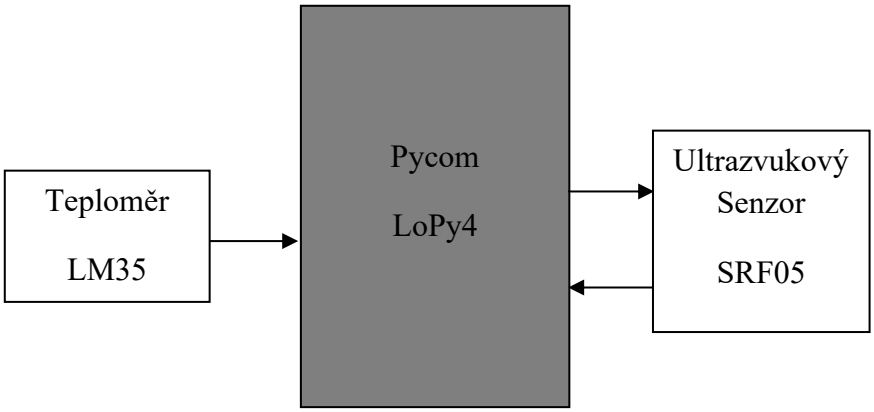


Obrázek 9: Rozšiřující deska [7]

Pro měření teploty okolí jsem si vybral jednoduchý tří pinový teploměr LM35, který poskytuje analogovou hodnotu napětí. Poté jednoduchým přepočtem lze získat naměřenou teplotu. Dle zvoleného postupu je možné získat výsledek ve stupních Celsia nebo Fahrenheita. Součástka má skutečně snadno zapamatovatelný poměr 10 mV/°C. Teploměr má velký rozsah od -55 do 150 °C, nicméně při 25 °C se hodnota může pohybovat ±0.5 °C. Je tedy určený spíše pro orientační měření. Napájecí napětí součástky činí od 4 do 30 V.

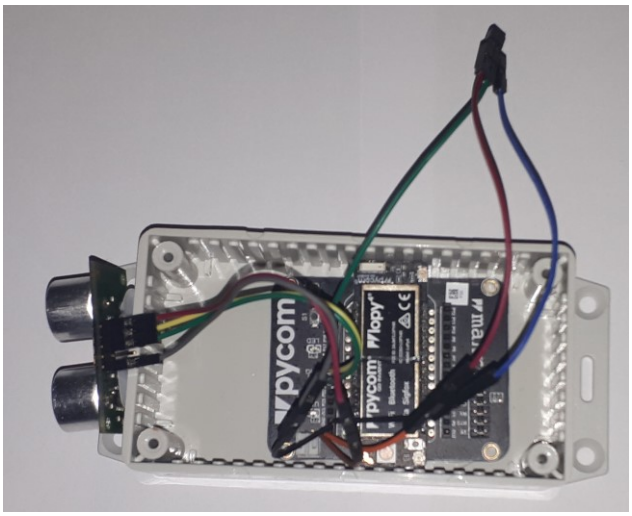
Další měřenou veličinou v našem příkladu je sypké krmivo, které je častým obsahem krmelců. Pro snímání jsem si tedy zvolil ultrazvukový senzor SRF05. Modul lze provozovat ve dvou režimech. Lze použít pro řídící a snímací signály pouze jeden kolík, ale rozhodl jsem se pro dvou pinové provedení. Spouštěcí a měřící kolíky jsou připojeny na rozdílné svorky a není potřeba měnit nastavení pinu v programu z výstupního na vstupní a naopak.

Zařízení při přijetí řídícího signálu vyšle zvukovou vlnu a nastaví pin odezvy na logickou jedničku. Po odrazu od blízkého objektu a přijetí vlny připne výstupní pin opět k nule. Doba, po kterou bylo napětí vyšší, se měří a po výpočtu určíme vzdálenost uraženou zvukovým pulsem. Senzor má rozsah měření od 1 po 400 cm. A napájen je stejnosměrnými 5 volty.



Obrázek 10: Blokové schéma zapojení

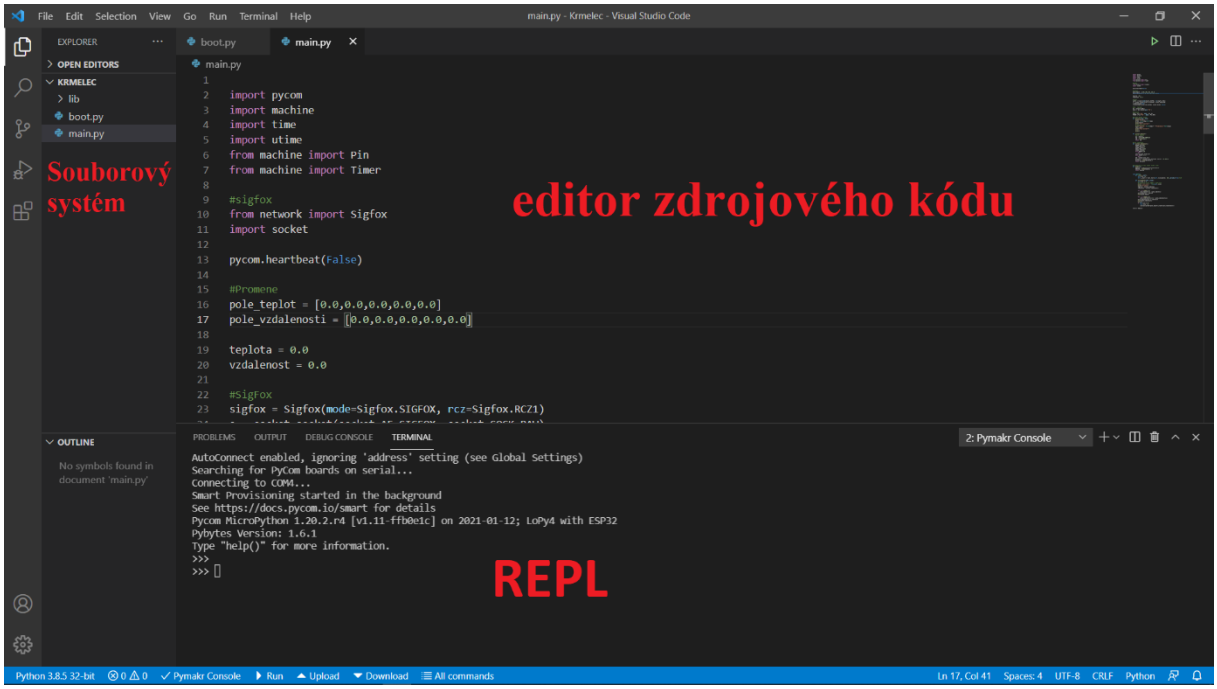
Z výše uvedeného blokového schématu lze vidět propojení všech modulů k řídící desce. Jako cestu informací jsem použil síť Sigfox, protože u těchto údajů není zapotřebí velkých objemů dat nebo velké rychlosti přenosu. Technologie také umožňuje operátorovi rozlehlé pokrytí a je tedy použitelná na většině území České republiky.



Obrázek 11: Hotové zapojení s modulem pycom

K vývoji softwaru jsem použil prostředí Visual studio code od firmy Microsoft. Zde bylo nutné přidat rozšíření Pymakr, které umožní komunikaci s mikroprocesorem. Jako programovací jazyk se pro modul využívá Micropython. Lze ho používat ve dvou různých režimech. Prvním z nich je klasicky napsaný program, nahrávající se do zařízení jako celek. Druhým už ne tak obvyklým způsobem je prostředí REPL (Read-eval-print loop), které

vytváří dojem terminálového přístupu, tedy že po zadání příkazu je ihned vykonán a očekává se další akce.



Obrázek 12Vývojové prostředí Visual studio Code

```
import pycom
import machine
import time
import utime
from machine import Pin
from machine import Timer

#sigfox
from network import Sigfox
import socket
```

Na začátku programu je nutné zavedení různých potřebných modulů, aby bylo možné použít např. alternativní funkce pinů, uspání modulu nebo nastavení parametrů pro komunikaci.

```
pycom.heartbeat(False)
```

Tento příkaz vypne výchozí nastavení blikání ledky pro signalizaci aktivity modulu.

```
pole_teplot = [0.0,0.0,0.0,0.0,0.0]
pole_vzdalenosti = [0.0,0.0,0.0,0.0,0.0]
teplota = 0.0
vzdalenost = 0.0
```

Zde lze vidět založení některých proměnných používaných v programu. V jazyce Micropython je prováděno dynamické typování, a proto není za potřebí deklarovat je na začátku programu. Vytvoření bylo nutné kvůli globálnímu použití v programu a nastavení velikostí polí.

```
sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ1)
s = socket.socket(socket.AF_SIGFOX, socket.SOCK_RAW)
s.setblocking(True)
s.setsockopt(socket.SOL_SIGFOX, socket.SO_RX, False)
```

Tato část kódu zajišťuje nastavení Sigfox modulu do režimu RCZ1 (zóna pokrývající Evropu) a socketu pro správnou komunikaci mezi zařízením a sítí. Poslední řádek zajistí, že data mohou být pouze odesílána, nikoli přijímána.

```
adc = machine.ADC()
anpin = adc.channel(pin='P16')
```

Nastavení pinu P16 na funkci AD převodníku.

```
ECHO = Pin('P20', mode = Pin.IN)
TRIGER = Pin('P21' , mode = Pin.OUT)
```

Příprava pinů P20 a P21 pro komunikaci s ultrazvukovým senzorem.

```
def posilani(temp, rang):
    print("Prevod")
    text = str(temp)+str(rang)
    print(text)
    print("Start vysilani")
    s.send(text)
    print("Teplota:" + str(temp)+" "+"Vzdalenost:"+str(rang))
    time.sleep(1)
    print("Stop vysilani")
    print()
    print()
```

Tato funkce přijímá dvě hodnoty typu float, které poté převede na textový řetězec a uloží je do proměnné zprava. Tu pak odešle na server Sigfox.

```
def vycteni_teplota():
    val = anpin()
    tep = val*1100/(4096*10)
    tep = round(tep,1)
    return tep
```

Vyčte hodnotu z analogového pinu. Pomocí referenčního napětí ADC a rozlišení převodníku lze vypočítat snímanou teplotu. Poté se číslo zaokrouhlí na jedno desetinné místo a navrátí hodnotu proměnné tep.

```
def vycteni_vzdalenost():
    TRIGER.value(0)
    time.sleep_us(2)
    TRIGER.value(1)
    time.sleep_us(10)
    TRIGER.value(0)
    while ECHO()==0:
        pass
    start = utime.ticks_us()
    while ECHO() == 1:
        pass
```

```
    end = utime.ticks_us()
    distance = ((utime.ticks_diff(end, start)) * 0.034)/2
    distance = round(distance,1)
    return distance
```

Tato část kódu zjišťuje měřenou vzdálenost objektu. V první řadě se ujistíme, že pin P21 je nastaven v na logickou nulu a poté ho nastavíme do jedničky po dobu minimálně 10 mikro sekund, aby senzor věděl, že má začít měřit. Poté se opět nastaví nízká úroveň a spustí se smyčka hlídající vstup P20. Jakmile je zaznamenána logická 1 uloží se čas mikroprocesoru do proměnné start a opět se čeká na změnu napětí. Po ukončení smyčky se zaznamená čas do end. Provede se výpočet a zaokrouhlení vzdálenosti a navrátí se hodnota.

```
def prumer(x):
    hodnota = (x[0]+x[1]+x[2]+x[3]+x[4])/5
    hodnota = round(hodnota, 1)
    return hodnota
```

Funkce obdrží pěti prvkové pole floatů a navrátí zaokrouhlený průměr hodnot.

```
class Hodiny:
    def __init__(self):
        self.tick = 0
        self.__alarm = Timer.Alarm(self.tick_handler, 180, periodic=True)
 #180

    def tick_handler(self, alarm):
        self.tick += 1
        print("Tick cislo:  " +str(self.tick))
        print("vycitani")
        teplota = vycteni_teplota()
        vzdalenost = vycteni_vzdalenost()

        for i in range(1,5):
            pole_teplot[i-1] = pole_teplot[i]
        pole_teplot[4] = teplota
        print(pole_teplot)

        for i in range(1,5):
            pole_vzdalenosti[i-1] = pole_vzdalenosti[i]
        pole_vzdalenosti[4] = vzdalenost
        print(pole_vzdalenosti)
        print()
        if self.tick == 5:
            self.tick = 0
            posilani(prumer(pole_teplot),prumer(pole_vzdalenosti))

clock = Hodiny()
```

Pro hlavní část programu jsem vytvořil třídu Hodiny, která obsahuje dvě funkce. V části __init__ probíhá inicializace a nastavení interního přerušení po třech minutách vyvoláním funkce tick_handler. Také přiřadí proměnné tick hodnotu nula. Druhá definice obsahuje hlavní logickou strukturu programu. Dochází k navýšení proměnné tick, vyčtení hodnot senzorů a zapsání do příslušných proměnných. Smyčky for přeskládají pole a na poslední pozici přidají aktuální teplotu a vzdálenost. Na konci se už jen vyhodnotí, zda je pole zcela naplněno nejnovějším obsahem. Jestliže je to pravda vynuluje tick a zavolá funkce prumer a posilani pro předání informací serveru. Posledním příkazem se spustí časovač.

2.2 Arduino(Včelín)

Další zajímavou oblastí, kde lze tyto technologie využít je včelařství. Tuto činnost je nutno vykonávat tam, kde je pro dělnice vhodné prostředí a to nemusí být vždy blízko domova. Také nevyžaduje mnoho zásahů, a proto ji často lidé provozují na chalupách či zcela mimo civilizaci. Takové informace z úlu mohou být velmi ku prospěchu a pomoci práci optimalizovat.

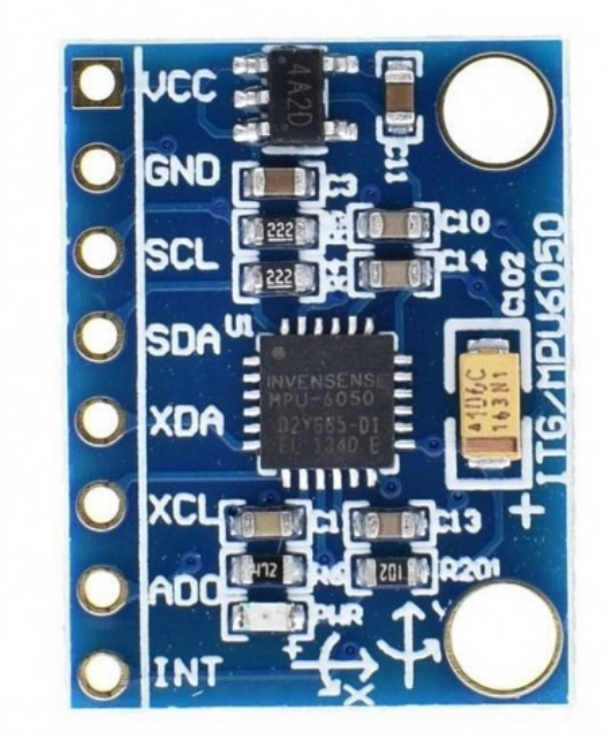
Možností pro monitorování zde nalezneme spoustu. Zajímavou skutečností může být, zda jsou včely stále v úlu nebo se vyrojily. Tuto variantu lze sledovat snímačem zvuku. Pro moji ukázku jsem si ovšem vybral mírně odlišnou oblast zájmu. Med je v současné době ceněnou komoditou a proto není možné při modernizaci zapomenout na zabezpečení proti ztrátě. Ke znehodnocení může dojít nejen vinou cizího člověka, ala také nedbalostí.

Příklad jsem postavil na Arduinu MKRFOX1200, která je řízena 32 bitovým mikrokontrolérem ARM Cortex SAMD21. Deska je také vybavena modulem pro komunikaci umožňujícím spojení se sítí Sigfox. Tato komponenta umožňuje ovšem provoz pouze v Evropě. Dále zde nalezneme konektor pro připojení vnější antény, signalizační led diodu, tlačítko tvrdého resetu, svorkovnici pro napájení z baterií (akumulátorů) a USB sloužící ke komunikaci s počítačem, programování čipu či napájení.

Zařízení obsahuje 15 programovatelných vstupně výstupních digitálních pinů. Další sloty jsou určeny pro analogové hodnoty. Použít zde lze 7 AD převodníku, z nichž jeden může být přepnut na funkci DAC. Mikrokontroler nabízí i komunikační sběrnice UART, I^2C nebo SPI. Naleznout zde můžeme i PWM či možnosti externího přerušení.

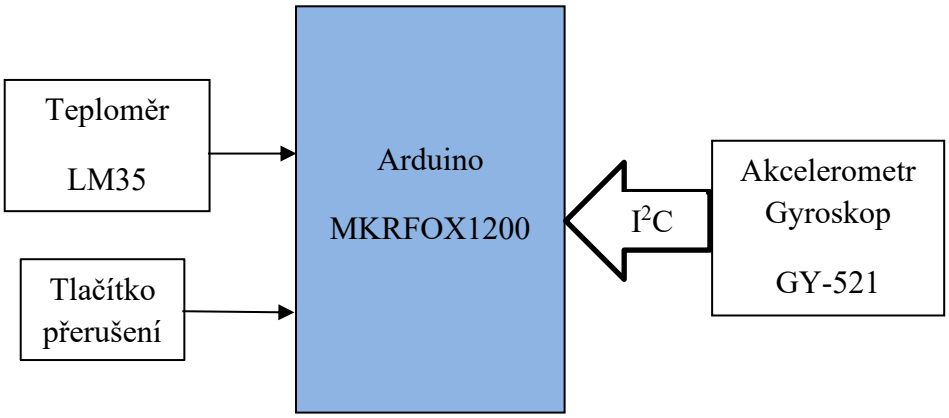
Obrázek 13: modul MKRFOX1200 se SigFox modemem [8]

Pro zabezpečení jsem se rozhodl použít GY-521. Tento modul je osazen čipem MPU-6050, který slouží jako akcelerometr a zároveň gyroskop. Toto zařízení zaznamenává pohyb a orientaci úlu, kdyby se jej někdo pokoušel odnést nebo selhala nosná konstrukce a včelín změnil svůj náklon. Komunikace probíhá pomocí I^2C sběrnice. Stanice se při komunikaci chová jako slave, to znamená, že nikdy neiniciuje komunikaci.



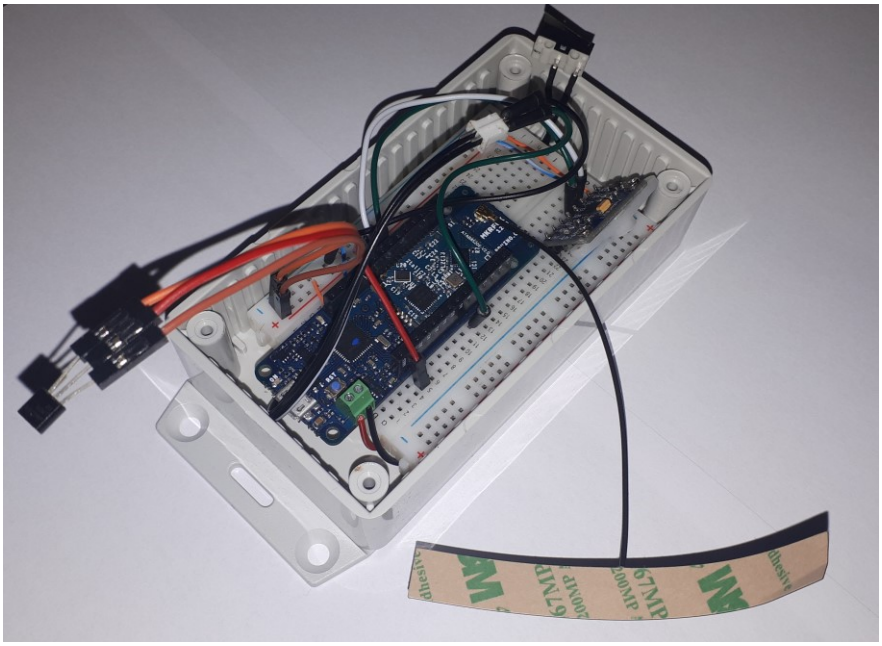
Obrázek 14: Inerciální modul [9]

Další možností může být zcizení pouze medových pláství. Na to je ovšem zapotřebí úl otevřít. Pro tuto variantu jsem do obvodu přidal spínač (tamper), který lze umístit pod horní okraj a zajistit tak rozpojení kontaktu v případě zvednutí víka. Pro zvýšení informovanosti používám také teploměr LM35. Ten byl již popsán v předchozím příkladu.



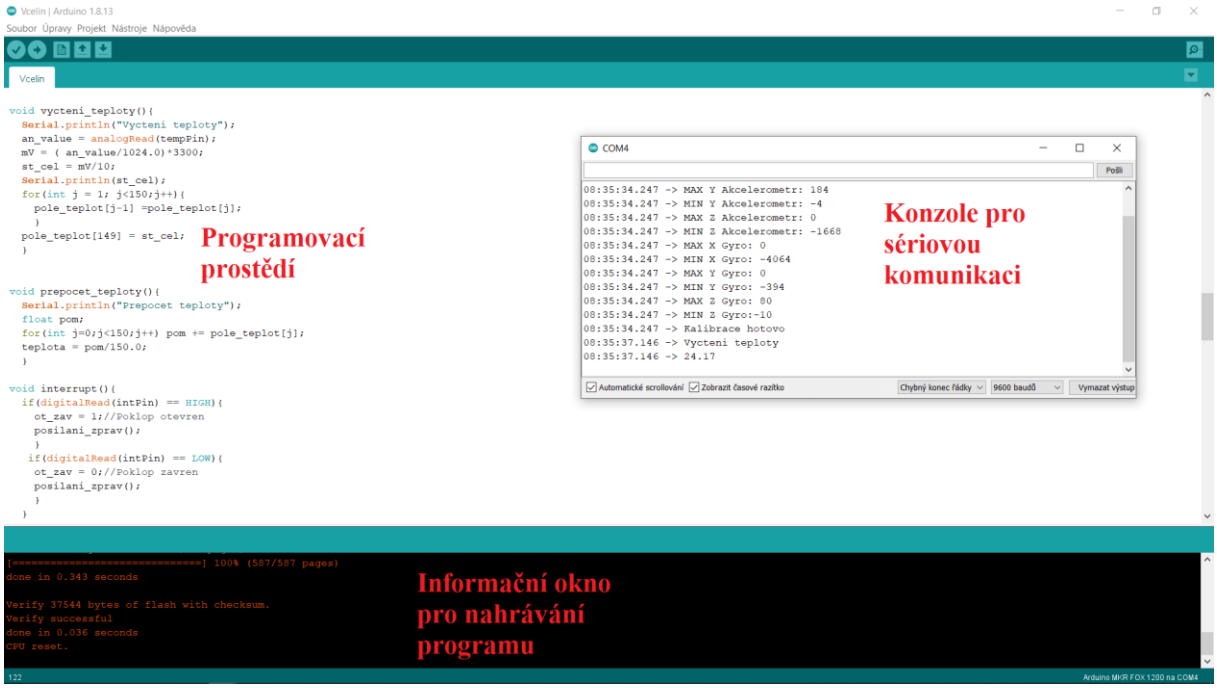
Obrázek 15: Blokové schéma zapojení

Blokové schéma ukazuje, jakým způsobem jsou periferie připojeny k řídící desce. Pro tento typ modulu jsem se rozhodnul kvůli možnosti užití sítě Sigfox, která nabízí zajímavé možnosti přeposílání informací, pomocí zpětného volání, na jinou službu v internetu.



Obrázek 16: Hotové zařízení

K vývoji softwaru jsem využil Arduino IDE. Prostředí je přímo určeno k programování těchto zařízení a využívá vlastního jazyka Wiring. Editor také obsahuje možnost sériového teminálu pro přímou komunikaci mezi počítačem a vývojovou platformou.



Obrázek 17: Arduino IDE

Doplnění instalace do Arduino IDE.

```
#include <SigFox.h>
#include <ArduinoLowPower.h>
#include <Wire.h>
```

V úvodní části je nutné naimportovat potřebné knihovny pro snížení spotřeby energie vypnutím některých periferií, komunikaci s modulem Sigfox nebo výměnu informaci po sběrnici I^2C.

```
String text;
uint8_t ot_zav;
uint8_t nove_prekroceni = true;
//Preruseni
uint8_t intPin = 0;

//Teplomer
uint8_t tempPin = 0;
uint16_t an_value;
float mV;
float st_cel;
float pole_teplot[150];
float teplota;

//Akcelerometr, Gyroskop
const int MPU=0x68;
uint8_t Akce_Gyro = 0;  //1-Prekroceni mezi
uint8_t pocitadlo_prekroceni = 0;

int16_t AkceX;
int16_t AkceY;
int16_t AkceZ;
int16_t GyroX;
int16_t GyroY;
```

```
int16_t GyroZ;
  //meze
  int16_t AkceX_MAX = 0;
  int16_t AkceX_MIN = 0;
  int16_t AkceY_MAX = 0;
  int16_t AkceY_MIN = 0;
  int16_t AkceZ_MAX = 0;
  int16_t AkceZ_MIN = 0;
  int16_t GyroX_MAX = 0;
  int16_t GyroX_MIN = 0;
  int16_t GyroY_MAX = 0;
  int16_t GyroY_MIN = 0;
  int16_t GyroZ_MAX = 0;
  int16_t GyroZ_MIN = 0;
```

Zde probíhá založení různých proměnných používaných k ukládání a výpočtům.

```
void init_AkGy();
void kalibrace_AkGy();
void vycteni_AkGy();
void vyhodnoceni_AkGy();
void vycteni_teploty();
void prepocet_teploty();
void posilani_zprav();
void interrupt();
```

Inicializace funkcí, které pomohou k přehlednosti programu.

```
void setup() {
  Serial.begin(9600);
  pinMode(intPin, INPUT_PULLUP);
  init_AkGy();
  Serial.println("Init hotovo");
  kalibrace_AkGy();
  Serial.println("Kalibrace hotovo");
  if(digitalRead(intPin) == HIGH) ot_zav = 1;
  else ot_zav = 0;
  attachInterrupt(intPin, interrupt, CHANGE);
}
```

Funkce `setup()` se provádí pouze jednou na začátku programu. Obsahuje nastavení rychlosti 9600 baudů pro přenos dat po sériové lince. Uvedení pinu A0 do stavu vstupu se zapnutým pull-up rezistorem. Inicializaci modulu GY-521. Kalibraci maximálního vychýlení systému. Rozhodnutí hodnoty proměnné `ot_zav` podle stavu přivedeného na `intPin`. Zavedení externího přerušení na základě změny stavu svorky A0 a následného vyvolání funkce `interrupt`.

```
void loop() {
  uint8_t k = 0;
  for(int i = 0; i<9000;i++){
    if(k == 60){
      vycteni_teploty();  //Vycita jednou za 6s
      k =0;
      }
    vycteni_AkGy();
    vyhodnoceni_AkGy();
    if(pocitadlo_prekroceni >= 18) {
      Akce_Gyro = 1;
      Serial.print("Prekroceni mezi-->");
      if(nove_prekroceni){
```

```
        posilani_zprav();
        nove_prekroceni = false;
        }
      i = 0;
      k = 0;
      pocitadlo_prekroceni = 0;
      Serial.print("Accelerometer: ");
      Serial.print("X = "); Serial.print(AkceX);
      Serial.print(" | Y = "); Serial.print(AkceY);
      Serial.print(" | Z = "); Serial.println(AkceZ);

      Serial.print("Gyroscope: ");
      Serial.print("X = "); Serial.print(GyroX);
      Serial.print(" | Y = "); Serial.print(GyroY);
      Serial.print(" | Z = "); Serial.println(GyroZ);
      Serial.println(" ");
      delay(30000);
      }
    else{
      nove_prekroceni = true;
      Akce_Gyro = 0;
      }
    k++;
    LowPower.idle(100);
    }
  posilani_zprav();
}
```

V nekonečné smyčce `loop()` probíhá hlavní struktura programu. Je zde vložen cyklus `for`, který v 15 minutách udělá 150 měření teploty a 9000 kontrol stavu GY modulu. Po uplynutí této doby odešle data. Pomocí funkce `vyhodnoceni_AkGy()` se stanoví překročení mezí. To při předchozím ustáleném stavu způsobí změnu proměnné a předčasnou komunikaci. Zařízení pracuje ve 100 ms intervalech, mezi kterými je částečně uspáno z důvodů šetření energie.

```
void posilani_zprav(){
  Serial.println("Posila se zprava");
  prepocet_teploty();
  text = String(teplota,1)+String(Akce_Gyro)+String(ot_zav);
  Serial.println(text);
  SigFox.begin();
  SigFox.beginPacket();
  SigFox.print(text);
  SigFox.endPacket();
  delay(1000);
  }
```

Zde lze vidět zpracování proměnných do textového řetězce. Komunikace začíná inicializací knihovny a zahájení odesílání. Poté se vytiskne obsah proměnné text a proces se ukončí.

```
void vycteni_teploty(){
  Serial.println("Vycteni teploty");
  an_value = analogRead(tempPin);
  mV = ( an_value/1024.0)*3300;
  st_cel = mV/10;

  for(int j = 1; j<150;j++){
    pole_teplot[j-1] =pole_teplot[j];
    }
```

```
  pole_teplot[149] = st_cel;
  }
```

V této části kódu vyčítáme analogovou hodnotu z pinu A0. Poté užitím hodnoty referenčního napětí a rozlišení převodníku získáme velikost napětí. Následujícím přepočtem zjistíme naměřenou teplotu ve °C. Následující krok přeskládá pole, ve kterém jsou uloženy předešlé měření a na poslední pozici přidá aktuální stav.

```
void prepocet_teploty(){
  Serial.println("Prepocet teploty");
  float pom;
  for(int j=0;j<150;j++) pom += pole_teplot[j];
  teplota = pom/150.0;
  }
```

Dochází ke zprůměrování hodnot uložených v poli.

```
void interrupt(){
  if(digitalRead(intPin) == HIGH){
    ot_zav = 1;//Poklop otevren
    posilani_zprav();
    }
   if(digitalRead(intPin) == LOW){
    ot_zav = 0;//Poklop zavren
    posilani_zprav();
    }
  }
```

Tato funkce je volána pouze v případě změny logické úrovně na pinu 0. Proběhne oprava stavu proměnné `ot_zav` a spustí se odesílání dat.

```
void vyhodnoceni_AkGy(){
  if(AkceX > AkceX_MAX||AkceX<AkceX_MIN)pocitadlo_prekroceni++;
  if(AkceY > AkceY_MAX||AkceY<AkceY_MIN)pocitadlo_prekroceni++;
  if(AkceZ > AkceZ_MAX||AkceZ<AkceZ_MIN)pocitadlo_prekroceni++;

  if(GyroX > GyroX_MAX||GyroX<GyroX_MIN)pocitadlo_prekroceni++;
  if(GyroY > GyroY_MAX||GyroY<GyroY_MIN)pocitadlo_prekroceni++;
  if(GyroZ > GyroZ_MAX||GyroZ<GyroZ_MIN)pocitadlo_prekroceni++;
  }
```

Při překročení maximálních rozsahů přičte jedničku k `pocitadlo_prekroceni`.

```
void vycteni_AkGy(){
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,12,true);
  AkceX=Wire.read()<<8|Wire.read();
  AkceY=Wire.read()<<8|Wire.read();
  AkceZ=Wire.read()<<8|Wire.read();
  GyroX=Wire.read()<<8|Wire.read();
  GyroY=Wire.read()<<8|Wire.read();
  GyroZ=Wire.read()<<8|Wire.read();
  }
```

Proběhne zahájení komunikace po sběrnici s modulem, navázání spojení s registrem 0x38 a čekání na odpověď. Hodnoty přijaté přes I^2C rozhraní se zpracují do 16 bitových proměnných.

```
void init_AkGy(){
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  }
```

Opět dojde k zahájení spojení s modulem a připojení k registru 0x6B. Zapsáním nul do této části způsobí reset senzoru. Poté je komunikace ukončena.

```
void kalibrace_AkGy(){
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,12,true);
  AkceX=Wire.read()<<8|Wire.read();
  AkceY=Wire.read()<<8|Wire.read();
  AkceZ=Wire.read()<<8|Wire.read();
  GyroX=Wire.read()<<8|Wire.read();
  GyroY=Wire.read()<<8|Wire.read();
  GyroZ=Wire.read()<<8|Wire.read();
  AkceX_MAX = AkceX;
  AkceX_MIN = AkceX;
  AkceY_MAX = AkceY;
  AkceY_MIN = AkceY;
  AkceZ_MAX = AkceZ;
  AkceZ_MIN = AkceZ;

  GyroX_MAX = GyroX;
  GyroX_MIN = GyroX;
  GyroY_MAX = GyroY;
  GyroY_MIN = GyroY;
  GyroZ_MAX = GyroZ;
  GyroZ_MIN = GyroZ;

  for(int i = 0; i<1500; i++){
    Wire.beginTransmission(MPU);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU,12,true);
    AkceX=Wire.read()<<8|Wire.read();
    AkceY=Wire.read()<<8|Wire.read();
    AkceZ=Wire.read()<<8|Wire.read();
    GyroX=Wire.read()<<8|Wire.read();
    GyroY=Wire.read()<<8|Wire.read();
    GyroZ=Wire.read()<<8|Wire.read();

    if(AkceX>AkceX_MAX)AkceX_MAX = AkceX;
    if(AkceX<AkceX_MIN)AkceX_MIN = AkceX;
    if(AkceY>AkceY_MAX)AkceY_MAX = AkceY;
    if(AkceY<AkceY_MIN)AkceY_MIN = AkceY;
    if(AkceZ>AkceZ_MAX)AkceZ_MAX = AkceZ;
    if(AkceZ<AkceZ_MIN)AkceZ_MIN = AkceZ;

    if(GyroX>GyroX_MAX)GyroX_MAX = GyroX;
    if(GyroX<GyroX_MIN)GyroX_MIN = GyroX;
    if(GyroY>GyroY_MAX)GyroY_MAX = GyroY;
    if(GyroY<GyroY_MIN)GyroY_MIN = GyroY;
    if(GyroZ>GyroZ_MAX)GyroZ_MAX = GyroZ;
```

```
    if(GyroZ<GyroZ_MIN)GyroZ_MIN = GyroZ;
    delay(100);
    }

  Serial.print("MAX X Akcelerometr: ");
  Serial.println(AkceX_MAX);
  Serial.print("MIN X Akcelerometr: ");
  Serial.println(AkceX_MIN);
  Serial.print("MAX Y Akcelerometr: ");
  Serial.println(AkceY_MAX);
  Serial.print("MIN Y Akcelerometr: ");
  Serial.println(AkceY_MIN);
  Serial.print("MAX Z Akcelerometr: ");
  Serial.println(AkceZ_MAX);
  Serial.print("MIN Z Akcelerometr: ");
  Serial.println(AkceZ_MIN);

  Serial.print("MAX X Gyro: ");
  Serial.println(GyroX_MAX);
  Serial.print("MIN X Gyro: ");
  Serial.println(GyroX_MIN);
  Serial.print("MAX Y Gyro: ");
  Serial.println(GyroY_MAX);
  Serial.print("MIN Y Gyro: ");
  Serial.println(GyroY_MIN);
  Serial.print("MAX Z Gyro: ");
  Serial.println(GyroZ_MAX);
  Serial.print("MIN Z Gyro:");
  Serial.println(GyroZ_MIN);
  }
```

Poslední částí je kalibrace. Zde dochází k vyčtení hodnot senzoru pohybu a uložení maximálních výchylek, když je zařízení v „klidném“ stavu. Kalibrace pobíhá přibližně dvě a půl minuty od spuštění mikrokontroléru. V této době lze se systémem hýbat, ovšem pouze v rozsahu, který nemá být vyhodnocován jako překročení mezí a tedy odeslání zprávy.

2.3 Registrace a callback SigFox

Pro registraci zařízení do sítě Sigfox je nutné znát 2 parametry. Identifikátor (ID) je číslo, je pro každé zařízení unikátní. Slouží k rozpoznání zařízení při komunikaci. Druhou sekvencí je PAC. Jedná se o jednorázový kód sloužící k registraci zařízení. Poté je zneplatněn. Při změně správy zařízení je nutné použít PAC z původního účtu. Tento postup zaveden z důvodu bezpečnosti přenosu informace.

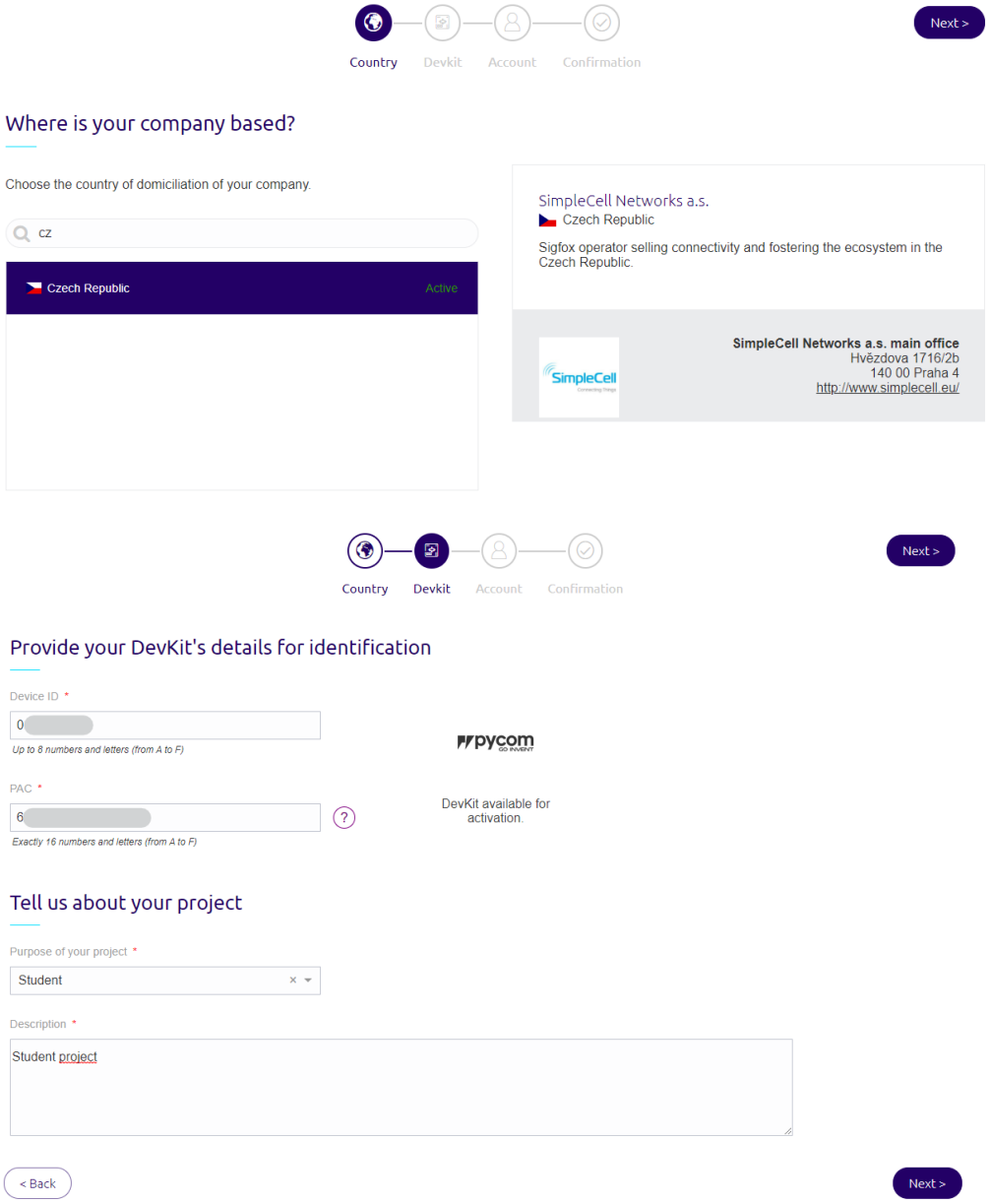
Pro získání klíčů od modulu Pycom jsem použil terminálový přístup REPL. Zadáním posloupnosti příkazů dojde k vytištění potřebných informací k registraci.

```
>>>from network import Sigfox
>>>import binascii
>>>sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ1)
>>>print(binascii.hexlify(sigfox.id()))
b'0*******'
>>>print(binascii.hexlify(sigfox.pac()))
b'b***************'
```

V případě Arduino desky je třeba nahrát tento kód a parametry se poté vytisknou po sériové lince.

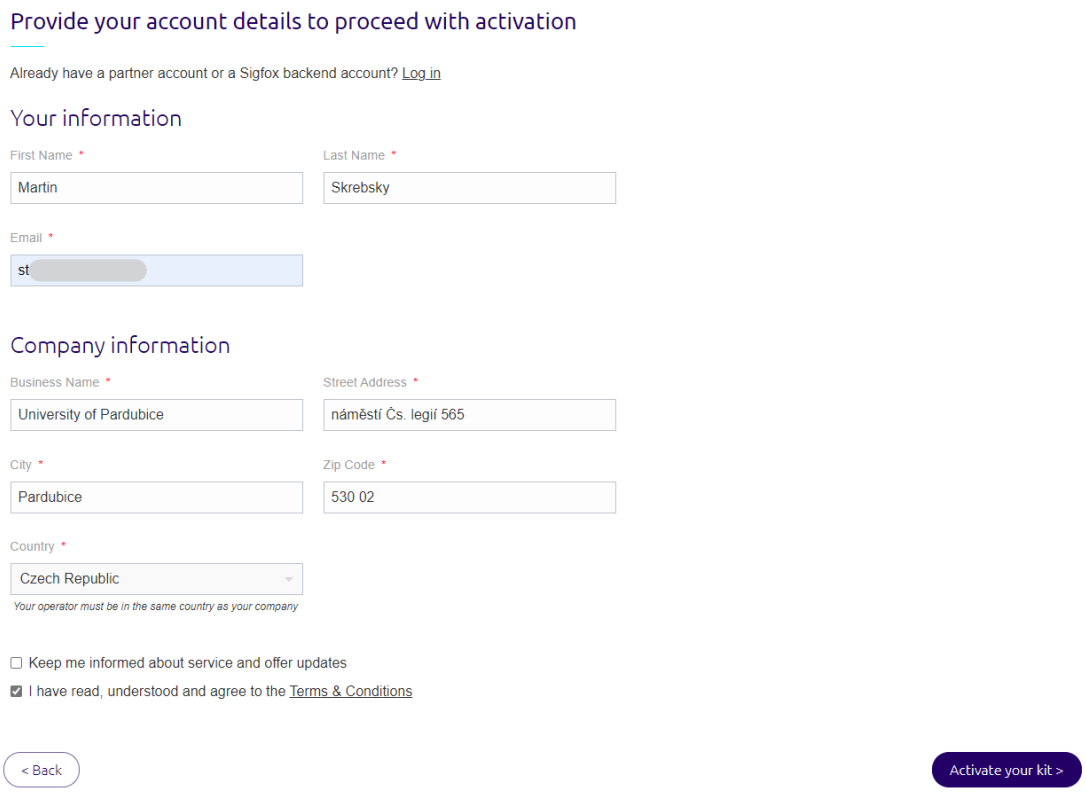
```
#include <SigFox.h>
void setup() {
  Serial.begin(9600);
  while(!Serial) {};
  if (!SigFox.begin()) {
    Serial.println("Chyba modulu");
    return;
  }
  String version = SigFox.SigVersion();
  String ID = SigFox.ID();
  String PAC = SigFox.PAC();
  Serial.println("ID= " + ID);
  Serial.println("PAC= " + PAC);
  delay(100);
  SigFox.end();
}
void loop() {}
```

Na stránkách buy.sigfox.com jsem provedl registraci. V prvních krocích zadává uživatel stát, ve kterém bude zařízení provozovat a identifikační čísla.



Obrázek 18: Registrace uživatele (nahoře) a registrace zařízení (dole) u provozovatele SigFox

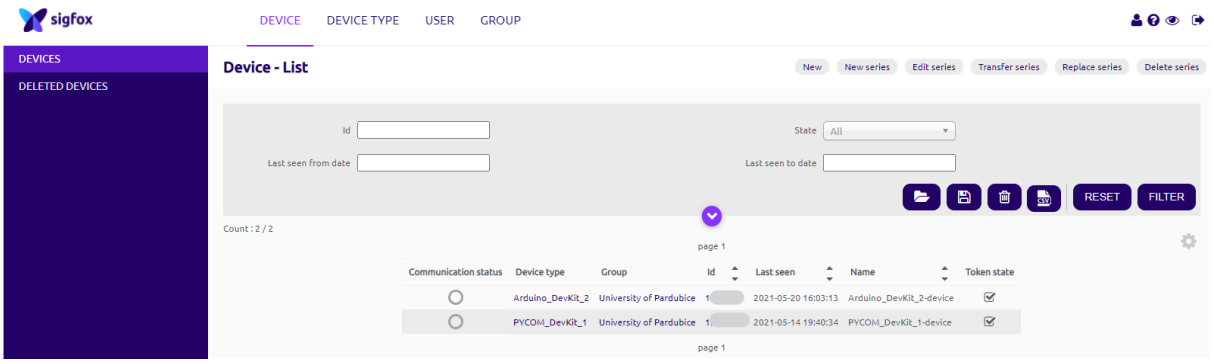
V následujícím kroku bylo nutné vytvořit účet Sigfox. Pro druhou registraci je nutné se přihlásit na stávající účet, aby bylo možné přepínat mezi zařízeními promocí jednoho přístupu.



Obrázek 19: Dokončení registrace

V zadaném e-mailu byl přidán odkaz pro zadání nového hesla a dokončení registrace.

Na webové stránce backend.sigfox.com s lze přihlásit k účtu a v sekci DEVICE se zobrazí všechna zaregistrovaná zařízení.



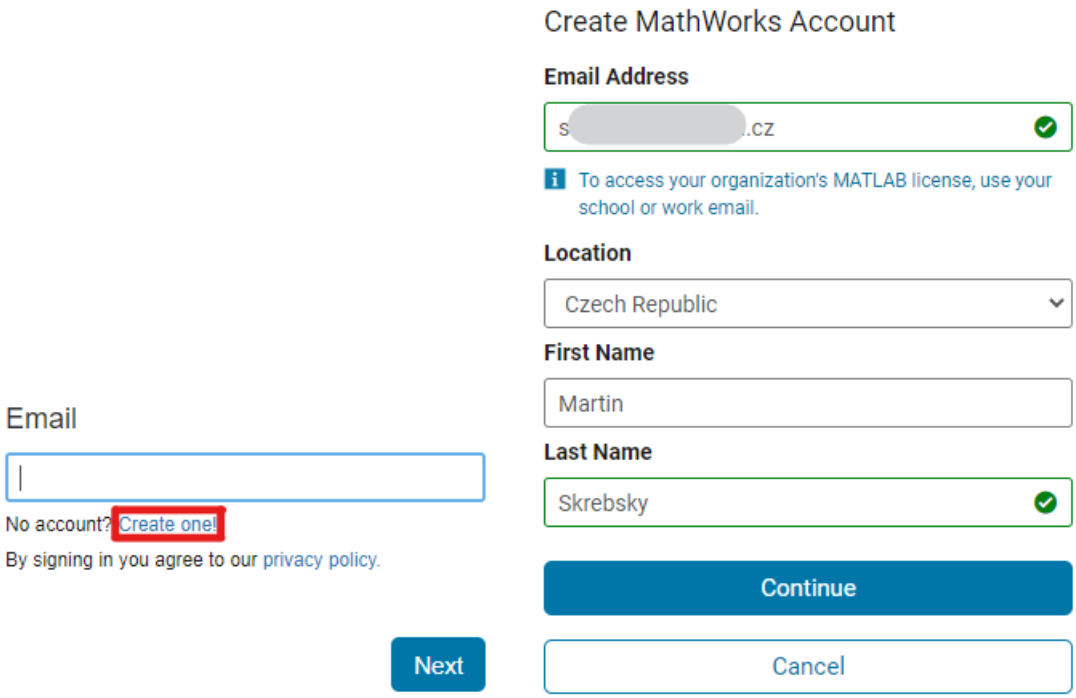
Obrázek 20: Přehled zařízení v síti poskytovatele

Tato stránka slouží jako uložiště pro všechny informace, které zařízení poslalo. Při kliknutí na parametr ID se otevře výpis informací o modulu. Vybráním možnosti MESSAGES lze zobrazit výpis přijatých zpráv.

2.4 Thingspeak

Jako zobrazovač dat jsem si zvolil službu Thingspeak, která je uživatelsky přívětivá a pro mé ukázky stačí bezplatná registrace. Na internetové stránce thingspeak.com jsem si založil účet kliknutím na Get Started For Free.

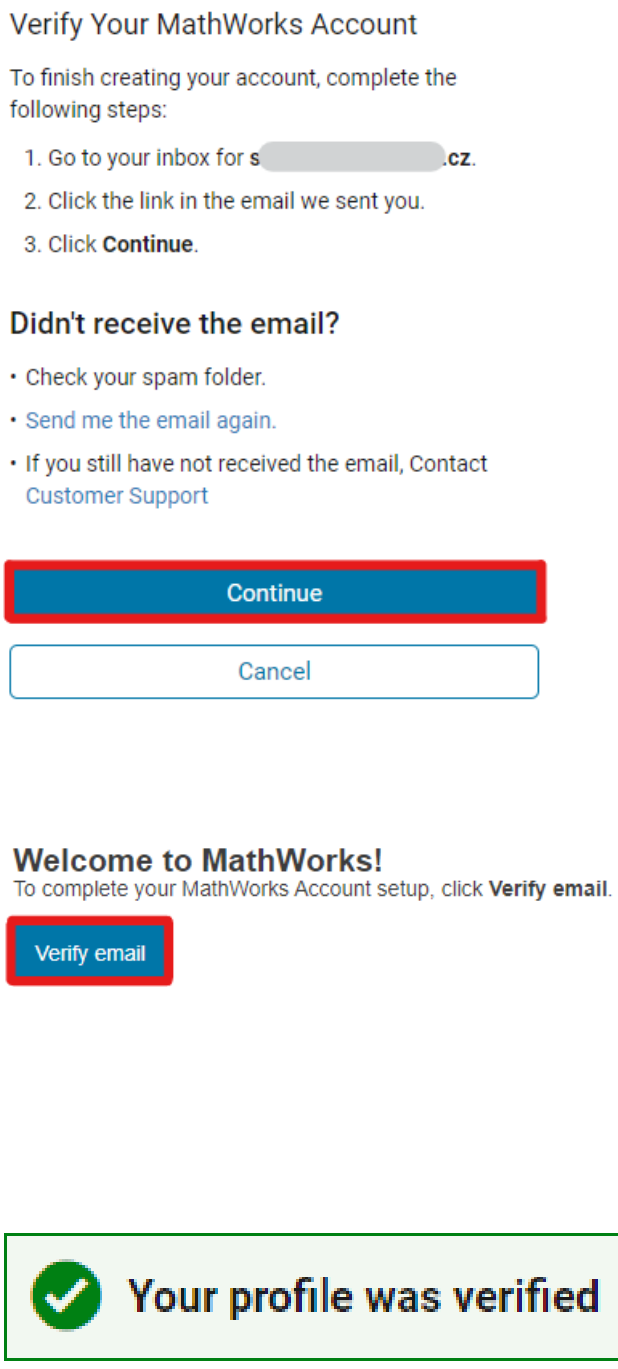
Obrázek 21: Registrace na webu thingspeak



Obrázek 22: Tvorba účtu u služby ThingSpeak

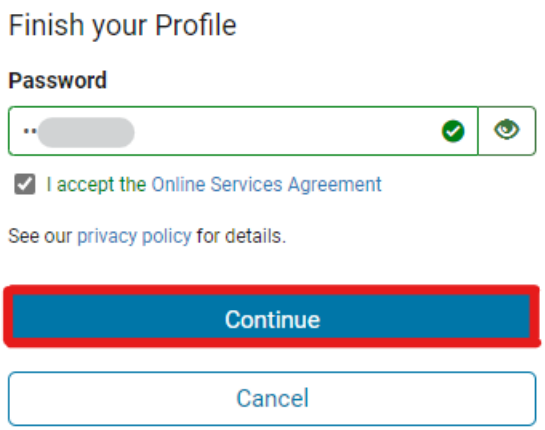
Po zvolení možnosti Create one a vyplnění osobních údajů lze pokračovat tlačítkem Continue.

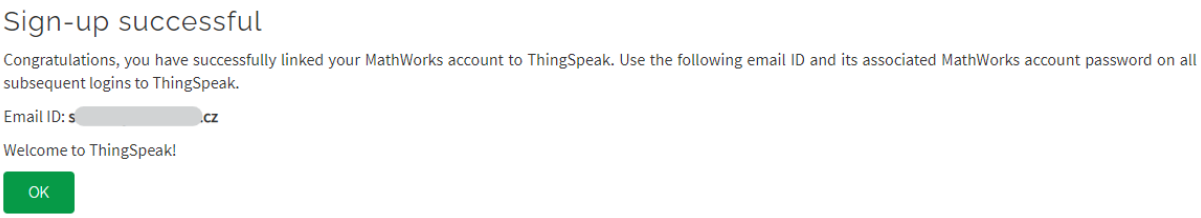
Dle návodu uvedeného na stránce je nutné přihlášení k zadanému e-mailovému účtu a potvrdit ověřovací zprávu tlačítkem Verify mail. Tato akce mě přesměrovala na stránku firmy Mathwokrs, která potvrdila vytvoření účtu. Poté jsem se navrátil do karty Thingspeak.



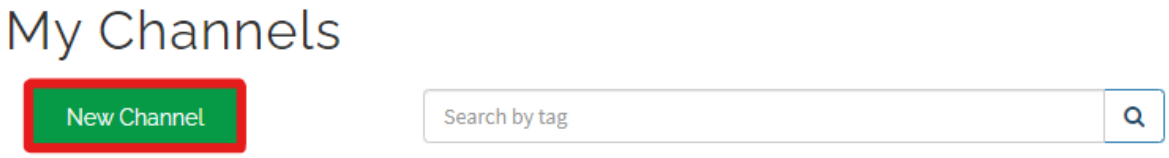
Obrázek 23: Postup registrace a ověření

Vybráním možnosti continue jsem byl vyzván k zadání hesla pro přihlášení.

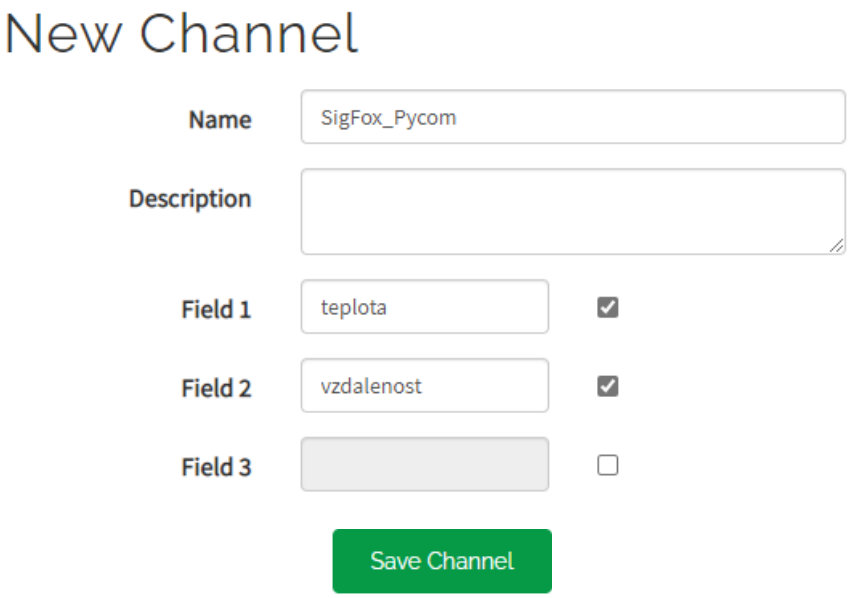


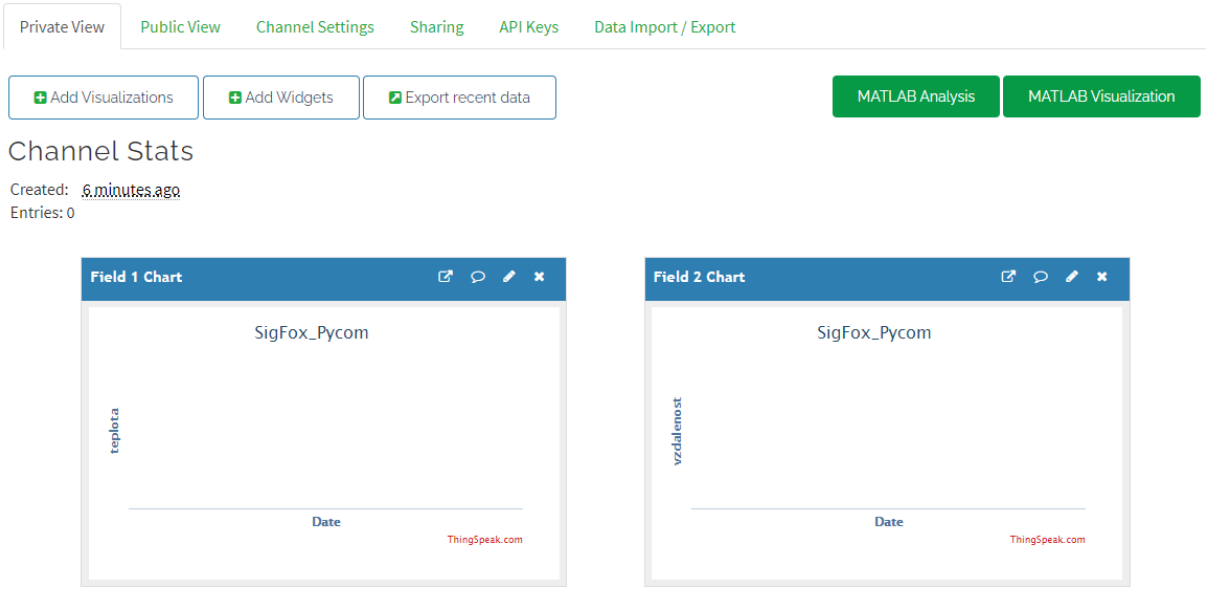


Pro příjem dat jsem si vytvořil nový kanál.



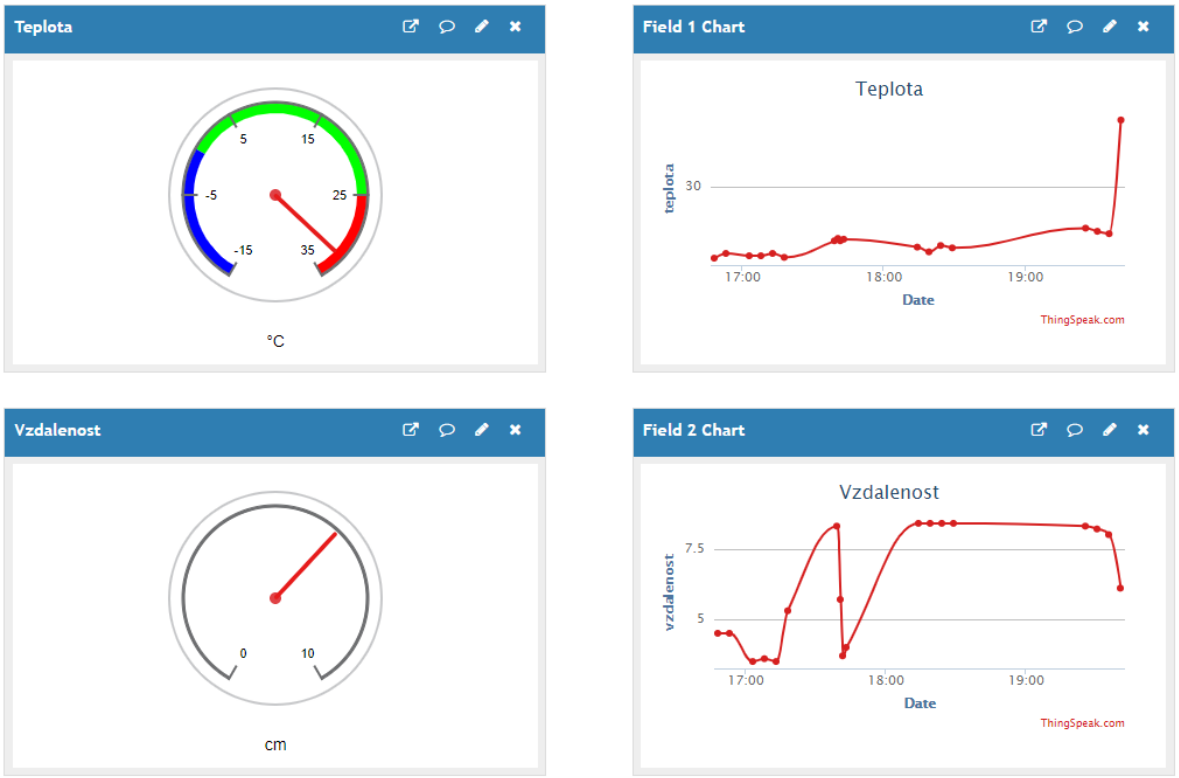
Povolením a vyplněním kolonek Field vytvořím požadovaný počet grafů zaznamenávajících přijaté hodnoty. Poté je nutné kanál uložit.





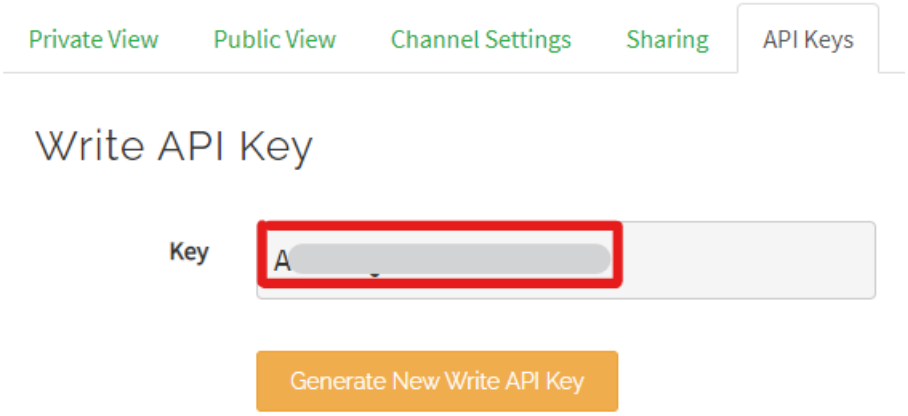
Obrázek 24: Tvorba kanálů

Lze přidat i Widgets pro lepší přehlednost. Konečná podoba kanálu pro příklad 1 je tedy:



Obrázek 25: Příklad widgetů

Nutnou součástí přeposílání je klíč API, který se nachází v záložce API Keys pod hlavičkou Write API Key:



Obrázek 26: API Key

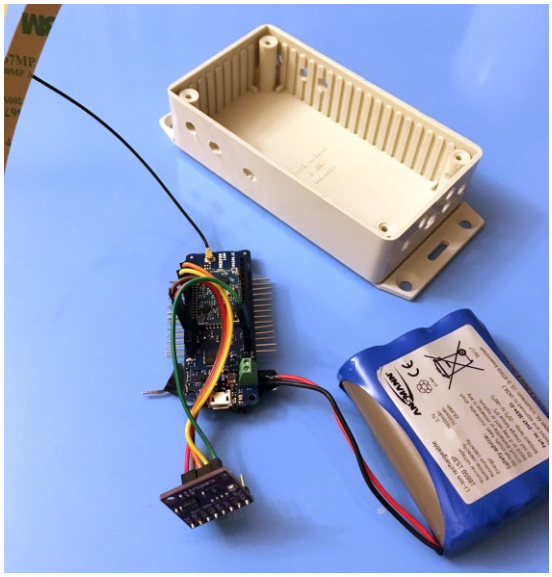
2.5 Měření meteorologických dat v oblasti honitby bez pokrytí mobilním signálem s vlastním úložištěm dat

Úkolem tohoto řešení je možnost zjištění aktuálních meteorologických dat v odlehlé honitbě Domašov u Šternberka. Je to typický příklad monitoringu environmentálních dat v lokalitě se špatným pokrytím mobilní sítí a bez možnosti externího napájení. Lokalita má své specifické mikroklima, které znemožňují odhad počasí z domova, a tedy vedou ke zbytečným cestám do lokality v době nevhodné k výkonu práva myslivosti.

Řešení tohoto příkladu je postaveno na modulu MKR FOX 1200 a environmentálním modulu CJMCU-8128 obsahující 3 kombinované snímače schopné měřit tlak, vlhkost, teplotu a koncentraci CO2. Vlhkost a teplota je měřena dvěma čidly nezávisle. Pro danou aplikaci je tento modul zbytečně vybavený, ale byl v danou chvíli k dispozici.

HW řešení se skládá pouze z modulu FOX 1200, CJMCU-8128 a lithiové baterie. Moduly jsou propojeny I2C sběrnicí a je připojen WAKEUP pin na pin A0 FOX 1200 modulu, pro řízení napájení senzoru.

Vše je vloženo do plastového boxu s vyvrtanými otvory na spodní straně a bočních stranách u vrcholu, pro zajištění cirkulace vzduchu, box je připevněn na severní straně mysliveckého zařízení, těsně pod střešním přesahem.



Obrázek 27: Modul pro monitoring počasí

Řešení pak leží v SW a správné implementaci spánkových režimů.

```
//spotreby 40mA mereni a seriovka (USB)
//72mA i s radiem
//20mA jen senzory + spánek
//<3mA vše spí  (mělo by být v uA)
```

Naměřená maximální spotřeba je 72mA, v tomto režimu se modul nachází na zlomky sekundy jednou za 15 minut, zbylí čas tráví modul ve spánkovém režimu, kde bohužel senzory i ve spánku díky nevhodnému zapojení odebírají zhruba 3mA. Jelikož je modul napájen lithiovým 3 článkem s kapacitou kolem 8 Ah což dává přes 100 dní provozu. Výměnou senzorů, případně osazení malým fotovoltaickým panelem by prodloužilo životnost na hranu životnosti baterie.

Obsluhu senzorů řeší knihovny jednotlivých čidel. Knihovny jsou psány objektově, pro každé čidlo se vytváří objekt, který řeší I2C komunikaci a přepočty hodnot na fyzikální jednotky.

```
#define WAKEUP A0
#include <SparkFunCCS811.h>
#include <SparkFunBME280.h>
#include <ClosedCube_HDC1080.h>
#include <SigFox.h>
#include <ArduinoLowPower.h>

// Objekty čidel
CCS811 myCCS811(0x5A);
ClosedCube_HDC1080 myHDC1080;
BME280 myBME280;

double tlak;
double teplota;
double vlhkost;
int CO2;
int Evoc;
int stat;

char vysledek[13];
```

Metoda setup() je v arduino prostředí spuštěna jednou a má za úkol provést inicializace, v tomto případě jsou nastaveny parametry čidel, spuštěn SIGFOX modul a sériová linka. Jelikož se používá sériová linka přes USB rozhraní procesoru, namísto fyzické sériové linky je nutné testovat, jestli je připojen a otevřen sériový port na PC. Program dává 15 sekund na připojení k sériové lince pro přímý výstup z programu na PC přes sériovou linku. Pokud není do 115 sekund připojena, jsou všechny operace se sériovou linkou ignorovány (standartní běh programu).

```
void setup() {
  pinMode(WAKEUP,INPUT);
  delay(15000);   //pockej jestli se pripoji seriovka
  Serial.begin(115200);
 //init cidel
  if(Serial) Serial.println("CCS811 + HDC1080 + BMP280 Init");
  myBME280.settings.commInterface = I2C_MODE;
  myBME280.settings.I2CAddress = 0x76;
  myBME280.settings.runMode = 3; //Normal mode
  myBME280.settings.tStandby = 0;
```

```
  myBME280.settings.filter = 4;
  myBME280.settings.tempOverSample = 5;
  myBME280.settings.pressOverSample = 5;
  myBME280.settings.humidOverSample = 5;
  myHDC1080.begin(0x40);
  if(Serial) Serial.println("HDC OK");
  myCCS811.begin();
  if(Serial) Serial.println("ccs OK");
  myBME280.begin();
  if(Serial) Serial.println("BME OK");
  if (!SigFox.begin()) {
   reboot();
  }
  if(Serial) Serial.println("sigfox OK");
  SigFox.end();
  if(Serial)SigFox.debug();
}
```

Samotný běh programu začíná příkazem SigFox.begin(), který aktivuje rádiový modul, SigFox.beginPacket(), zahájí přípravu paketu, data k odeslání se do rádia poté odesílají metodou print objektu SigFox, po naplnění daty se metodou endPacket() ukončí vkládání a rádiový modul se pokusí data odeslat, metoda vrací 0, pokud se odeslání podaří. Zavoláním metody SigFox.end() se rádiový modul přepne do úsporného režimu.

Vyčítání dat ze senzorů řeší metody read() jednotlivých objektů čidel. Data jsou metodou sprintf naformátována do textového pole ve formě stringu kde první 3 číslice jsou teplota, 6 číslic tlak a 3 číslice vlhkost. Je použit formát s pevným počtem číslic. Přičemž vše je zapsáno s pevnou řádovou čárkou. Teplota je na 3 platné číslice a odesílá se jako 10x násobek, teplota 21.3°C se odesílá jako 213, ostatní data jsou jen zaokrouhlena na celá čísla. Vzhledem k omezení na 12 byte, není CO2 odesíláno, jelikož má hodnota význam spíše ve vnitřních prostorách.

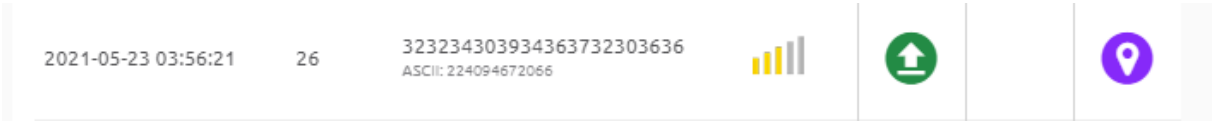
Data se mohou poslat maximálně 140x denně, modul se tedy po odeslání na 15 minut uspí a poté se vše opakuje.

Software obsahuje podrobné vypisování hodnot pro případ ladění. Sérové rozhraní lze aktivovat pouze v 15 vteřinách na začátku a nemělo by být v průběhu odpojeno. Opětovné připojení a odpojení není řešeno a může vézt k neočekávanému běhu programu.

```
void loop() {
    SigFox.begin();
    digitalWrite(WAKEUP, LOW);
    myBME280.setMode(3);
    delay(100);
    if (myCCS811.dataAvailable())
    {
      myCCS811.readAlgorithmResults();
      delay(2);
      tlak=myBME280.readFloatPressure();
      delay(2);
      teplota=myHDC1080.readTemperature();
      delay(2);
      vlhkost=myHDC1080.readHumidity();
```

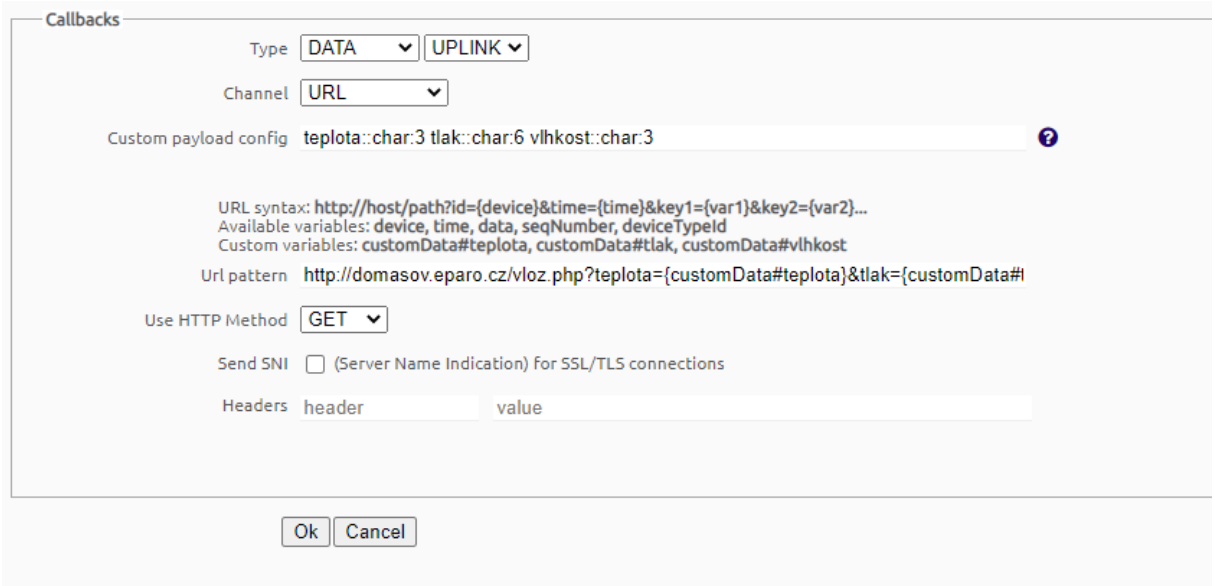
```
  delay(2);
  CO2=myCCS811.getCO2();
  delay(2);
  Evoc=myCCS811.getTVOC();
  delay(2);
   myCCS811.setEnvironmentalData(myHDC1080.readHumidity(),
  myHDC1080.readTemperature());
  delay(10);
}
myBME280.setMode(0);
digitalWrite(WAKEUP,HIGH);
SigFox.beginPacket();
  sprintf(vysledek,"%03d%06d%03d\0",(int)(teplota*10),(int)tlak,(int)
  round(vlhkost));
SigFox.print(vysledek);
stat = SigFox.endPacket();
SigFox.end();
if(Serial)
{
  Serial.print(teplota);
  Serial.print("|");
  Serial.print(tlak);
  Serial.print("|");
  Serial.print(vlhkost);
  Serial.print("|");
  Serial.print(CO2);
  Serial.print("|");
  Serial.print(Evoc);
  Serial.println(";");
  Serial.println(vysledek);
  Serial.println(stat);
}
   LowPower.sleep(900000);  //15 minut spanek
```

Pokud máme modul registrován u provozovatele SigFox sítě, měli bychom v rozhraní backendu vidět případné zprávy z našeho modulu, kdy na obrázku vidíme zprávu s informací 22.4°C 94672 Pa a 66% vlhkostí, vše složeno do jednoho „čísla“.



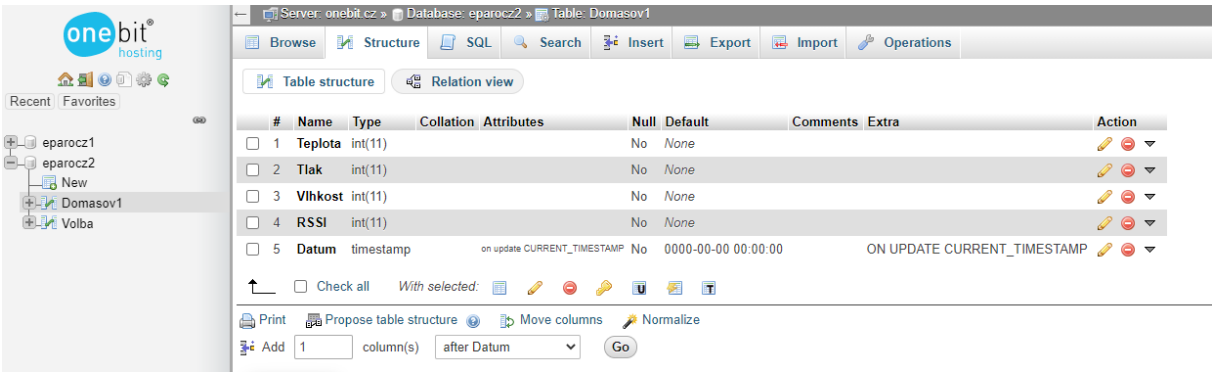
Obrázek 28: Surová data, jak jsou vidět u poskytovatele rozhraní SigFox

K zařízení si nastavíme callback, který se spustí vždy při příjmu nové zprávy. Můžeme použít průvodce pro velké provozovatele cloudových služeb, nebo si vytvořit vlastní. V tomto případě, byl požadavek na uložení informace do SQL databáze na soukromém serveru. Na server to tedy odešleme metodou GET a ve filtru callbacku si nastavíme jak má zprávu rozdělit na parametry GET.



Obrázek 29: Tvorba dotazu, pro přeposlání dat z brány SigFoxu do mé aplikace

Požadavek GET je zpracován PHP skriptem, který parametry převezme a vloží do příslušné tabulky.



Obrázek 30: Tvorba tabulky v SQL databázy

```
<html>
<body>
<?php
$teplota=$_GET["teplota"];
$tlak=$_GET["tlak"];
$vlhkost=$_GET["vlhkost"];
$RSSI=$_GET["rssi"];
$servername = "127.0.0.1";
$username = "*****";
$password = "*****";
$dbname = "******";
```

```
echo "<br>";
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn)
{
    die("Connection failed: " . mysqli_connect_error());
}
$sql = "INSERT INTO`Domasov1` (`Teplota`, `Tlak`, `Vlhkost`, `RSSI`)
VALUES ($teplota,$tlak,$vlhkost,$RSSI)";

if (mysqli_query($conn, $sql)) {
    echo "Provedeno";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Tabulka byla vytvořena s daty ve stejném formátu, jako byla přijata, vše tedy INT, plus byl přidán sloupec s časem provedení záznamu.

Pro ověření byla vytvořena jednoduchá stránka se zobrazením dat s databáze, a jelikož data nejsou v textové podobě příliš čitelná, je přidáno zobrazení dat v grafu za použití služeb google charts. Po připojení k databázi je vytvořen požadavek na vrácení všech hodnot a jejich výtisk v cyklu. Pravděpodobně bude v budoucnu nutné omezit množství tištěných záznamů.

```
</body>
</html>

<html>
<body>
<?php
$servername = "127.0.0.1";
$username = "*****";
$password = "*****";
$dbname = "******";
echo "<br>";
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn)
{
    die("Connection failed: " . mysqli_connect_error());
}
$sql = "SELECT * FROM `Domasov1` WHERE 1";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
      while ($row = $result->fetch_assoc())
 {
            echo "Cas: ".$row['Datum']."Teplota: ".$row['Teplota']."
Vlhkost: ".$row['Vlhkost']." Tlak: ".$row['Tlak']."<br />";
      }
} else {
      echo "<p>Tabulka ".$db_table." je prazdna.</p>";
}
```

```
echo "hotovo";
$sql = "SELECT Datum, Teplota, Vlhkost FROM Domasov1";
$result = $conn->query($sql);
```

Po vytištění, jsou znovu načteny hodnoty teploty a vlhkosti, které jsou vloženy do pole $entry, které je pak použito skriptem pro vygenerování grafu.

```
while ($row = $result->fetch_assoc()) {
    $entry .=
"['".$row{'Datum'}."',".$row{'Teplota'}.",".$row{'Vlhkost'}."],";
}

mysqli_close($conn);
```

Samotný prvek grafu je vložen pomocí div s odpovídajícím parametrem id.

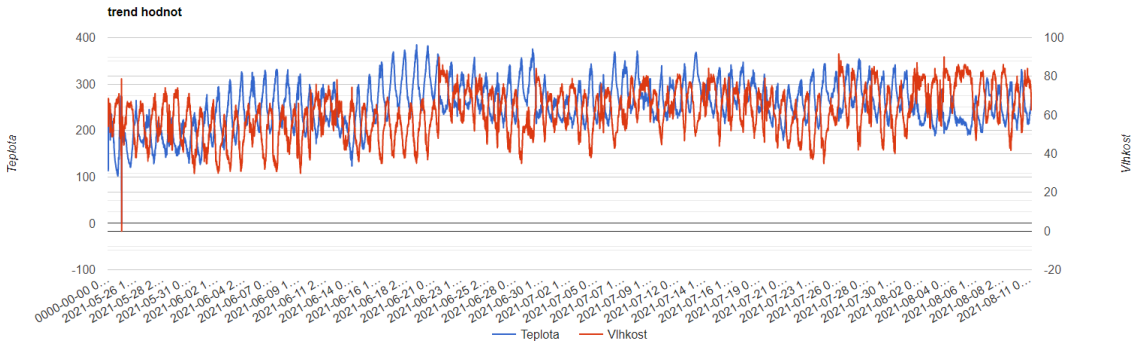
```
<div id="chart_div" style="width: 100%; height: 500px;"></div>
```

Samotný skript, podle doporučení google chart, vygeneruje graf se dvěma y osami.

```
<script type="text/javascript"
src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
    google.load("visualization", "1", {packages:["corechart"]});
    google.setOnLoadCallback(drawChart);
    function drawChart() {
        var data = google.visualization.arrayToDataTable([
        ['Datum', 'Teplota', 'Vlhkost'],
        <?php echo $entry ?>
    ]);

        var options = {
            title: 'trend hodnot',
            curveType: 'function',
            legend: { position: 'bottom' },
series: {
          0: {targetAxisIndex: 0},
          1: {targetAxisIndex: 1}
        },
        vAxes: {
          // Adds titles to each axis.
          0: {title: 'Teplota'},
          1: {title: 'Vlhkost'}
        }
        };
        var chart = new
google.visualization.LineChart(document.getElementById('chart_div'));
        chart.draw(data, options);
    }
</script>
?>
</body>
</html>
```

Výsledkem je pak stránka s výpisem hodnot a grafickou interpretací, google graf je částečně interaktivní, lze například prohlížet jednotlivé body.



Cas: 2021-08-11 04:45:25Teplota: 220 Vlhkost: 77 Tlak: 95235
Cas: 2021-08-11 05:00:32Teplota: 214 Vlhkost: 79 Tlak: 95307
Cas: 2021-08-11 05:15:41Teplota: 214 Vlhkost: 80 Tlak: 95356
Cas: 2021-08-11 05:30:49Teplota: 217 Vlhkost: 78 Tlak: 95299
Cas: 2021-08-11 05:45:57Teplota: 218 Vlhkost: 78 Tlak: 95287
Cas: 2021-08-11 06:01:05Teplota: 217 Vlhkost: 78 Tlak: 95307
Cas: 2021-08-11 06:16:13Teplota: 219 Vlhkost: 77 Tlak: 95270
Cas: 2021-08-11 06:31:21Teplota: 215 Vlhkost: 79 Tlak: 95334
Cas: 2021-08-11 06:46:30Teplota: 223 Vlhkost: 77 Tlak: 95203
Cas: 2021-08-11 07:01:37Teplota: 225 Vlhkost: 77 Tlak: 95170
Cas: 2021-08-11 07:16:46Teplota: 222 Vlhkost: 78 Tlak: 95247
Cas: 2021-08-11 07:31:54Teplota: 233 Vlhkost: 75 Tlak: 95086
Cas: 2021-08-11 07:47:04Teplota: 239 Vlhkost: 74 Tlak: 94997
Cas: 2021-08-11 08:02:09Teplota: 237 Vlhkost: 76 Tlak: 95031
Cas: 2021-08-11 08:17:18Teplota: 239 Vlhkost: 75 Tlak: 95014
Cas: 2021-08-11 08:32:26Teplota: 239 Vlhkost: 77 Tlak: 95009
Cas: 2021-08-11 08:47:35Teplota: 236 Vlhkost: 78 Tlak: 95046
Cas: 2021-08-11 09:02:42Teplota: 243 Vlhkost: 77 Tlak: 94944
Cas: 2021-08-11 09:17:50Teplota: 244 Vlhkost: 76 Tlak: 94922
Cas: 2021-08-11 09:32:59Teplota: 246 Vlhkost: 76 Tlak: 94902
Cas: 2021-08-11 09:48:06Teplota: 248 Vlhkost: 75 Tlak: 94868
Cas: 2021-08-11 10:03:14Teplota: 250 Vlhkost: 75 Tlak: 94835
Cas: 2021-08-11 10:18:22Teplota: 252 Vlhkost: 73 Tlak: 94806
Cas: 2021-08-11 10:33:30Teplota: 255 Vlhkost: 73 Tlak: 94776
Cas: 2021-08-11 10:48:38Teplota: 259 Vlhkost: 71 Tlak: 94706
Cas: 2021-08-11 11:03:47Teplota: 265 Vlhkost: 71 Tlak: 94597
Cas: 2021-08-11 11:18:55Teplota: 267 Vlhkost: 63 Tlak: 94547
Cas: 2021-08-11 11:34:03Teplota: 269 Vlhkost: 64 Tlak: 94540
hotovo

Obrázek 31: Ústřižek dat zobrazených na rozhraní pro snímač počasí v honitbě

Jelikož lze jednomu zařízení nastavit více callbacků, je možné tato data zároveň předávat i na jiná úložiště (opět například thinkspeak), stránky, případně odeslat emailem.

Závěr

Úkolem práce bylo navrhnout systém pro monitoring odlehlých lokalit s omezenými možnostmi napájení a připojení. V průběhu práce jsem provedl rešerši možných řešení, ze kterých vyšla v podmínkách ČR nejvýhodnější možnost využití sítě SigFox, která má celorepublikové (víceméně celosvětové) pokrytí a v případě nutnosti dokrytí je možné po domluvě zřizovat další stanice pro dokrytí oblastí se špatným signálem. Jako ukázka využití byli sestrojeny tři řešení, monitoring krmelce, monitoring včelína a monitoring počasí v honitbě kvůli předpovědi pravděpodobnosti výskytu mlh. Posední zmiňované řešení je půl roku v neustálém provozu a stále bez výpadků měří. Data je možné vidět v prezentaci řešení, nebo přímo na webu v živém náhledu. Ukázkami řešení a vytvořením frontendu pro uživatele, z ukázkami webových aplikací a možností využití cloudových služeb jsem splnil všechny podmínky zadání. Drobnou nevýhodou řešení je potřeba licence pro koncová zařízení v případě komercializace řešení a nutnost udržovat bastlířskou licenci pro provoz prototypových řešení jako je moje. Dalším drobným omezením je datová propustnost řešení, ale vzhledem k velmi nízké energetické náročnosti a malým overhead protokolu sigfox je to pochopitelné a bylo by nutné i při případné tvorbě vlastního řešení bez závislosti na stávajících sítí.

Literatura

[1] Microchip, „datasheet MRF24WG0MA/MB,“ [Online]. Available: https://www.farnell.com/datasheets/1694199.pdf. [Přístup získán 14 2 2021].

[2] Vodafone, „mapa pokrytí vodafone,“ 2021. [Online]. Available: www.vodafone.cz/pokryti. [Přístup získán 1 8 2021].

[3] Quectel, „datasheet Quectel EC25,“ [Online]. Available: https://sixfab.com/wp-content/uploads/2021/02/Quectel_EC25_Series_Mini_PCIe_LTE_Standard_Specification_V2.0.pdf. [Přístup získán 14 2 2021].

[4] Microchip, „datasheet RN2903,“ [Online]. Available: https://www.farnell.com/datasheets/2595631.pdf. [Přístup získán 14 2 2021].

[5] wikipedia, „SigFox,“ [Online]. Available: https://cs.wikipedia.org/wiki/Sigfox. [Přístup získán 14 2 2021].

[6] LPRS, „datasheet eRIC sigfox,“ [Online]. Available: https://www.farnell.com/datasheets/2549154.pdf. [Přístup získán 14 2 2021].

[7] PyCom, „PyCom,“ [Online]. Available: https://docs.pycom.io/. [Přístup získán 15 1 2021].

[8] Arduino, „Arduino store,“ [Online]. Available: https://store.arduino.cc/arduino-mkr-fox-1200-1408. [Přístup získán 2 2 2021].

[9] Pajenicko. [Online]. Available: https://pajenicko.cz/gyroskop-akcelerometr-gy-521-s-mpu6050-i2c. [Přístup získán 1 3 2021].

Příloha

Elektronická příloha obsahuje kódy aplikací s výjimkou webových aplikací, jejichž kód je uveden v práci celý.