Středoškolská technika 2022

Setkání a prezentace prací středoškolských studentů na ČVUT

SCOUT

Libor Miller, Ondřej Gaman

Vyšší odborná škola a Střední průmyslová škola elektrotechnická Plzeň
Koterovská 85, Plzeň 326 00

Anotace

Ve své ročníkové práci jsme se zabývali vývojem, výrobou a programováním našeho zařízení. Chceme zaujmout novým pohledem na trackovací zařízení. Na rozdíl od většiny konkurence je zařízení schopno fungovat samostatně (nepotřebuje mobilní telefon), stačí mu jen pokrytí LTE, nevyužívá ale klasické sítě jako 4G, nebo 5G, ale síť NB-Iot, které dovoluje využívat moduly s nízkou spotřebou. A vzhledem k tomu, že na nízkou spotřebu bylo dbáno i u zbytku komponent, je výdrž lepší než u mobilních telefonů.

Klíčová slova

IoT, Tracking, GPS, NB-IoT

Annotation

In our year work we have been involved in the development, manufacturing and programming of our device. We want to take a new looks at tracking devices. Unlike most of the competitors, the device is able to work independently (it does not need a mobile phone), it only needs LTE coverage, but it does not use common networks like 4G or 5G, but the NB-Iot network, which allows the usage of low power modules. And since low power consumption has been taken care of in the rest of the components, the battery life is better than mobile phones.

Keywords

IoT, Tracking, GPS, NB-IoT

Obsah

1.1 Úvod

Primárním cílem našeho projektu je zajistit vyšší úroveň bezpečí dětí při různých volnočasových aktivitách jako jsou přespolní běhy nebo různé hry kde je možnost, že dítě sejde ze stezky a jednoduše zabloudí. Což se může stát nejen noční můrou pro dítě, ale také pro rodiče, vedoucí, či organizátora akce. Z vlastní zkušenosti víme, že se takovéto situace mohou stát a v realitě jsou velmi časté. Náš tracker by přesně takovým situacím měl předcházet. Vzhledem k velikosti by mělo být možné zařízení připevnit na pásek, vložit do batohu, nebo dát do kapsy. Díky velikosti celého zařízení a také tomu, že zařízení je schopné pracovat samostatně (bez mobilního telefonu), by to pro dítě neměla být v podstatě žádná zátěž. Samozřejmě by o tom mělo být dítě či jiný uživatel obeznámen, že má náš tracker v blízkosti.

Díky bezdrátové technologii Narrow band IoT, která má vysoké procento pokrytí po celé ČR (trošku konkrétněji ve všech místech, které pokrývá klasické LTE), můžeme přenášet aktuální pozici zařízení a data sledovat pomocí webové aplikace. Tracker bude také uživatele ve webové aplikaci upozorňovat na stav baterie zařízení.

Samozřejmě, že by to nebyl tracker bez žádného lokalizačního systému, vzhledem k vybranému modulu GNSS jsme schopni využívat systémy: GPS, GLONASS, GALILEO, BEIDOU, což znamená, že lokalizace bude velice přesná. S výdrží baterie cílíme na 1-2 týdny běžného používání, čehož budeme schopni dosáhnout díky vhodně vybraným komponentám, u kterých jsme dbali hlavně na poměr spotřeby a výkonu. Bohužel vzhledem k tomu, že krize nedostatku polovodičů stále nepominula, portfolio produktů na výběr bylo tedy menší, ale i tak si myslíme že se nám podařilo vytvořit povedený kompromis. Abychom spotřebu nesnižovali jenom takto „pasivně“, námi vybrané komponenty obsahují různé spánkové módy, kterých budeme využívat.

Funkčnost těchto low power možností nám bude zařizovat firmware zařízení. Díky periferiím na desce jsme schopni celé zařízení probouzet pouze při pohybu, takže pokud bude stacionární, využije co nejméně energie z baterie.

Podle popisu vyplývá, že náš tracker se dá využít daleko více způsoby a náš původní záměr je jen jedním z nich. Ke sportovním aktivitám ho samozřejmě mohou využít i dospělí, od sledování trasy při víkendové cyklistice, až po běhy ultramaratonu, při aktivitách jako tyto se bude hodit i detekce pádu umožněná integrovaným gyroskopem a akcelerometrem BMI270. S malou úpravou je možné zařízení použít i pro zabezpečení majetku či jiných objektů, se kterými se přiliž často nemanipuluje a dokážou tak využít dlouhého běhu zařízení na baterii a velkého pokrytí bezdrátové sítě v těžko přístupných oblastech. Počet různých použití je široký a je jen na uživateli, jak zařízení využije.

Mohla by zde vzniknout obava o nechtěné sledování, jako to bylo v případě jiných zařízení sledujících polohu uživatele. Momentálně není v našich možnostech, jak tomuto zabránit, případně jak tuto možnost zcela eliminovat, ale tím že k technologii budeme mít přístup pouze my či námi zvolené osoby a webové stránky budou řádně zabezpečeny, tak by možnost jakéhokoliv zneužití měla být nulová.

Tracker plánujeme využít tento rok na letním táboře a dalších akcích, kde budeme testovat převážně funkčnost, aplikaci a výdrž baterie.

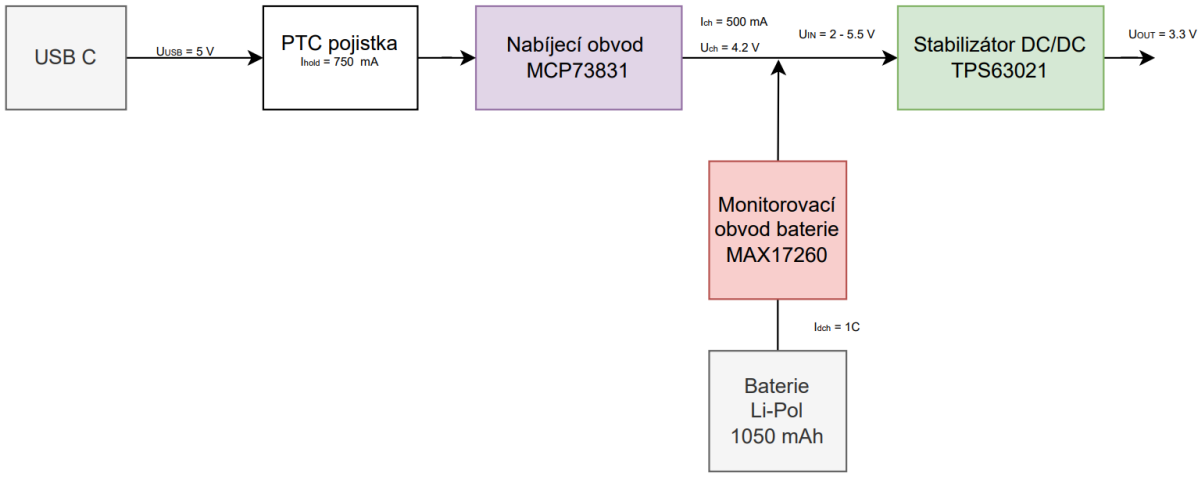
Obrázek 1-3D render zařízení

2.0 Hardware

U všech klíčových komponent jsme dbali hlavně na jejich spotřebu a fyzickou velikost. Ve výběru také samozřejmě hrála roli cena i momentální dostupnost komponent, což vzhledem k stále trvajícímu nedostatku polovodičů bylo těžší, než se na první pohled zdá, ale věříme, že i přes to se nám podařilo vytvořit povedený kompromis mezi všemi aspekty.

2.1 Nabíjení

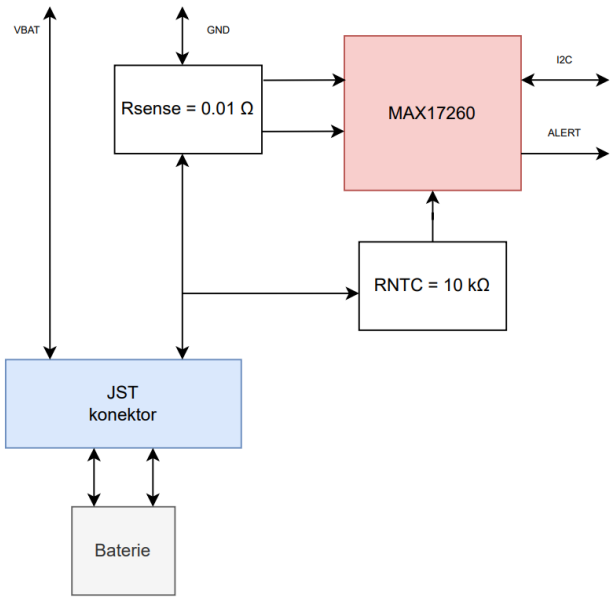
Zařízení je napájeno skrze USB-C a Li-Pol baterií připojenou JST konektorem. Používáme nabíjecí obvod MCP73831, který má maximální nabíjecí výkon 500 mA. Všechny naše periferie podporují napájecí napětí 3,3 V, proto používáme spínaný stabilizátor TPS63021.



Obrázek 2-Blokové schéma napájecí části

2.1.1 Monitorovací obvod baterie MAX17260

Obvod měří proud protékající skrze snímací odpor (bočník). Ten je zapojen sériově s baterií u jejího záporného pólu. Podle výrobce je obvod schopný (při správném nastavení registrů)



Ob

měřit procentuelní nabití baterie s přesností ±1 %. Integrovaný obvod podporuje také indikaci přehřátí (potřebuje externí komponentu v podobě NTC termistoru).
Obvod komunikuje s mikrokontrolerem pomocí sběrnice I2C. Pro měření je nutné nastavení registrů s hodnotou kapacity baterie, napětí při vybití, proud, při kterém se ukončí napájení. Dále využíváme možnost programovatelného pinu ALERT, který může mikrokontroler „upozornit“ v případě že nastane námi zvolená situace (např. překročení nastaveného napětí, teploty, upozornění na začátek napájení…).

MAX17260	MCU	Funkce
SDA	PA-12	Napojení na sběrnici I2C
SCK	PA-13	
ALERT	PA-18	Programovatelný interrupt

2.1.2 Spínaný stabilizátor TPS63021

Vzhledem k tomu že hodnota napětí z baterie může fluktuovat od 3 V do 4,2 V, byl použit buck-boost konvertor, díky němuž docílíme stálého napětí 3,3 V i v případě nízké úrovně nabití baterie. Důležitým faktorem pro výběr tohoto stabilizátoru je přítomnost Power Save módu, díky němuž je zajištěna dobrá efektivita i při nízké zátěži, což je stav, který bude převažovat většinu času, ale při zapínání modulů a mikrokontroleru může být zátěž i kolem 1A.
Při výběru cívky bylo dbáno na maximální saturační proud, stejnosměrný proud, maximální napětí i sériový odpor. Abychom zjistili maximální proud, který bude cívkou protékat použili jsem vzorce dané od výrobce stabilizátoru (1).

Duty Cycle Boost

$$D = \frac{U_{OUT} - U_{IN}}{U_{OUT}}$$

$$D = \frac{3{,}3 - 3}{3{,}3} \cong 0{,}09$$

Maximální proud protékající cívkou
Parametry od výrobce stabilizátoru:

- f = spínací frekvence stabilizátoru (typicky 2.5 MHz)
- L = typicky 1 – 1,5 µF
- η = odhadovaná efektivita stabilizátoru (0.9)

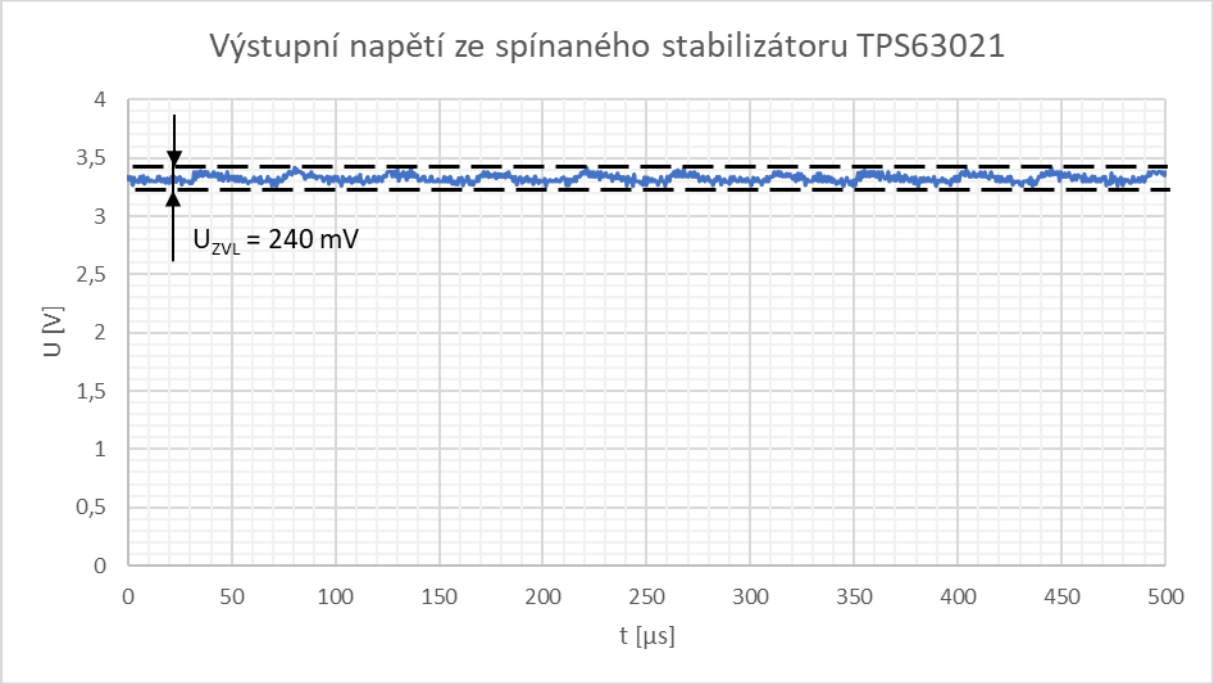
$$I_{PEAK} = \frac{I_{OUT}}{\eta * (1 - \mathrm{D})} + \frac{U_{IN} * D}{2 * f * L}\ [A]$$

$$I_{PEAK} = \frac{1}{0{,}9 * (1 - 0{,}09)} + \frac{3 * 0{,}09}{2 * 2{,}5 * 10^{6} * 1 * 10^{-6}} \cong 1{,}28\ A$$

Vzhledem k tomu, že maximální velikost proudu I_{OUT} byla stanovena odhadem (sečtení proudových odběrů všech komponent při jejich maximu uvedeném v datových listech výrobců) je reálná cívka, co se maximálního proudu týče, naddimenzovaná oproti výpočtu přibližně 2x.
Parametry použité cívky

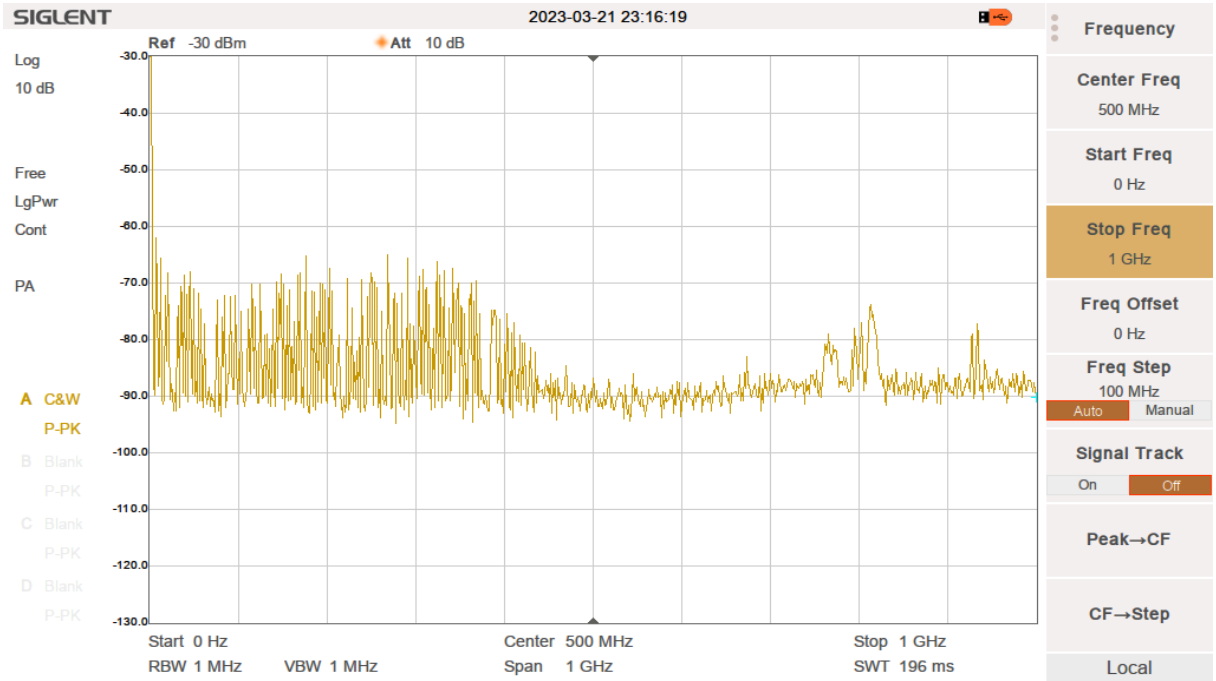
- L = 1 µH
- R = 60 mΩ
- I_{MAX} = 3.1 A
- U_{MAX} = 20 V

- Pouzdro: 0806 (in)



Obrázek 4-Graf zvlnění výstupního napětí ze stabilizátoru TPS63021

Vzhledem k použití spínaného stabilizátoru je výstupní napětí lehce zvlněné což je umocněné tím že stabilizátor v době měření fungoval v Power Save módu. Naměřený činitel zvlnění $\varphi_{ZVL} = \frac{0,24}{3,3} * 100 = 7,27\ \%$. V datovém listu výrobce (1) je uvedené maximální zvlnění při fungování v Power Safe módu ±5 %, což žádná z naměřených hodnot nepřesáhla.



Na výstupu stabilizátoru jsme také změřili frekvenční spektrum. Nejsilnější rušivý signál vzniká přímo ve stabilizátoru, kvůli jeho spínací frekvenci 2,4 MHz.

2.1.3 USB C a MCP73831

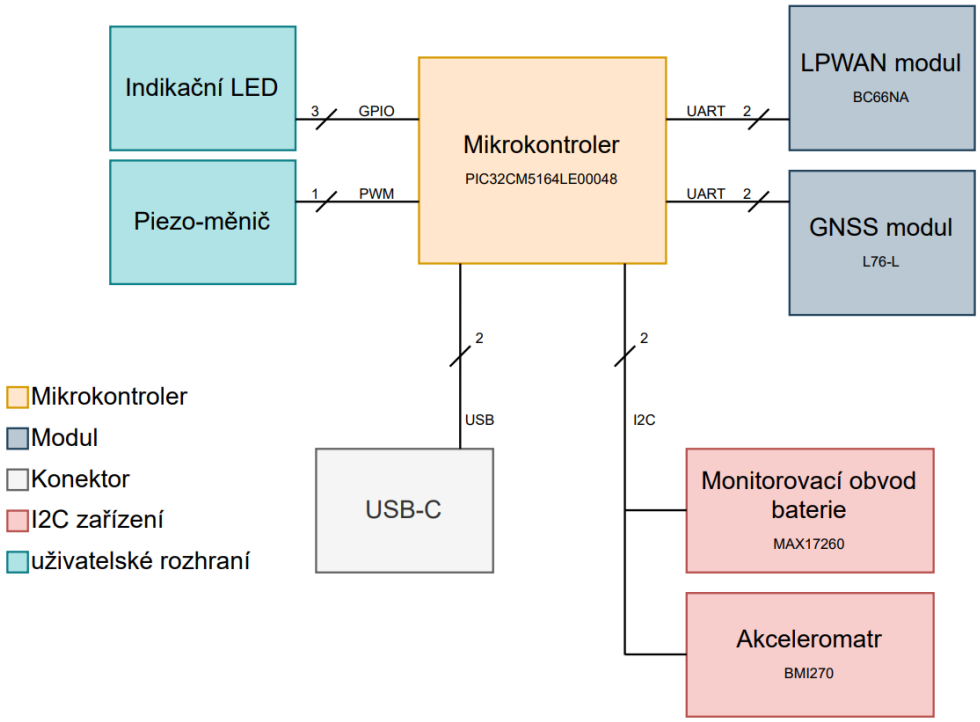
USB 2.1 je používáno pro napájení, ale i pro přenos dat rychlostí 12 Mbit/s. USB konektor je chráněn proti elektrostatickému výboji pomocí obvodu PESD4USB5BTBR-Q (C_D = 0.22 pF) u tohoto vysokorychlostního rozhraní nebylo možné použít klasické zenerovy diody kvůli velikosti jejich parazitních kapacit.
Aby fungovala detekce zařízení i u USB-C na USB-C nabíječek, a vzhledem k tomu že nevyužíváme Power Delivery, bylo nutné přidat na linky CC1 a CC2 pull-down rezistory R_{PD}= 5,1 kΩ, tato hodnota je definována standardem pro USB-C, maximální napájecí výkon v této konfiguraci je 15 W (I_{USB} = 3 A, U_{USB} = 5 V). Za USB je vratná PTC pojistka, která se rozepne v případě, že by protékající proud byl vyšší než I_{HOLD} = 750 mA.
Rychlost nabíjení je také omezena nabíjecím obvodem baterie, který je schopen dodávat proud I_{CH} = 500 mA při napětí V_{CH} = 4,2 V. Jako vizuální detekci nabíjení používáme zelenou diodu. Předřadný odpor před diodu $R_P = \frac{V_{IN} - U_D}{I_D} = \frac{5-1,9}{0,005} \cong 1\ k\Omega$; odpor 1 kΩ sice omezí proud na 3,1 mA, ale vzhledem k využití je celková svítivost diody dostatečná.

2.2 Hlavní komponenty

Celé zařízení je řízeno mikrokontrolerem PIC32CM5164LE00048. Ke komunikaci používá LPWAN síť NB-IoT kvůli jejímu pokrytí a spotřebě samotného modulu. Pro získávání samotných souřadnic může naše zařízení využívat sítě GPS, GLONASS, GALILEO, BEIDOU. Pro zajištění co nejmenší spotřeby je zde také akcelerometr/gyroskop. Díky němuž se nemusí zapínat energeticky náročný GNSS modul, ani LPWAN modul, když se zařízení nepohybuje.

2.2.1 Mikrokontroler PIC32CM5164LE00048

Mikrokontroler založený na Cortexu M23. Obsahuje flash o velikosti 512 kB a SRAM 64 kB, jeho maximální rychlost je 48 MHz, každopádně vzhledem k tomu, že cílíme na co nejlepší životnost baterie, tato maximální rychlost používána většinu času nebude. V návaznosti na vyšší efektivitu jsme také zapojili mikrokontroler tak, aby mohlo fungovat v tzv BUCK módu, který zajišťuje vyšší efektivitu tím, že pro napájení svých vnitřních částí používá vestavěný spínaný stabilizátor namísto vestavěného LDO, je k tomu ale nutná přídavná cívka s indukcí L = 10 μH. Zařízení budeme programovat pomocí externího programátoru MPLAB SNAP, proto je na desce kompatibilní 8 pinový konektor. Další z výhod je množství dostupných sběrnic, my využíváme UART, I2C a USB.



.

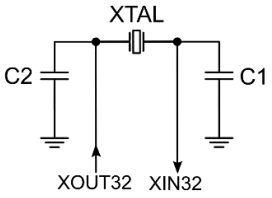
Obrázek 5-Blokové schéma komunikací s MCU

Externí krystal

Důvod proč byl použit externí krystal je, že ten interní není dostatečně přesný, má odchylku -23% až +25 % (2), což je pro sběrnici USB nedostatečné. Proto jsme vybrali externí krystal s těmito parametry:

- f = 32,768 kHz
- Přesnost 20 ppm (0,002 %)
- C_L = 12,5 pF

Pro funkčnost krystalu jsou nutné kondenzátory, které jsme podle vzorce daného výrobcem mikrokontroleru dopočítaly (2).



Obrázek 6 - schéma pro zapojení externího krystalu viz. (2)

Hodnoty dané z datasheetu výrobců:

- C_{XTAL_EFF} = 5,5 pF
- C_{PCB} = 1,5 pF na 12,5 mm
- C_L = 12,5 pF

$$C1 = C2 = \left((2 * C_L) - CXTAL_{EFF} - (2 * C_{PCB})\right)$$

$$C1 = C2 = \left((2 * 12{,}5) - 5{,}5 - (2 * 0{,}75)\right)$$

$$C1 = C2 = 18\ pF$$

USB

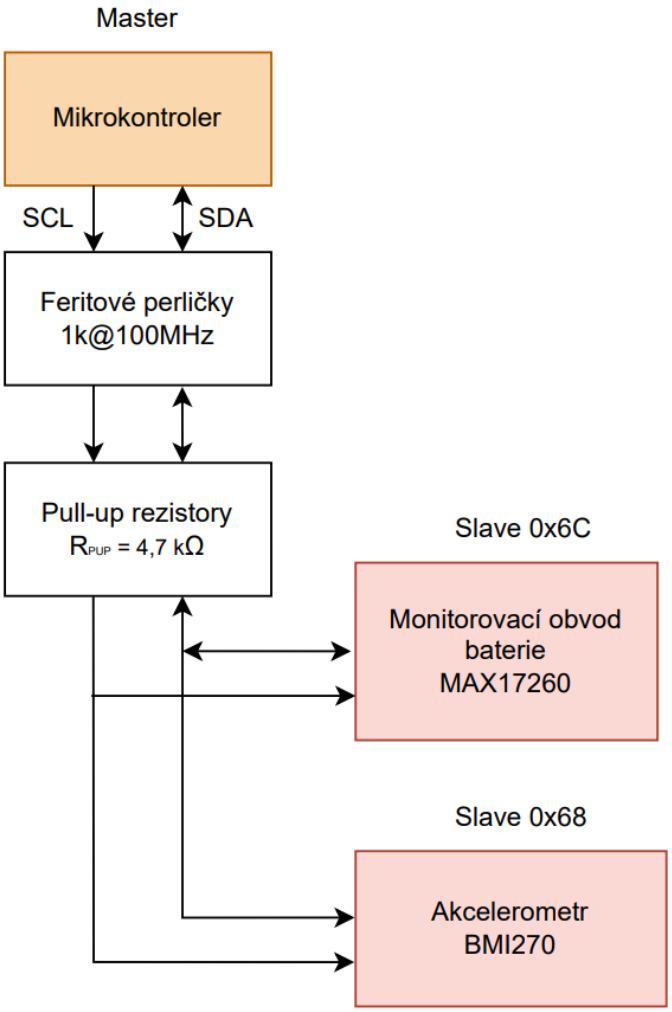
Mikrokontroler přímo podporuje USB 2.1 s max. rychlostí 12 Mbit/s. Pro provoz USB je zapotřebí přesnější externí oscilátor, využili jsme proto řídící jednotku oscilátoru XOSC32K (obsažena v mikrokontroleru) v kombinaci s externím krystalem na frekvenci 32.768 kHz. A pomocí vnitřních násobičů jsme byly schopni dosáhnout frekvence 12 MHz.
Vzhledem k tomu že je USB diferenční sběrnice, je nutné dbát na rozdíl délky jednotlivých cest, v našem případě je rozdíl 0,2 %, což je pro USB 2.1 dostatečně malý rozdíl.
USB je využíváno v módu CDC device. To znamená že je zařízení schopné navázat sériovou komunikaci s PC. USB využíváme pro výpis textu do konzole (v rámci debugování zařízení).

Programování mikrokontroleru
K programování používáme programátor Microchip MPLAB SNAP, a pro připojení používáme 8 pinový konektor Microchip SIL (mikrokontroler je kompatiblní i s Cortex Debug Connectorem, nebo s IDC JTAG, ale toto byla cenově nejvýhodnější cesta, vzhledem k cenám samotných programátorů). Vzhledem k tomu že RESET (MCLR) je invertovaný bylo nutné přidání pull-up rezistoru R_{PUP} = 100 kΩ.

Konektor na programátoru	MCU (číslo pinu)
/MCLR	40
VCC	-
GND	-
-	-
SWCLK	45
-	-
-	-
SWDIO	46

2.2.2 sběrnice I2C

Na sběrnici budou napojena 2 slave zařízení a 1 master zařízení v podobě mikrokontroleru. Používáme fast mode (rychlost sběrnice je 400 kHz). Co se samotného designu sběrnice týče bylo nutné abychom nepřekročili maximální kapacitní zátěž sběrnice 200 pF (3) a zároveň je nutné použít pull-up rezistory.



Obrázek 7 - Blokové schéma pro zapojení sběrnice I2C

Výpočet velikosti parazitní kapacity C_L (4)

- x = délka cesty (cm) = 5,9 cm
- w = šířka cesty (cm) = 0,0254 cm
- h = výška dielektrika pod cestou (cm) = 0,1 cm
- t = výška cesty (cm) = 0,035 cm
- ε_r = permitivita dielektrika = 4,5
- C_{BMI270} = 5 pF
- $C_{MAX17260}$ = 6 pF

$$C_{trace} = \frac{0{,}264 * x * (\varepsilon_r + 1{,}41)}{\ln\left(\frac{5{,}98 * h}{0{,}8 * w + t}\right)} \; [pF]$$

$$C_{trace} = \frac{0{,}264 * 5{,}9 * (4{,}5 + 1{,}41)}{\ln\left(\frac{5{,}98 * 0{,}1}{0{,}8 * 0{,}0254 + 0{,}035}\right)} \cong 3{,}9 \; pF$$

$$C_{BUS} = 2 * C_{trace} = 2 * 3{,}9 = 7{,}8 \; pF$$

$C_L = C_{BMI270} + C_{MAX17260} + C_{BUS} = 5+6+7{,}8 = 18{,}8$ pf

Výpočet pull-up rezistorů pro I2C (3)

- T_{RMAX} = 300 ns
- U_{CC} = 3,3 V
- U_{OLMAX} = 0,4 V
- I_{OL} = 3 mA

$R_{PUPMAX} = \frac{T_{rMAX}}{(0{,}847 * C_L)} = \frac{300 * 10^{-9}}{(0{,}847 * 18{,}8 * 10^{-12})} \cong 18{,}839 \; k\Omega$

$R_{PUPMIN} = \frac{U_{CC} - U_{OLMAX}}{I_{OL}} = \frac{3{,}3 - 0{,}4}{3 * 10^{-3}} \cong 967 \; \Omega$

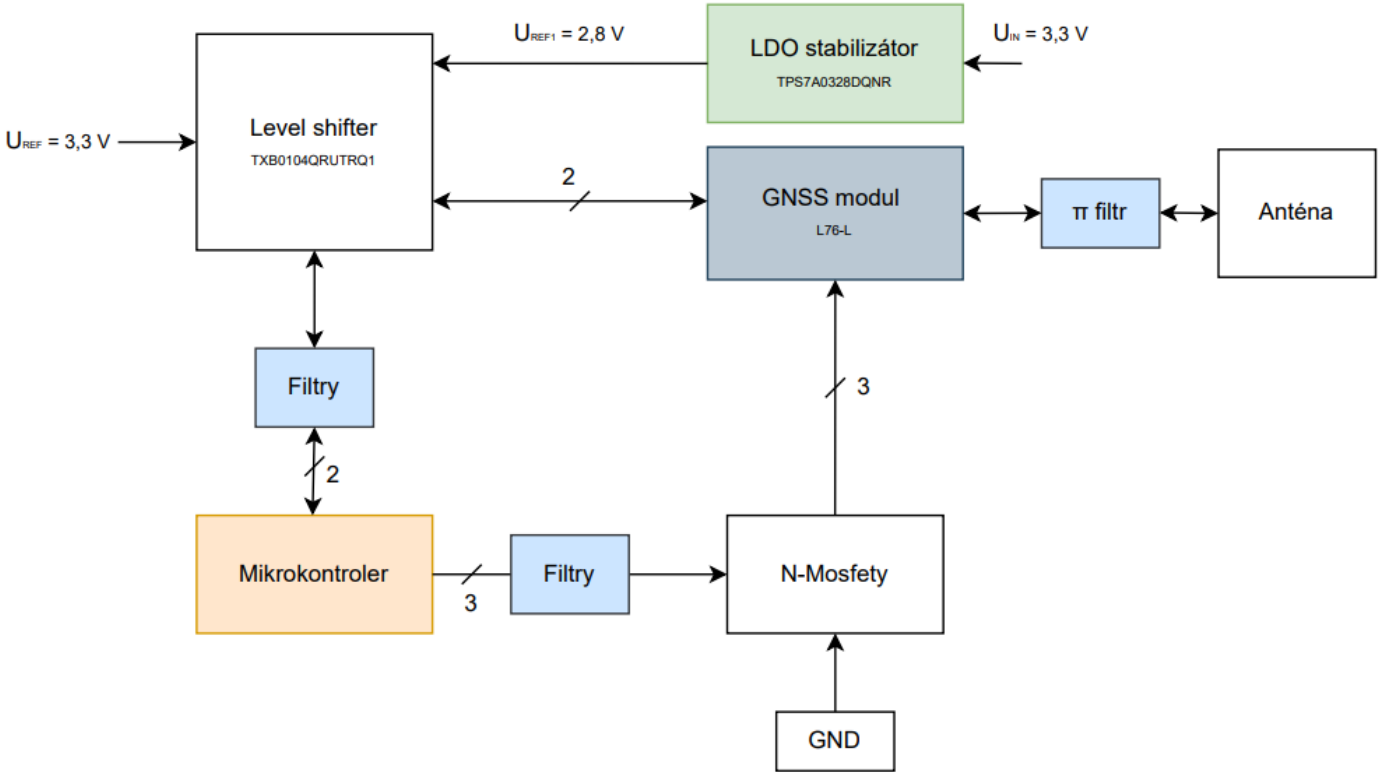
Z výpočtů nám vyplívá že pull-up rezistor musí mít hodnotu mezi 967 Ω a 18,839 kΩ. Zvolili jsme jednu z možných standartních velikostí a to 4,7 kΩ.

Feritové perličky

Pro ochranu proti šumu jsme na sběrnici umístili i feritové perličky s hodnou impedance 1000 Ω na frekvenci 100 MHz (5).

2.2.3 GNSS modul Quectel L76L

Modul podporuje několik globálních navigačních systémů: GPS, GLONASS, GALILEO, BEIDOU. V samotném modulu jsou obsaženy filtry, veškerá výpočetní technika potřebná ke zpracování signálu i ARM procesor. Používáme výchozí firmware od výrobce modulu. Modul používá frekvence mezi 1559 a 1609 MHz.
GPIO piny nejsou přímo napojené na mikrokontroler, ale pomocí mosfetů jsou nastavovány mezi GND a NC (nepřipojeno).
Pro komunikaci mezi modulem a naším mikrokontrolerem používáme UART a GPIO. Vzhledem k tomu, že na I/O pinech má GNSS modul doporučené napětí mezi -0,3 a 3,1 V a mikrokontroler má na I/O pinech napětí 3,3 V (bylo by možné použít i nižší, aby napěťové úrovně byly kompatibilní, ale vzhledem ke zbytku desky by se to nevyplatilo). Museli jsme použít převodník úrovní TXB0104QRUTRQ1, pro získání referenčního napětí používáme malé LDO TPS7A0328DQNR. Jako druhé referenční napětí slouží napětí z hlavního stabilizátoru TPS63021.



Obrázek 8 - Blokové schéma zapojení L76L

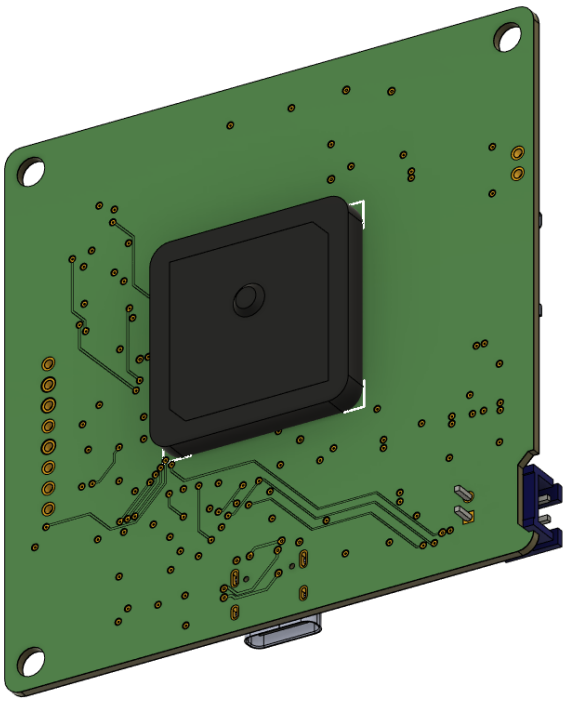
Propojení s mikrokontrolerem a funkce

Chceme využít maximum užitečných funkcí modulu, jednou z priorit je optimalizace spotřeby energie, což je možné díky několika módům. Díky přítomnosti akcelerometru BMI270, může být GNSS modul v Contiuous módu (6) jen když se zařízení skutečně hýbe, jinak bude využívat Backup mód nebo Standby mód, v závislosti na době od posledního pohybu.

L76L	MCU	Funkce
RDX	PA-08	UART pro komunikaci
TXD	PA-09	
FORCE-ON (WAKEUP)	PA-10	Probouzí modul z Backup módu
STANDBY	PA-07	Přepíná, zda je zapnutý nebo vypnutý STANDBY mód
RESET	PA-06	Resetuje modul

Anténa

Pro GNSS modul jsme zvolili patchovou anténu 204286-0001, která se nalepuje ze spodní strany desky, z druhé strany je signál pomocí prokovu naveden skrze π filtr (pro případné dorovnání impedance na 50 Ω) do modulu. Pro nejlepší efektivitu antény je nutné aby ze strany kde je anténa nebyla žádná další součástka a ideálně aby tam nebyly ani žádné prokovy, nebo cesty, ale vzhledem ke kapacitám výrobce a případné ceně jsme udělali PCB 4 vrstvé. Takže cesty i prokovy z dolní strany jsou. Při designu jsme ale dbali na to, aby tam nebylo moc vysokofrekvenčních cest, jediná taková cesta je I2C sběrnice. Při reálném testu GPS jsme dosahovali přesností ±5 m i při komunikaci I2C, takže se to prozatím nezdá jako velký problém.



Obrázek 9 - Umístění patchové antény z dolní strany PCB

Základní informace o anténě z datových listů výrobce (7) (8)

- Velikost: 25x25 mm
- a_{uMAX} = 5,4 dB
- celková efektivita >70 %

Při použití na naší desce se parametry budou samozřejmě lišit, ale nemáme nástroje pro jejich přesné změření.

RC filtry na komunikačních linkách

Vzhledem k tomu že jsme chtěli zvýšit odolnost vůči šumu (který může ve větší míře vznikat např. ze spínaných stabilizátorů, NB-Iot modulu, mikrokontroleru… signály řádově jednotky MHz až jednotky GHz) jsou důležité komunikační kanály opatřeny RC filtry.
Všechny filtry na komunikačních cestách mezi L76L a mikrokontrolerem, většinou i na zbytku desky, jsou RC, s hodnotou R = 1 kΩ, kvůli omezení proudu na maximální hodnotu 3,3 mA.
Kapacita kondenzátorů byla vypočítána s ohledem na přechodové jevy.
Filtr pro UART s baud rate 115 200 Bd.

$$f_m = \frac{1}{2\pi RC}$$

$$T = \frac{1}{115\,200} = 8{,}6\ \mu s$$

Aby UART zpracovával logické hodnoty správně omezili jsme náběžnou hranu na $\frac{1}{5}\ T$

$$T_P = \frac{T}{5} = 1{,}72\ \mu s$$

$$T_p = 5\tau$$

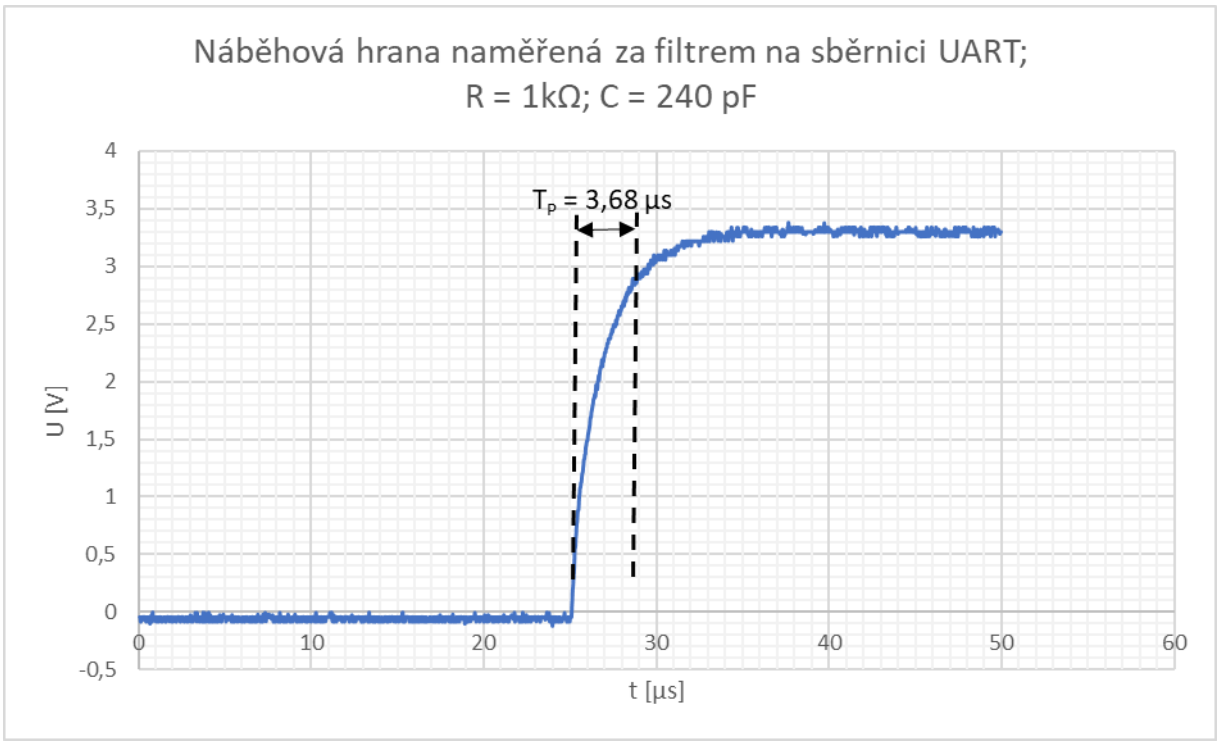
$$T_p = 5RC$$

$$1{,}72 * 10^{-6} = 5 * 1000 * C$$

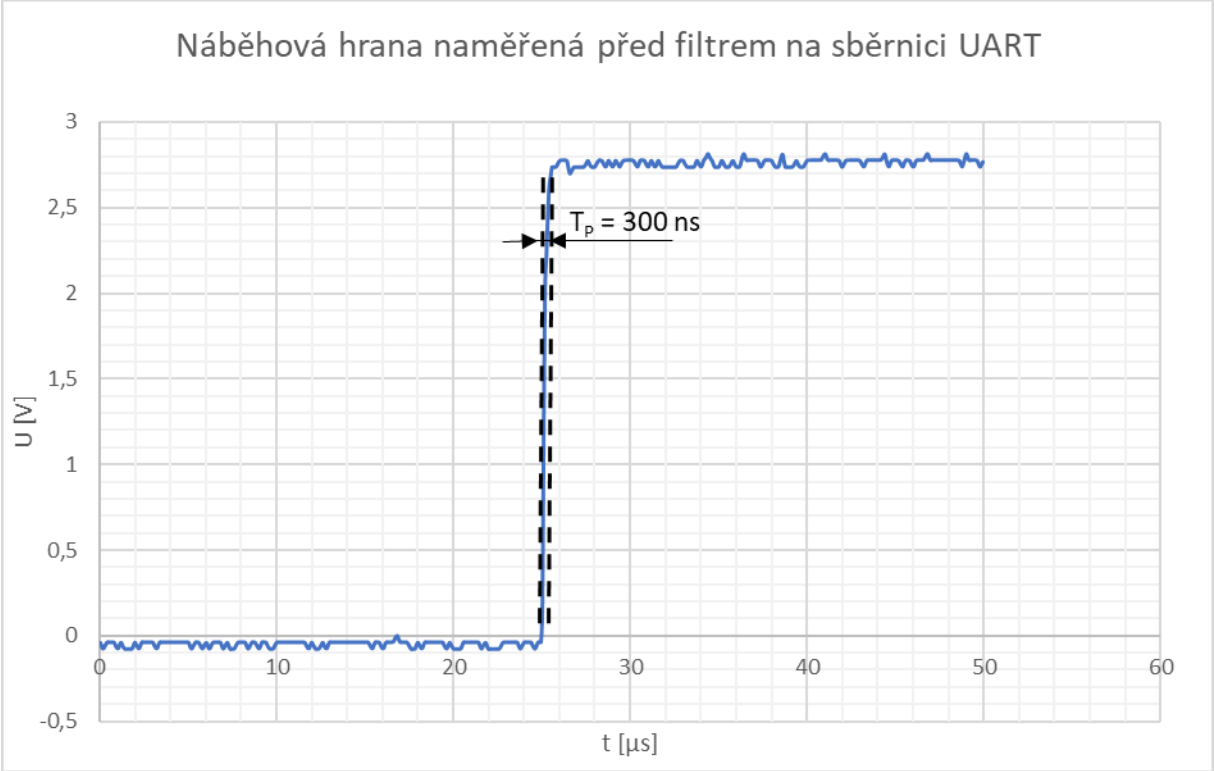
$$C = 344\ \text{pF}$$

Protože jsme ale měli zrovna k dispozici hodnotu 240 pF použili jsme tu. Tím se sice zkrátí náběžná hrana a mezní frekvence se zvýší, ale vůči frekvencím, které chceme z cest odfiltrovat je tento rozdíl zanedbatelný.
Zbytek RC filtrů byl počítán obdobným způsobem.

Náběhová hrana naměřená na reálném filtru se liší od teoretické hodnoty o 1,96 µs (viz graf 10). UART bez problému funguje i když je reálná náběhová hrana delší.



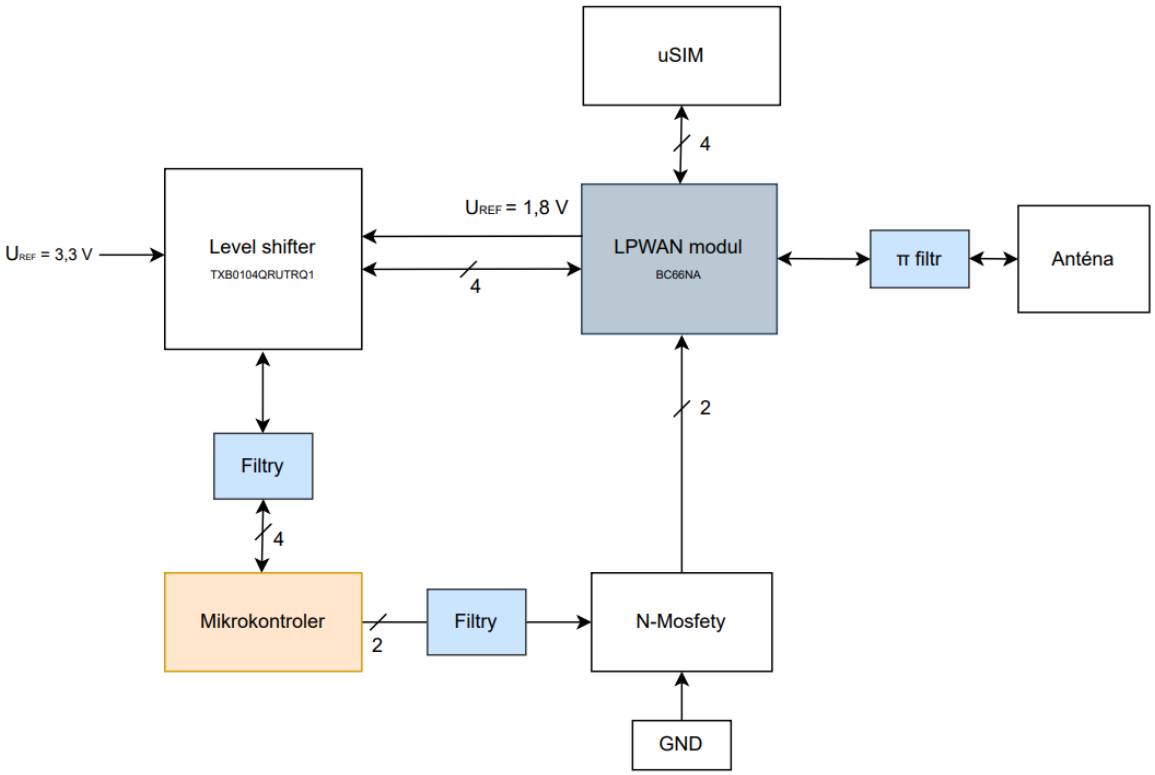
Obrázek 10-graf náběhové hrany za filtrem



Obrázek 11- graf náběhové hrany před filtrem

2.2.4 LPWAN modul Quectel BC66NA

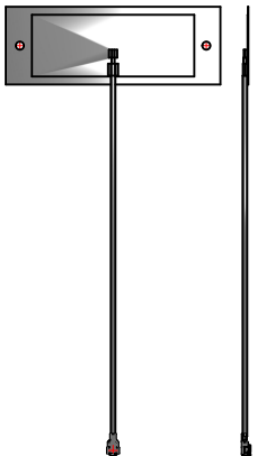
Je to energeticky úsporný NB-IoT modul, stejně jako GNSS modul obsahuje vše potřebné ke zpracování signálu, a s naším mikrokontrolerem komunikuje pomocí UARTU, GPIO, které je také převedené level shifterem a GPIO, které je uzemňováno (nebo odpojováno), skrze mosfety ovládané mikrokontrolerem. Zde také budeme využívat dostupné low power módy pro minimalizaci spotřeby. Je zde také problém s rozdílnými logickými úrovněmi, který je řešen stejně jako u L76L, jen s tím rozdílem že zde není zapotřebí externí LDO, protože jako referenci můžeme použít výstup LDO integrovaného v modulu BC66NA.



Obrázek 12-Diagram zapojení LPWAN modulu

Anténa

Anténa je vyvedena přes π filtr skrze u.fl konektor. Anténa bude přilepená z vnitřní strany krabičky.



Obrázek 13-3D model antény 2072350100

Propojení s mikrokontrolerem

BC66NA	MCU	Funkce
TXD	PB-09	UART pro komunikaci s MCU
RXD	PB-08	
PSM-EIN	PA-05	Externí interrupt, který probouzí modul ze sleep módu
RI	PA-04	Modul upozorní MCU že přišla nová data
PWRKEY	PA-03	Zapíná modul
RESET	PA-02	Resetuje modul
DBG_RXD	*	UART interface pro externí debuggování modulu, vyveden jen na dva header piny
DBG_TXD	*	

2.2.5 Akcelerometr a gyroskop BOSCH BMI270

Je kombinovaný senzor obsahující Akcelerometr, gyroskop a teploměr. Teploměr bude použit jen jako telemetrický údaj (zda-li se zařízení nepřehřívá). 16-bitový akcelerometr má maximální rozsah do 16g. A 16-bitový gyroskop má maximální rozsah 2000dps (stupňů za sekundu). Vzhledem k tomu že naším primárním cílem je trackování osob, či jen málo se pohybujících předmětů, jsou pro nás tyto rozsahy naprosto dostačující. Senzor má velice nízkou spotřebu i při maximálním výkonu, ale nachází se zde i low power módy. Velikou výhodou jsou dva programovatelné interrupt piny, díky kterým budeme moct MCU probouzet při zaznamenání pohybu senzorem.

BMI270	MCU	Funkce
INT1	PA-16	programovatelný interrupt
INT2	PA-17	programovatelný interrupt
SDX	PA-12	napojení na I2C sběrnici
SCX	PA-13	

2.3 Ochrana zařízení

PCB deska s komponenty samozřejmě nemůže být přímo vystavena vlivům okolí, v nynější fázi však není hotová finální krabička, kvůli jednoduchosti měření na jednotlivých komponentách použito jen jednoduché otevřené pouzdro na kterém je deska přišroubována (viz příloha 7.6).

3.0 Software

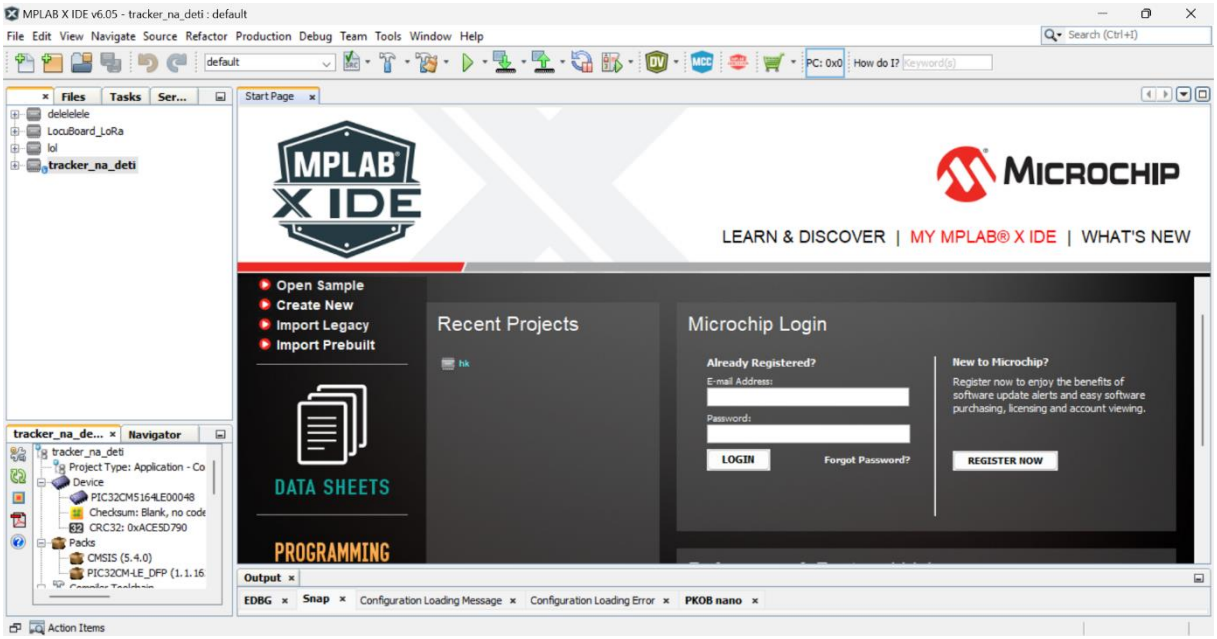
Firmware zařízení se zabývá sběrem dat z jednotlivých komponent na zařízení. Bude zde využita I2C sběrnice pro komunikaci s akcelerometrem a fuel gauge. Sběrnice UART bude použita pro nastavení GPS a příjem dat z GPS. Stejná komunikace bude využita pro odesílání dat na Narrow Band a následně komunikaci s ním.
Firmware zařízení musí brát ohled na spotřebu jednotlivých komponent, a proto pro zaručení co nejdelší životnosti zařízení, v něm musí být zakomponováno řešení, které bude co nejvíce low power.

3.1 Programovací prostředí

Díky zvolení mikrokontroleru PIC32 od společnosti Microchip Technology Inc. využíváme pro programování firmwaru našeho zařízení program MPLAB X IDE s nainstalovaným pluginem Harmony v3 a MPLAB Code Configurator. Firmware zařízení je programován pomocí embedded C (ANSIC). Pro kompilaci kódu je třeba stáhnout MPLAB XC Compiler 8/16/32. Jakou verzi compileru potřebujeme stáhnout je dáno mikrokontrolerem, který programujeme.

3.2 MPLAB X IDE

Program MPLAB X IDE slouží k vývoji a programování firmwaru embedded zařízení. Je to jednoduše rozšiřitelný a lehce konfigurovatelný software, který je určen převážně pro 8, 16 nebo 32 bitové mikrokontrolery vyráběné firmou Microchip. Pro programování se zde používá embedded C.
Výhodou programu MPLAB X IDE je možnost rychlého přidání doplňků a založení projektu. Díky podpoře všech mikrokontrolerů od společnosti Microchip nemusíme pro vývoj jiných zařízení používat a učit se pracovat v jiném prostředí. Další výhodou je Debug mode, který slouží k následnému debugování. Debug mode umožňuje naprogramovat zařízení a poté v prostředí MPLAB X IDE ukázat, v jaké části kódu se momentálně zařízení nachází. Prostředí dále nabízí možnost simulace. To znamená, že nemusíme mít k počítači připojené vyvíjené zařízení, ale funkčnost programu můžeme pouze simulovat



Obrázek 14-Základní obrazovka programu MPLAB X IDE

Program MPLAB X IDE je open source, tudíž nemusíme pro jeho využívání platit licenci a je volně ke stažení na oficiálních stránkách Microchip. Výhodou open source je také možnost úpravy vlastní úpravy softwaru a přidání či úpravy funkcí dle vlastní potřeby.
Pro naprogramování vyvíjené desky potřebuje, mimo fyzického připojení k desce, také programátor nebo debugger. Volně prodejné vývojové desky jej na sobě mají implementován, avšak my budeme používat externí debugger/programátor MPLAB Snap. Toto zařízení zajistí programování mikrokontroleru umístěného na desce. Podle typu použitého mikrokontroleru a vlastnosti vyvíjeného řešení máme více možností programátorů jako třeba MPLAB PICkit 4 a další.
Do programu MPLAB X IDE jsme si stáhli pluginy Harmony v3 a MPLAB Code Configurator. MPLAB Code Configurator patřil dříve pod plugin Harmony jako MPLAB Harmony Configurator.

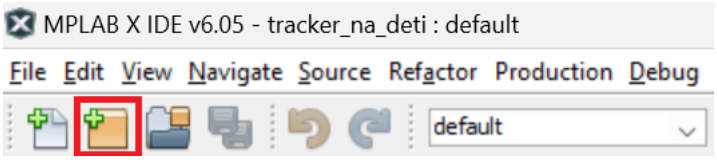
3.3 Harmony v3

Harmony v3 patří pod plugin MPLAB Code Configurator do programu MPLAB X IDE. Slouží k rychlejší a lehčí práci vývojářů pracujících na mikrokontrolerech od společnosti Microchip. Plugin obsahuje sadu knihoven a nástrojů pro jednodušší práci s programem. Díky tomu lze na začátku vývoje firmwaru zrychleně vytvořit základní strukturu programu.
Umožní rychlé nastavení periferií pro následnou práci s nimi. Harmony funguje pouze na 32 bitových mikrokontrolerech od firmy Microchip.

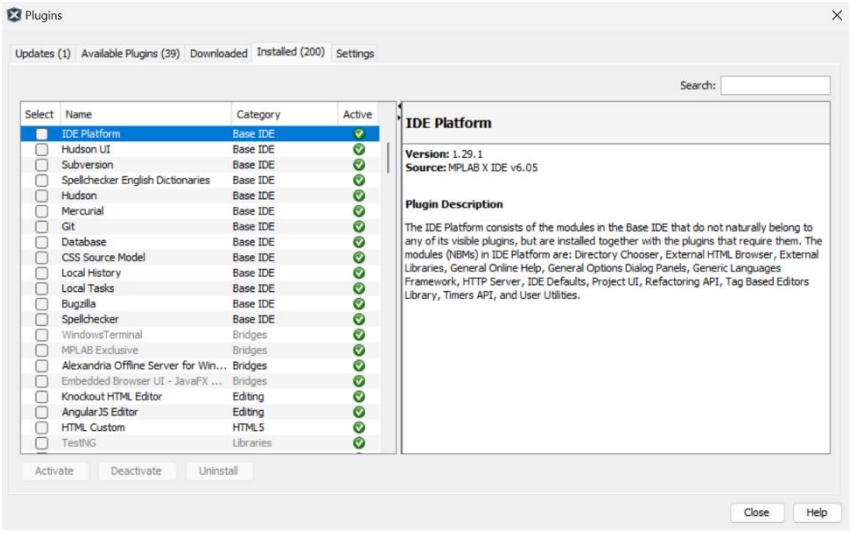
V tomto dialogovém okně pouze zaškrtneme box pro instalaci pluginů MPLAB Code Configurator.

3.3.1 Založení nového projektu pomocí MPLAB Code Configurator

Založení je poměrně jednoduché. Na začátek je potřeba instalovat plugin s MPLAB Code Configurator. Po jeho stažení založíme nový projekt a projdeme veškeré nastavení názvů a konfigurace, jak nám dialogová okna říkají.

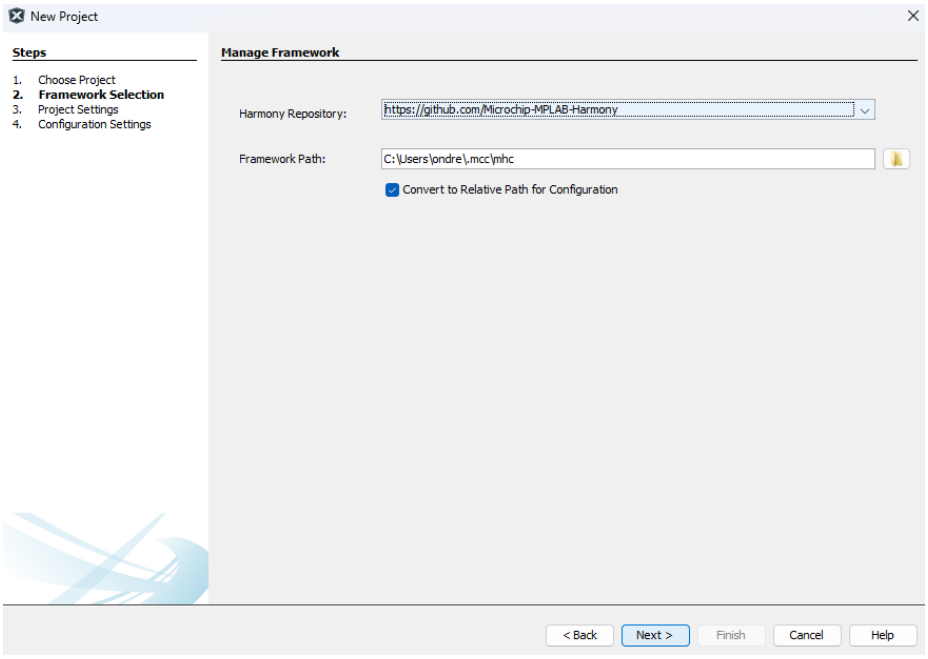


Obrázek 16-Návod na založení projektu -1



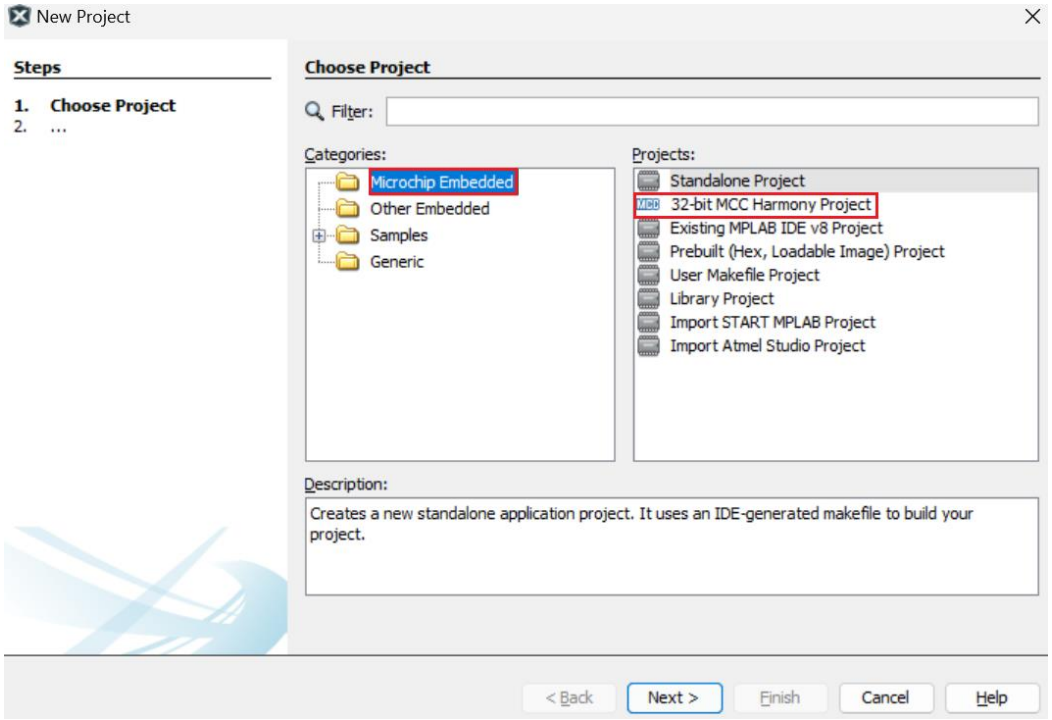
Obrázek 15-Dialogové okno pro instalaci nebo aktualizaci pluginů v programu MPLAB X IDE

Po zakliknutí ikony pro tvorbu nového projektu zvolíme ve vyskakovacím okně kategorii Microchip Embedded a projekt 32-bit MCC Harmony Project.



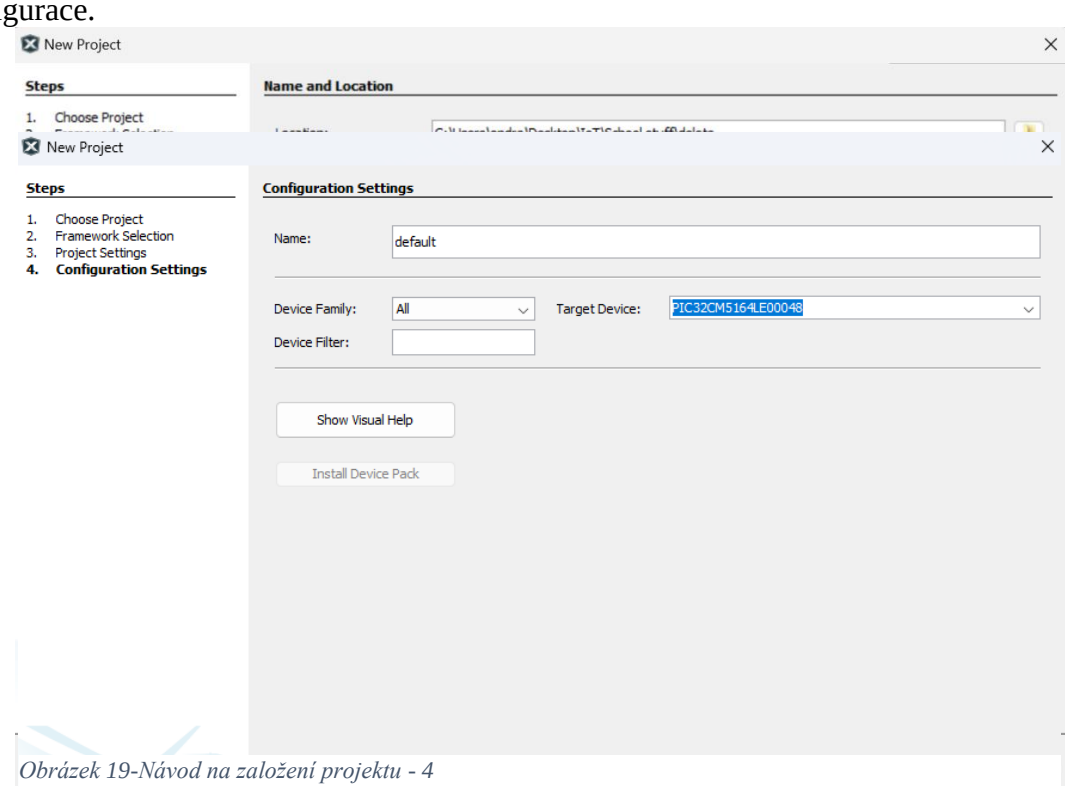
Obrázek 17-Návod na založení projektu - 2

V dalším kroku určujeme, v jakém adresáři na disku se bude nacházet framework projektu. Zvolíme si cílové místo projektu a jeho název.



Obrázek 18-Návod na založení projektu – 3

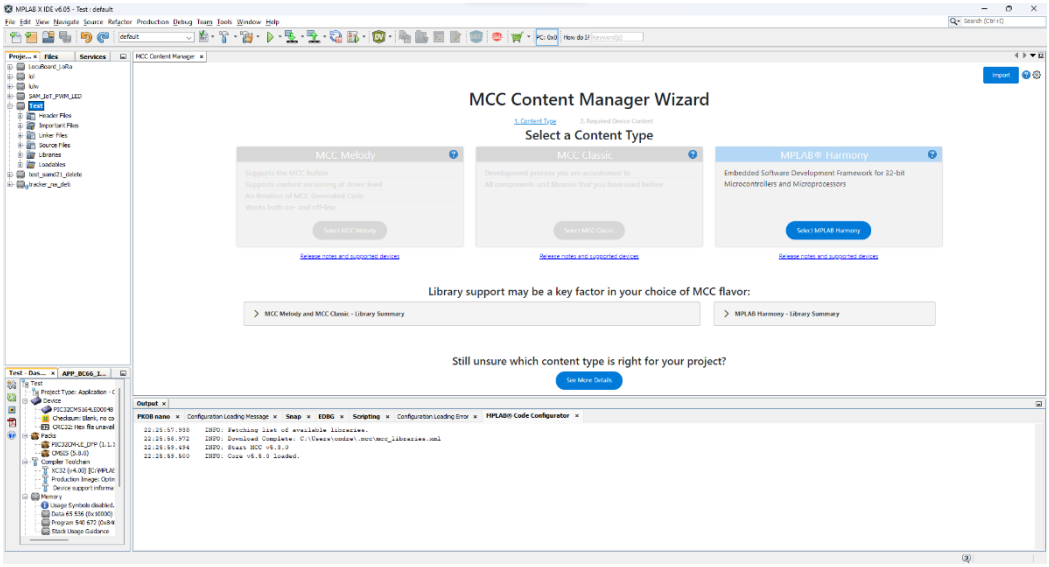
Nastavení konfigurace projektu provedeme zde.
Nastavíme si přesný model mikrokontroleru, pro který firmware vyvíjíme a jméno konfigurace.



Obrázek 19-Návod na založení projektu - 4

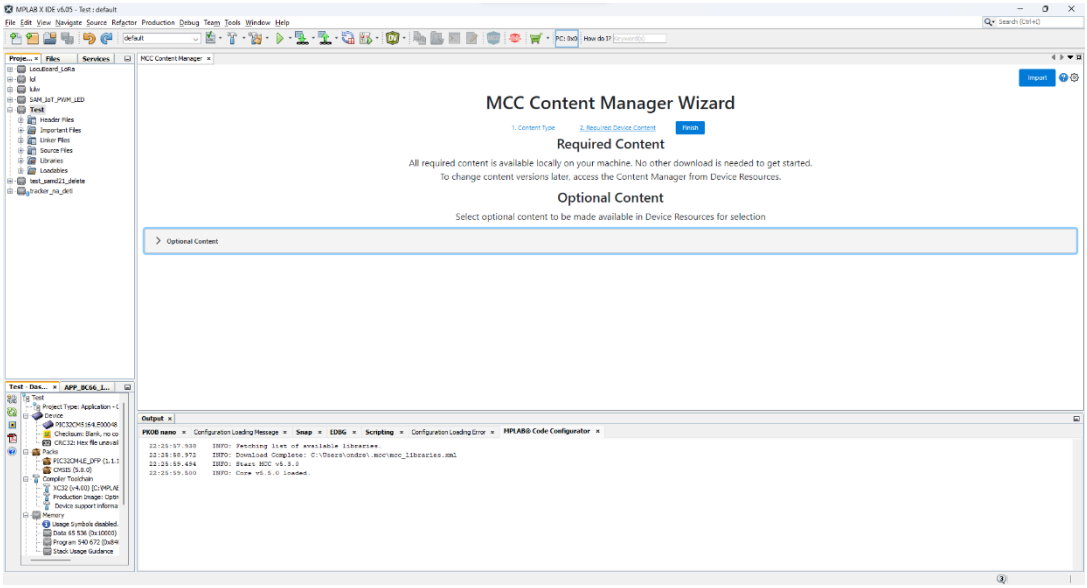
Obrázek 20-Návod na založení projektu - 5

Po stisku tlačítka finish se nám začne načítat MPLAB Code Configurator a zobrazí se nám okno MCC Content Manager Wizard. Zde si vybereme možnost MPLAB Harmony.



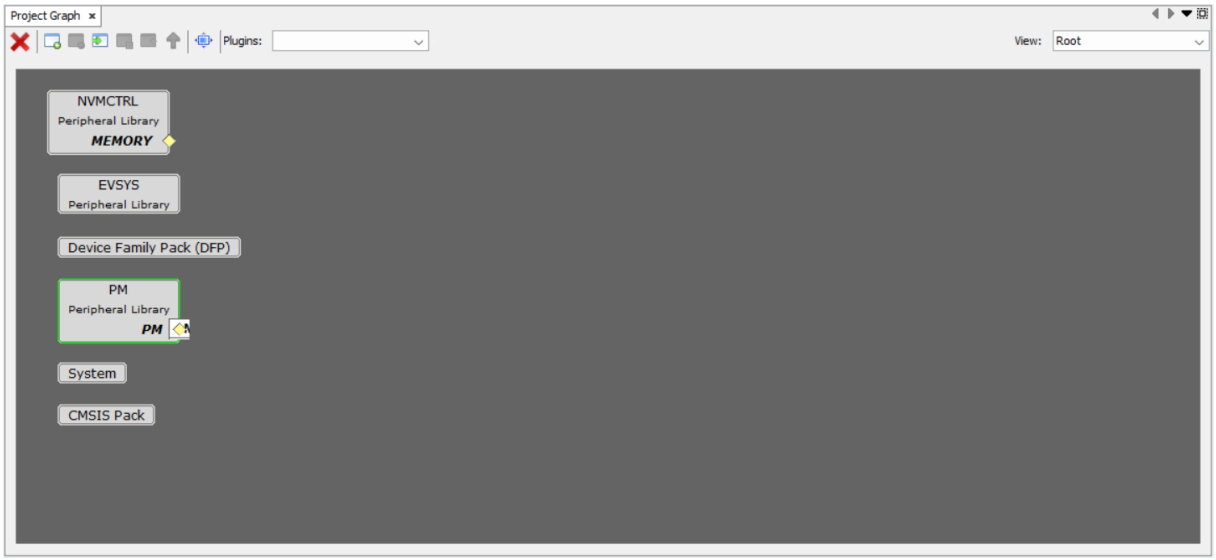
Obrázek 21-Návod na založení projektu – 6

V dalším okně je možnost instalace obsahu navíc. Tuto možnost zvolíme, pokud ji vyvíjený firmware vyžaduje. Jinak můžeme stisknout tlačítko finish.



Obrázek 22-Návod na založení projektu – 7

3.3.2 Projektový graf

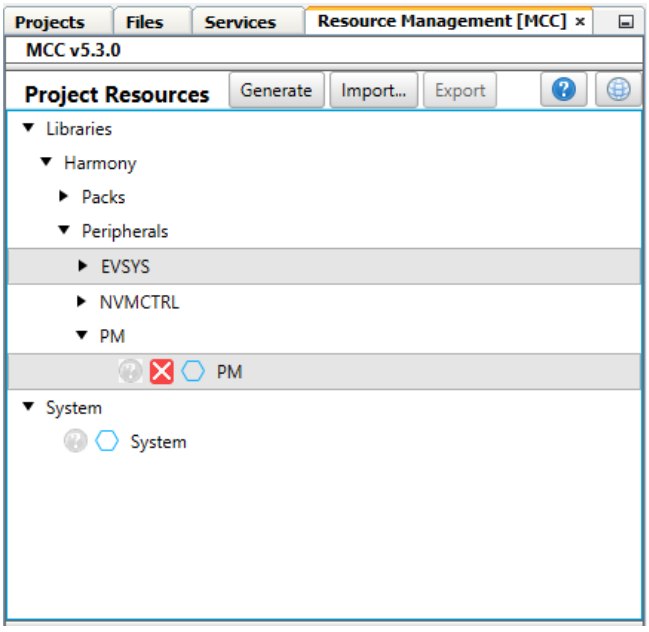


Obrázek 23-Projektový graf

Tato obrazovka slouží ke konfiguraci periferií v projektu a k jejich přidání či odebrání.

3.3.3 Project resources

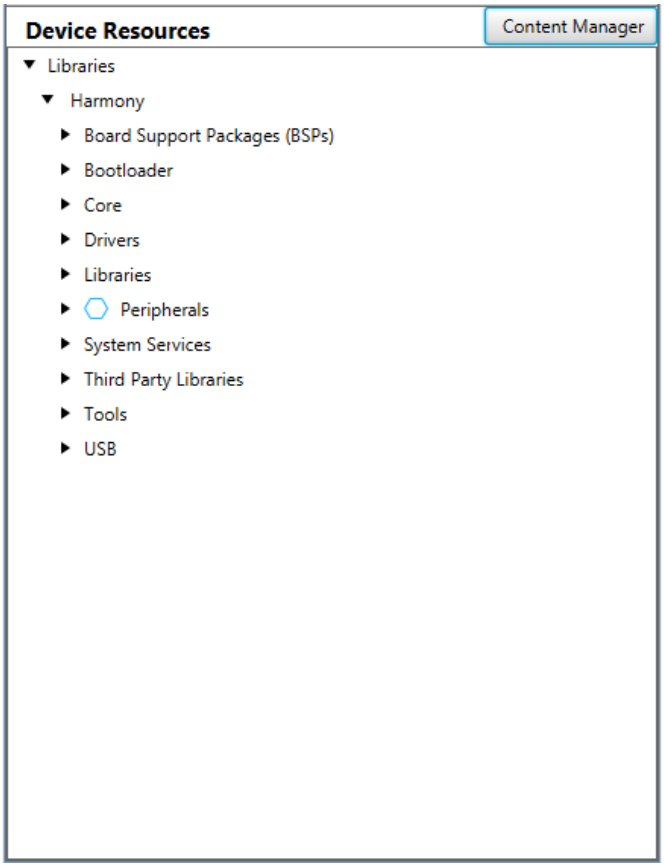
Toto okno slouží ke správě jednotlivých periferií přidaných do projektového grafu. Pomocí tlačítek na horní straně okna následně generujeme kód po úpravě vlastností periferií nebo jakéhokoliv zásahu do nastavení projektu.



Obrázek 24-Project Resources

3.3.4 Device Resources

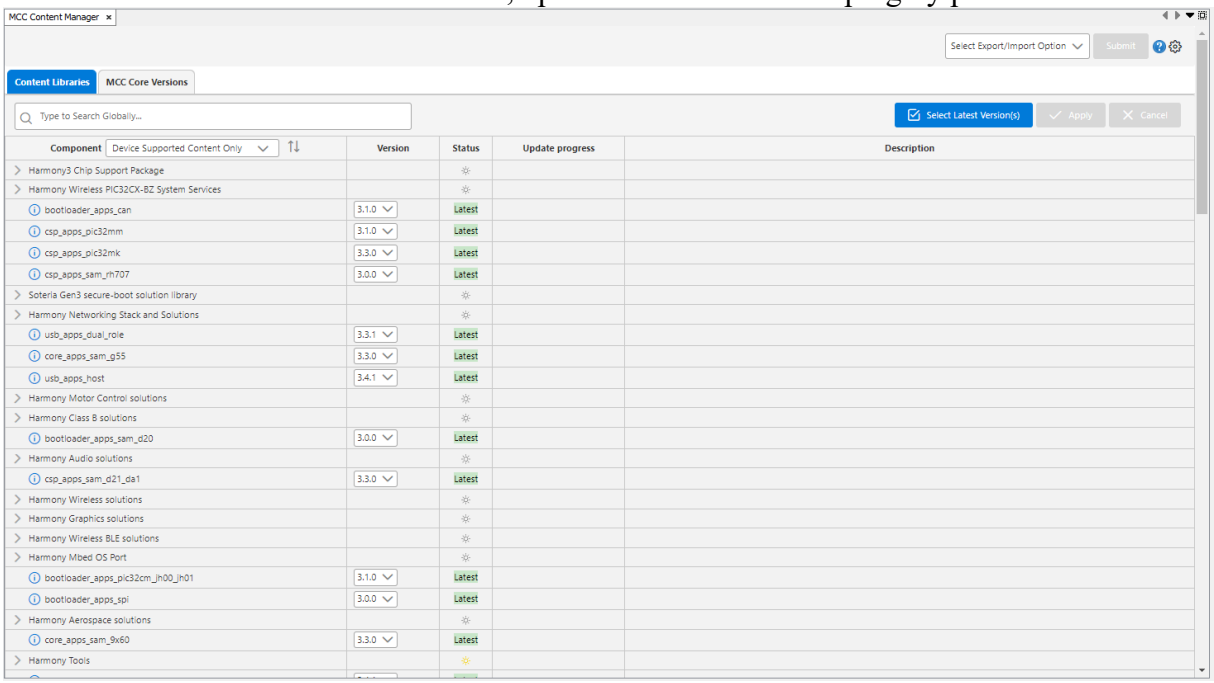
Zde si můžeme vybrat vlastnosti či periferie mikrokontroleru, které následně přidáváme do projektového grafu a používáme v aplikaci.
Pomocí tlačítka v pravém rohu spouštíme okno obsahu MCC.



Obrázek 25-Device Resources

3.3.5 Content Manager

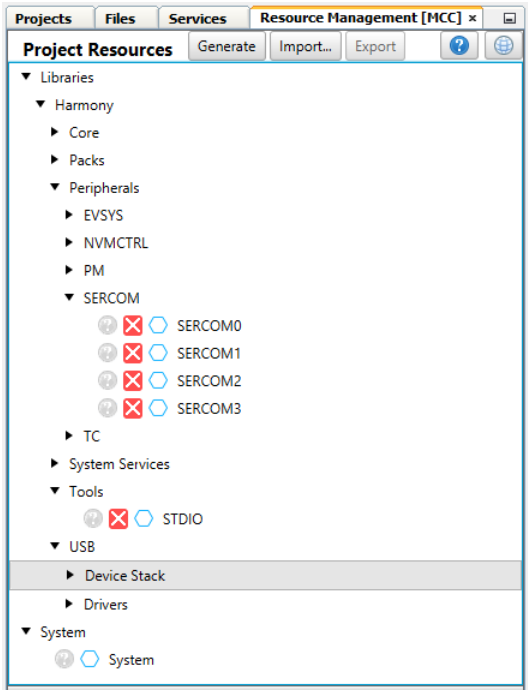
Přes toto okno máme možnost stahovat, updatovat či odstraňovat pluginy pro MCC.



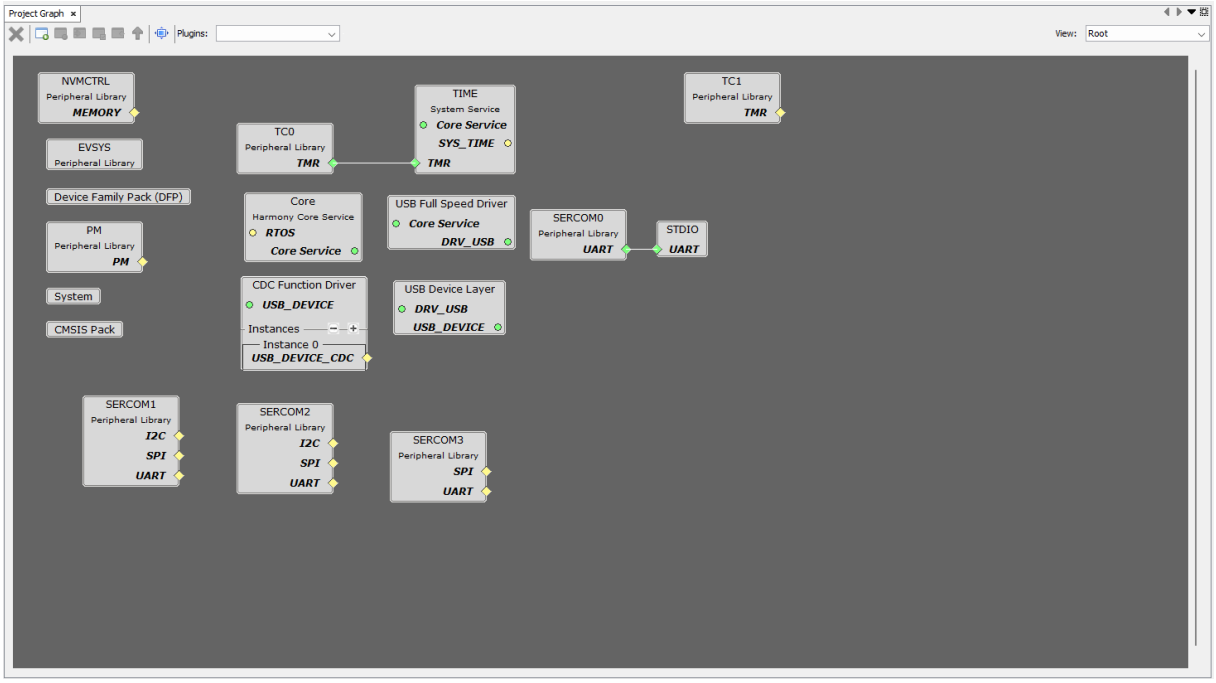
Obrázek 26-MCC Content Manager

3.4 Nastavení zařízení SCOUT

Nejdříve jsme si přesunuli všechny potřebné periferie z Device Resources do projektového grafu a Project Resources

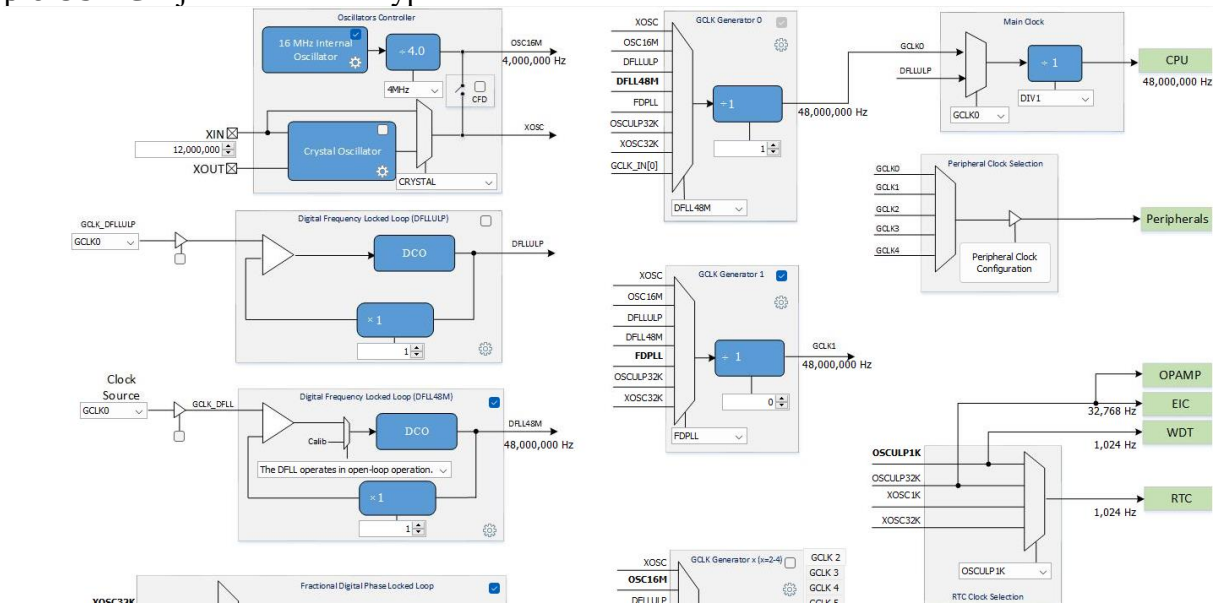


Obrázek 27-Nastavení Project Resources v aplikaci SCOUT

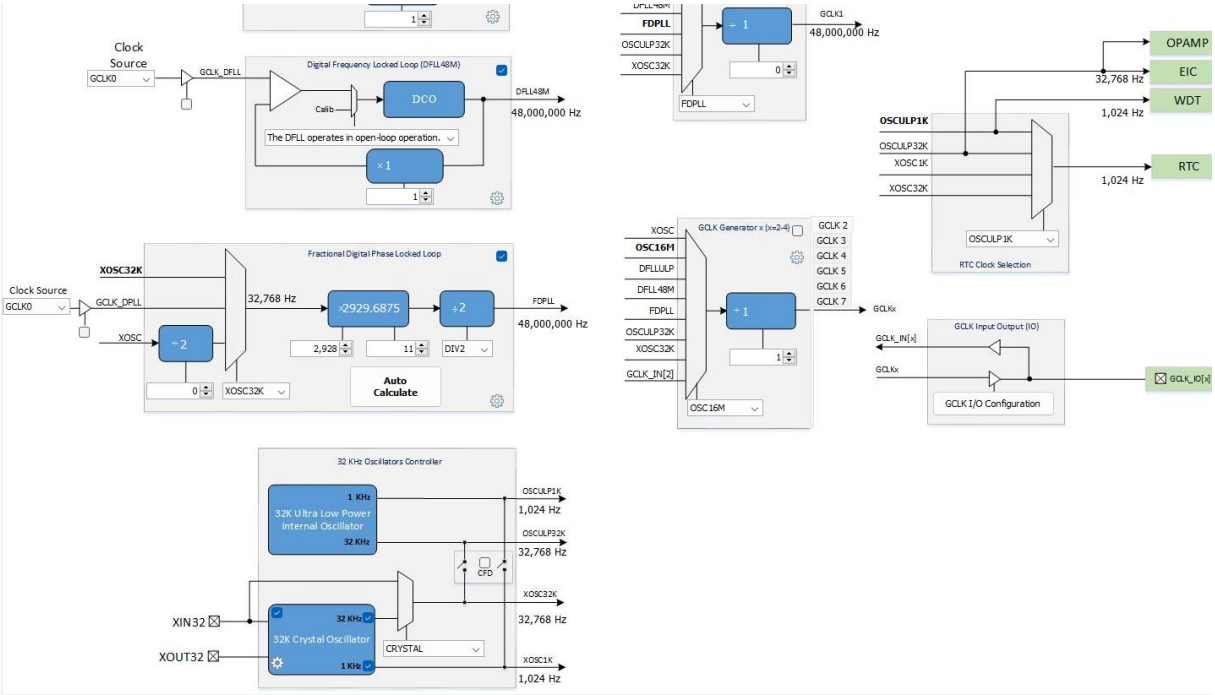


Obrázek 28-Project graph v aplikaci SCOUT

Protože víme, jaké periferie budeme potřebovat v naší aplikaci, tak budeme přibližně vědět i jejich nastavení. V aplikaci jsme použili veškerou periferii SERCOM a dva TC timery. Jeden je využíván k řízení interních hodin zařízení pomocí externího krystalu. Tyto hodiny jsou pak násobičem převáděny na vyšší hodnoty a tato frekvence je využívána pro USB C. Druhý využíváme pro funkci delay. Máme zde periferii STDIO, která je využívána pomocí driverů pro USB C k jednoduchému vypisování do konzole.



Obrázek 29-Interní nastavení hodin - 1



Obrázek 30--interní nastavení hodin - 2

Z designu desky víme, jaké piny mikrokontroleru využíváme k čemu, a proto je tak nastavíme pomocí Pin Configuration v našem projektu. Jakých funkcí tyto piny mohou nabývat se dozvíme z datasheetu k mikrokontroleru.

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
3	PA02	BC66NA_RESET	GPIO	Digital	Out	Low	☐	☐	NORMAL
4	PA03	BC66NA_PWRKEY	GPIO	Digital	Out	Low	☐	☐	NORMAL
5	AVSS			Digital	High Impedance	Low	☐	☐	NORMAL
6	AVDD			Digital	High Impedance	Low	☐	☐	NORMAL
7	PB08	BC66NA_TX	SERCOM3_PAD0	Digital	High Impedance	n/a	☐	☐	NORMAL
8	PB09	BC66NA_RX	SERCOM3_PAD1	Digital	High Impedance	n/a	☐	☐	NORMAL
9	PA04	BC66NA_RI	EIC_EXTINT4	Digital	In	n/a	☐	☐	NORMAL
10	PA05	BC66NA_EINT	GPIO	Digital	Out	Low	☐	☐	NORMAL
11	PA06	L76L_STANDBY	GPIO	Digital	Out	Low	☐	☐	NORMAL
12	PA07	L76L_RESET	GPIO	Digital	Out	Low	☐	☐	NORMAL
13	PA08	L76L_TX	SERCOM1_PAD0	Digital	High Impedance	n/a	☐	☐	NORMAL
14	PA09	L76L_RX	SERCOM1_PAD1	Digital	High Impedance	n/a	☐	☐	NORMAL
15	PA10	L76L_WKUP	GPIO	Digital	Out	Low	☐	☐	NORMAL
16	PA11	PA11_DNC	GPIO	Digital	In	Low	☐	☑	NORMAL
17	AVSSPLL			Digital	High Impedance	Low	☐	☐	NORMAL
18	VDD			Digital	High Impedance	Low	☐	☐	NORMAL
19	VSS			Digital	High Impedance	Low	☐	☐	NORMAL
20	VDDPLL			Digital	High Impedance	Low	☐	☐	NORMAL
21	PA12	I2C_SDA	SERCOM2_PAD0	Digital	High Impedance	n/a	☐	☐	NORMAL
22	PA13	I2C_SCL	SERCOM2_PAD1	Digital	High Impedance	n/a	☐	☐	NORMAL
23	PA14	PA14_DNC	GPIO	Digital	In	Low	☐	☑	NORMAL
24	PA15	PA15_DNC	GPIO	Digital	In	Low	☐	☑	NORMAL
25	PA16	BMI270_INT1	EIC_EXTINT5	Digital	High Impedance	n/a	☐	☐	NORMAL
26	PA17	BMI270_INT2	EIC_EXTINT6	Digital	High Impedance	n/a	☐	☐	NORMAL
27	PA18	MAX1760_ALRT	EIC_EXTINT7	Digital	High Impedance	n/a	☐	☐	NORMAL
28	PA19	PA19_DNC	GPIO	Digital	In	Low	☐	☑	NORMAL
29	PA20	PWM_BUZZER	TCC0_WO6	Digital	High Impedance	n/a	☐	☐	NORMAL
30	PA21	LED3	GPIO	Digital	In/Out	Low	☐	☐	NORMAL
31	PA22	LED1	GPIO	Digital	In/Out	Low	☐	☐	NORMAL
32	PA23	LED2	GPIO	Digital	In/Out	Low	☐	☐	NORMAL
33	PA24	D_N	USB_DM	Digital	High Impedance	n/a	☐	☐	NORMAL
34	PA25	D_P	USB_DP	Digital	High Impedance	n/a	☐	☐	NORMAL
35	VSS			Digital	High Impedance	Low	☐	☐	NORMAL
36	VDD			Digital	High Impedance	Low	☐	☐	NORMAL
37	PB22	PB22_DNC	GPIO	Digital	In	Low	☐	☑	NORMAL
38	PB23	PB23_DNC	GPIO	Digital	In	Low	☐	☑	NORMAL
39	VSS			Digital	High Impedance	Low	☐	☐	NORMAL

Obrázek 31-Nastavení pinů v našem projektu

3.5 Programování a ukázky kódu

Zařízení programujeme pomocí jazyka embedded C a aplikace MPLAB X IDE. Na desce nemáme programátor/debugger, tudíž využíváme externí programátor MPLAB Snap. Firmware zařízení je ve velmi brzkém stádiu vývoje, a ne vše zatím funguje. O tom více v části debugging.

3.5.1 STDIO

STDIO je knihovna pro funkce printf, scanf atd. Aby to mohlo fungovat s driverem pro USB CDC, museli jsme pozměnit její nastavení.

```
#include <stddef.h>
#include "definitions.h"

extern int read(int handle, void *buffer, unsigned int len);
extern int write(int handle, void * buffer, size_t count);

USB_DEVICE_CDC_TRANSFER_HANDLE transferHandle;
volatile extern APP_DATA appData;


int read(int handle, void *buffer, unsigned int len)
{
int nChars = 0;
bool success = false;
(void)len;
if ((handle == 0) && (len > 0))
{
do
{
success = SERCOM0_USART_Read(buffer, 1);
}while( !success);
nChars = 1;
}
return nChars;
}

int write(int handle, void * buffer, size_t count) {
USB_DEVICE_CDC_RESULT write_result = 0;
char data_tx[count];
if (handle == 1) {
strncpy(data_tx, (char *) buffer, sizeof(data_tx) );
appData.isWriteComplete = false;
write_result = USB_DEVICE_CDC_Write(USB_DEVICE_CDC_INDEX_0,
&transferHandle, &data_tx, count,
USB_DEVICE_CDC_TRANSFER_FLAGS_DATA_COMPLETE);
while (!appData.isWriteComplete) {};
if( write_result != USB_DEVICE_CDC_RESULT_OK)
{
//Error handling
}
}
return count;
}
```

3.5.2 Soubor main.c

Obrázek 329-Kód pro zajištění debug konzole

V tomto souboru voláme jednotlivé soubory s kódem pro jednotlivá zařízení připojená k mikrokontroleru. Tyto soubory voláme přes funkce SYS_Tasks a SYS_Initialize.

```
#include "definitions.h" // SYS function prototypes

int main ( void )
{
/* Initialize all modules */
SYS_Initialize ( NULL );

while ( true )
{
SYS_Tasks ( );

}

/* Execution should not come here during normal operation */

return ( EX
}
```

Obrázek 33-Soubor main.c

```
void SYS_Tasks ( void ){
```

Obrázek 34-Funkce main.c

```
/* Maintain Middleware & Other Libraries */
/* USB Device layer tasks routine */
USB_DEVICE_Tasks(sysObj.usbDevObject0);

/* USB FS Driver Task Routine */
DRV_USBFSV1_Tasks(sysObj.drvUSBFSV1Object);


/* Maintain the application's state machine. */
/* Call Application task APP. */
APP_Tasks();

/* Call Application task APP_GPS. */
APP_GPS_Tasks();

//Call app for bc66
APP_BC66_Tasks();
}
```

Obrázek 31-Funkce SYS_Tasks

```
void SYS_Initialize ( void* data ){

PM_Initialize();

NVMCTRL_REGS->NVMCTRL_CTRLB = NVMCTRL_CTRLB_RWS(2);

STDIO_BufferModeSet();

PORT_Initialize();

CLOCK_Initialize();

SERCOM3_USART_Initialize();

SERCOM2_I2C_Initialize();

NVMCTRL_Initialize();

SERCOM1_USART_Initialize();

SERCOM0_USART_Initialize();

EVSYS_Initialize();

SYSTICK_TimerInitialize();
TC1_TimerInitialize();

TC0_TimerInitialize();

TC2_TimerInitialize();

sysObj.sysTime = SYS_TIME_Initialize(SYS_TIME_INDEX_0, (SYS_MODULE_INIT
*)&sysTimeInitData);

sysObj.usbDevObject0 = USB_DEVICE_Initialize (USB_DEVICE_INDEX_0 , (
SYS_MODULE_INIT* ) & usbDevInitData);

sysObj.drvUSBFSV1Object = DRV_USBFSV1_Initialize(DRV_USBFSV1_INDEX_0,
(SYS_MODULE_INIT *) &drvUSBInit);

APP_Initialize();
APP_GPS_Initialize();
NVIC_Initialize();
}
```

Obrázek 35-Funkce SYS_Initialize

3.5.3 Program pro BC66

Tento program se zaobírá odesíláním dat pomocí BC66. Na začátku si vypíšeme veškeré AT příkazy sloužící k ovládání BC66 a následně je použijeme při inicializaci.
Program je nyní pouze demo, které posílá AT příkazy na BC66 a zároveň vyčítá data zpět.
Na začátku inkludujeme veškeré potřebné knihovny a zaznamenáme potřebné AT příkazy.
Následně si přizpůsobíme funkci APP_BC66_Initialize. Ta proběhne pouze jednou na začátku kódu. Sem dáme veškeré potřebné věci, co je ze začátku třeba nastavit, zapnutí modulu, poslání AT příkazů kvůli stabilizaci Baud Rate. Poté se přesuneme na APP_BC66_Tasks. Tato funkce běží stále dokola a pomocí stavového automatu můžeme přecházet mezi aktivními částmi jednoduše na základě různých parametrů. Aplikace funguje, avšak kvůli hardwarové chybě na zařízení ji není možné nahrát.
Pomocí externího připojení na UART BC66 jsme byli schopni zařízení nastavit a odeslat data, která jsme následně i úspěšně přijali ve webovém rozhraní.

```
#include "app_BC66_nb.h"

APP_BC66_DATA app_bc66Data;
extern APP_DATA appData;
volatile bool TC2Done = false;
char nbData[128];
char ATNetlightOn [] = "AT+QLEDMODE=1\r";
char ATI[] = "ATI\r";
char OK[] = "OK\r";
char ATOpenPorts[] =
"AT+QIOPEN=1,0,\"UDP\",\"192.168.0.20\",4242,5000,0\r";
char ModuleReset[] = "AT+QRST=1";
char ATSend [] = "AT+QISEND=0,10,1303232155\r";


void TC2_Callback(TC_TIMER_STATUS status, uintptr_t context){
    TC2Done = true;
}
```

Obrázek 36-Program BC66 – 1

```
void APP_BC66_Initialize (void){
    app_bc66Data.state = APP_BC66_INIT;
    TC2_TimerCallbackRegister(TC2_Callback, 0);
    TC2_TimerStart();
    SYSTICK_TimerStart();
    BC66NA_PWRKEY_Toggle();
    SYSTICK_DelayMs(500);
    BC66NA_PWRKEY_Toggle();
    LED3_Toggle();
    if(SERCOM3_USART_ReceiverIsReady() == true){
               SERCOM3_USART_Read(nbData, 128);
               if (appData.isConfigured){
               printf("%s\r\n", nbData);
               }
          }
    if (TC2Done){
        if(appData.isConfigured){
            printf("MOVING FROM INIT\r\n");
        }
        TC2Done = false;
    }
}
```

Obrázek 37-Inicializace BC66 – 2

```
void APP_BC66_Tasks (void){
    switch (app_bc66Data.state){
        case APP_BC66_INIT:{
            bool appInitialized = true;

            if(appInitialized){
                app_bc66Data.state = APP_BC66_PRINTF_STATE;
            }
            break;
        }

        case APP_BC66_PRINTF_STATE:{
            SERCOM3_USART_Write(ModuleReset, sizeof(ModuleReset) -1);
            SERCOM3_USART_Write(ATOpenPorts, sizeof(ATOpenPorts) -1);
            LED3_Toggle();
            if(TC2Done){
                TC2Done = false;
            }
            SERCOM3_USART_Write(ATSend, sizeof(ATSend) -1);
            LED3_Toggle();
            if(SERCOM3_USART_ReceiverIsReady() == true){
               SERCOM3_USART_Read(nbData, 128);
               if (appData.isConfigured){
               printf("%s\r\n", nbData);
               }
           }
            bool printfState = true;

            if(printfState){
                app_bc66Data.state = APP_BC66_SERVICE_TASKS;
            }
            break;
        }

        case APP_BC66_SERVICE_TASKS:{
            SERCOM3_USART_Write(ATI, sizeof(ATI) - 1);
            if(TC2Done){
                TC2Done = false;
            }
            if(SERCOM3_USART_ReceiverIsReady() == true){
               SERCOM3_USART_Read(nbData, 128);
               if (appData.isConfigured){
               printf("%s\r\n", nbData);

               }
           }
            break;
        }

        default:{

            break;
        }
```

```
    }
}
```

Obrázek 38-Funkce Tasks - 3

3.5.4 Program pro GPS

Tento program nám nastavuje a vyčítá data o GPS poloze z modulu L76-L.
Struktura je stejná jako u BC66. Jediný rozdíl je, že místo AT příkazů používáme PMTK příkazy.
Ve funkci APP_GPS_Initialize si nastavíme veškeré potřebné hodnoty po zapnutí modulu.
Ve funkci APP_GPS_Tasks budou probíhat jednotlivé stavy state machine.

```
#include "app_gps.h"

APP_GPS_DATA app_gpsData;
extern APP_DATA appData;
volatile bool timerDone = false;
char gpsData[128];
char PmtkTxtMsg[] = "$PMTK011\r";
char isEasyEnabled [] = "$PMTK869,029\r";
```

Obrázek 39-Program GPS – 1

```
void TimerCallback(TC_TIMER_STATUS status, uintptr_t context){
    timerDone = true;
}

 void GPS_Read_Callback( uintptr_t context )//is not in use
 {
     LED2_Toggle();
     SERCOM1_USART_Read(gpsData, 1);
 }
```

Obrázek 40-Funkce použité v programu – 2

```
void APP_GPS_Initialize ( void )
{
    /* Place the App state machine in its initial state. */
    app_gpsData.state = APP_GPS_STATE_INIT;
    TC1_TimerCallbackRegister(TimerCallback, 0);
    TC1_TimerStart();


    /* TODO: Initialize your application's state machine and other
     * parameters.
     */
}
```

Obrázek 41-Inicializace modulu L76-L – 3

```
void APP_GPS_Tasks ( void )
{

    /* Check the application's current state. */
    switch ( app_gpsData.state ){
        /* Application's initial state. */
        case APP_GPS_STATE_INIT:{
            bool appInitialized = true;

            if (appInitialized)
            {
                app_gpsData.state = APP_GPS_STATE_SERVICE_TASKS;
            }
            break;
        }

        case APP_GPS_STATE_SERVICE_TASKS:{
            SERCOM1_USART_Write(PmtkTxtMsg, sizeof(PmtkTxtMsg) -1);
            SERCOM1_USART_Write(isEasyEnabled, sizeof(isEasyEnabled) -1);
            if(SERCOM1_USART_ReceiverIsReady() == true){
                LED2_Toggle();
                SERCOM1_USART_Read(gpsData, 128);
                if (appData.isConfigured){
                printf("%s", gpsData);
                }
            }
            break;
        }
        default:
        {
          break;
        }}}
```

Obrázek 42-Funkce Tasks - 4

3.6 Log dat přijatých z GPS modulu

GPS modul komunikuje pomocí sběrnice UART přes NMEA protokol. Modul nám posílá veškerá data, která přijme, to znamená i data která nepotřebujeme. Tato data jsou následně zpracována v mikrokontroleru a upravena pro naši potřebu filtrováním.
Polohu budeme zjišťovat pomocí NMEA zprávy $GPRMC. Po jejím upravení získáme souřadnice polohy, kde se zařízení nachází.

```
4,26,18,293,,20,15,078,22,18,12,185,14,05,05,106,,0*6F
$GPGSV,4,4,14,32,04,233,,04,01,348,,0*6D
73,170,,82,52,042,19,80,46,130,20,1*71
$GLGSV,2,2,08,74,25,316,,84,13,204,,65,11,352,,81,05,032,,1*72
.5775,E,133754.000,A,A*45
$GNRMC,133755.000,A,4946.6487,N,01341.5775,E,0.00,58.89,250223,,,A,V*31
0,K,A*018.89,T,,M,0.00,N,0.0h
$GPGGA,133755.000,4946.6487,N,01341.5775,E,1,9,1.25,431.1,M,46.1,M,,*54
5,0.93,1*0125,12,29,20,11,31,02,,,,,1.55,1.2h
$GNGSA,A,3,80,73,,,,,,,,,,1.55,1.25,0.93,2*03
DGPGSV,4,1,13,29,83,233,20,25,62,106,14,28,56,268,,31,52,283,16,0*6h
$GPGSV,4,2,13,02,47,157,23,12,31,104,18,11,28,049,29,26,18,293,,0*66
4,233,,0*6913,20,15,078,21,18,12,185,14,05,05,106,,32,0h
$GPGSV,4,4,13,04,01,348,,0*5D
$GLGSV,2,1,08,73,73,321,19,83,73,170,,82,52,042,19,80,46,130,19,1*71
74,25,316,,84,13,204,,65,11,352,,81,05,032,,1*72
$GNGLL,4946.6487,N,01341.5775,E,133755.000,A,A*44
87,N,01341.5775,E,0.00,58.89,250223,,,A,V*32
$GPVTG,58.89,T,,M,0.00,N,0.00,K,A*01
,8,1.26,431.1,M,46.1,M,,*55,N,01341.5775,E,1h
$GNGSA,A,3,12,29,20,11,31,02,,,,,,,1.57,1.26,0.95,1*01
*04GSA,A,3,80,73,,,,,,,,,,1.57,1.26,0.95,2h
$GPGSV,4,1,14,29,83,233,20,25,62,106,14,28,56,268,,31,52,283,15,0*69
26,18,293,,0*662,47,157,24,12,31,104,18,11,28,049,29,h
$GPGSV,4,3,14,40,16,122,,20,15,078,21,18,12,185,13,05,05,106,,0*6C
$GPGSV,4,4,14,32,04,233,,04,01,348,,0*6D
GLGSV,2,1,08,73,73,321,19,83,73,170,,82,52,042,18,80,46,130,18,1*71
*72GSV,2,2,08,74,25,316,,84,13,204,,65,11,352,,81,05,032,,1h
$GNGLL,4946.6487,N,01341.5775,E,133756.000,A,A*47
hGNRMC,133757.000,A,4946.6487,N,01341.5775,E,0.00,58.89,250223,,,A,V*33
```

Obrázek 43-Příklad přijaté komunikace z L76-L

Z tohoto logu můžeme vynést pouze data která potřebujeme, tedy:
$GNRMC, 133755.000,A,4946.6487,N,01341.5775,E,0.00,58.89,250223

Po správném rozdělení všech jednotlivých částí nám vznikne:

UTC = 13 h 37 m 55 s
Status připojení k satelitní síti = A (Připojeno)
Geografická šířka = 4946.6487
Směr = N (sever)
Geografická délka = 1341.5775
Směr = E (východ)

Z těchto dat tedy můžeme následně vytvořit zprávu, kterou odešleme.

3.5.5 BMI270

Senzor BMI270 nám bude sloužit jako externí interrupt pro probuzení modulů. Abychom dosáhli vyšší životnosti baterie, tak musíme zařízení přepínat do režimů spánku. Aplikace pro tento senzor nám zařídí probuzení dalších modulů při pohybu zařízení. Pro výčet dat zde využijeme I2C sběrnici. Gyroskop nám tak bude poskytovat parametry o zrychlení zařízení a dodá i bezpečnost. Díky této komponentě můžeme monitorovat náhlé pády a díky tomu následně predikovat problémy se zařízením nebo uživatelem.

3.5.6 MAX17260

Tento fuel gauge nám bude udávat hodnotu napětí na baterii a zároveň sledovat stav baterie. Bude komunikovat přes I2C sběrnici. Data budou zaznamenávána do proměnné a následně i s daty o poloze vysílána přes BC66 do aplikace.
Díky této komponentě dokážeme prodloužit životnost baterie. Velkou výhodou je jednoduchost implementace zařízení. Díky algoritmům implementovaných v samostatné komponentě je tak podstatně snížen nárok na firmware v zařízení.

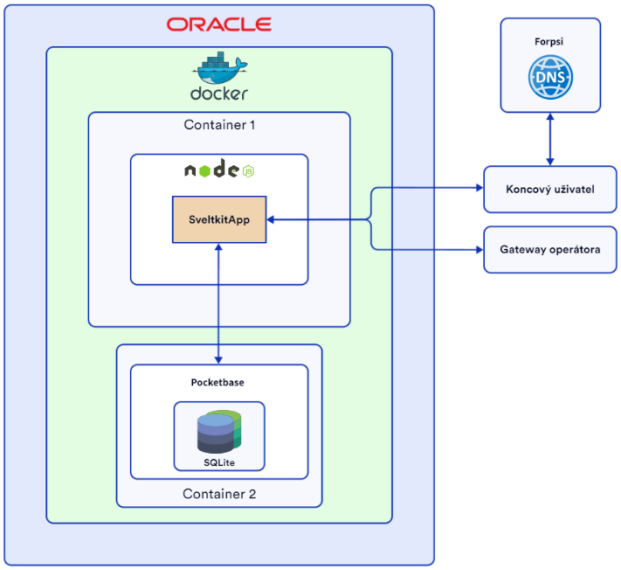
4.0 Webová aplikace

Webovou aplikaci jsme se snažili navrhnout hlavně s ohledem na bezpečnost, protože si uvědomujeme citlivost dat o poloze.

Celý projekt je přístupný na GitHubu: https://github.com/libormiller/scout-web

4.1 Struktura

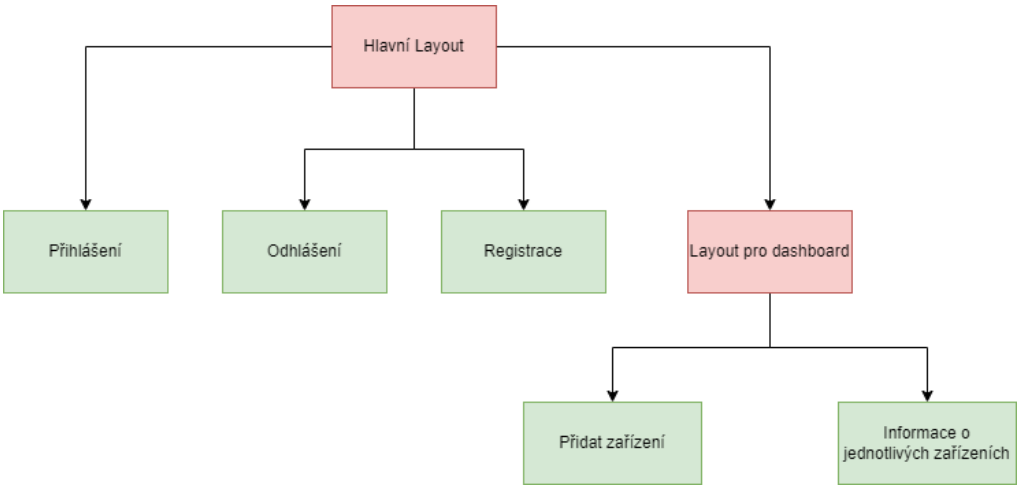
Využíváme možnosti virtuálního serveru u firmy Oracle, na němž běží Ubuntu s nainstalovaným Dockerem, v jednom containeru běží NodeJS s naší webovou aplikací vytvořenou pomocí full-stack framworku sveltekit a typescriptu a v druhém containeru je pocketBase, což je BackendAsAService s databází v podobě SQLite. Zároveň jsme také zakoupili doménu od českého poskytovatele Forpsi abychom mohly využít šifrování SSL.



Obrázek 44-struktura webového řešení

4.2 SvelteKit aplikace

Tento fullstack framework využívá jako svojí UI knihovnu Svelte, což znamená, že celý web, na který má uživatel přístup je napsán v něm, aplikace se skládá z několika komponent. Jedna komponenta se skládá z jedné nebo dvou částí, v případě že část operací musí být provedena na serveru (např. načtení dat). Pro stylizaci stránek je použit TailwindCSS.



Obrázek 45-Struktura jednotlivých komponent ve webové aplikaci

Příklad komponenty Layout pro dashboard skládající se z +layout.svelte a +layout.ts.

+layout.svelte

```
<script lang="ts">
    import { each } from "svelte/internal"
/* načtení dat z +Layout.ts ()*/
export let data;
</script>

<div class="flex space-x-4 items-start max-h-full">
<div>
<ul class="menu bg-base-200 w-56 p-2 rounded-box shadow-xl">
    <li>
      <!-- odkaz na komponentu ve které uživatel přidá zařízení -->
      <a href="/dashboard/add-device" >
        <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" fill="none" stroke="#000000" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round"><line x1="12" y1="5" x2="12"
y2="19"></line><line x1="5" y1="12" x2="19" y2="12"></line></svg>
      Přidat zařízení
      </a>
    </li>
    <!-- list vygenerovaný ze zařízení konkrétního uživatele, které byly
uloženy v databázi -->
    {#each data.devices as device}
    <li>
      <!-- Odkaz na dynamické stránky které se generují z dat pro každé
načtené zařízení -->
      <a data-sveltekit-reload href="/dashboard/map/{device.deviceID}">
        <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" fill="none" stroke="#000000" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round"><circle cx="12" cy="10" r="3"/><path
d="M12 21.7C17.3 17 20 13 20 10a8 8 0 1 0-16 0c0 3 2.7 6.9 8 11.7z"/></svg>
         Zařísení s ID: {device.deviceID}
      </a>
    </li>
    {/each}
  </ul>
</div>
<div class="container max-w-full ">
<div class="bg-base-200 rounded-box shadow-xl relative">
<slot />
</div>
</div>
</div>
```

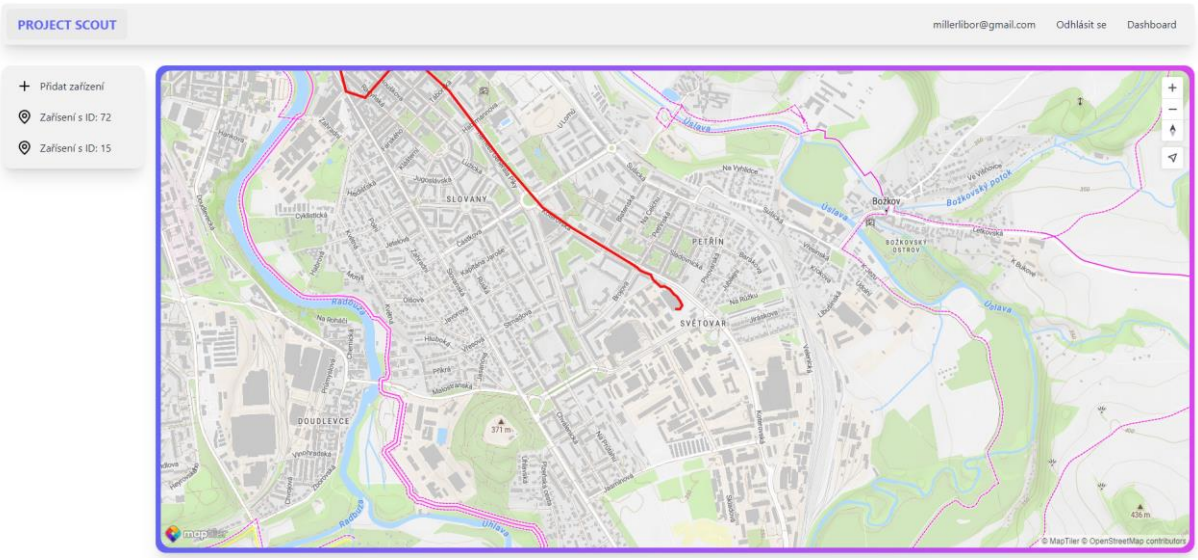
+layout.ts

```
import { pb } from "$lib/pocketbase"

/* Asynchronní funkce která načítá data z databáze */
export const load = (async({}) => {

    /* sdk pošle dotaz na API naší běžící instance PocketBase */
    const records = await pb.collection('devices').getFullList()
    return {
        devices: records
    }
})
```

Pro práci s mapou využíváme SDK od maptileru, aby se data vykreslila, jsou načtena z databáze a následně převedena do formátu geoJSON.



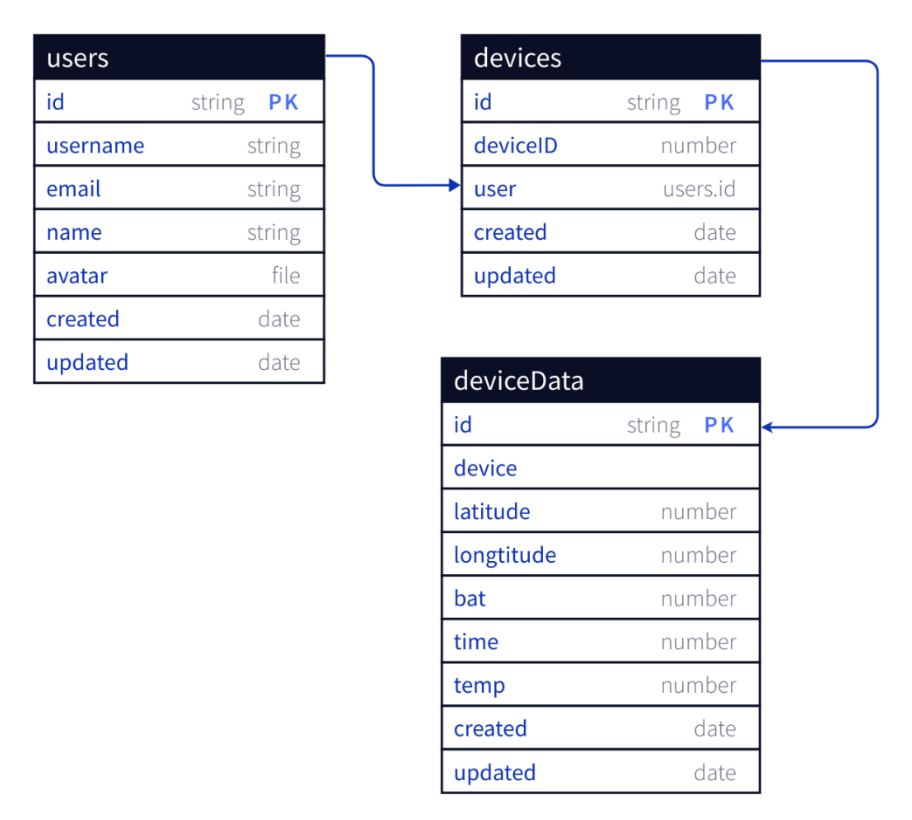
Obrázek 46-UI s mapou a vygenerovanou trasou

4.2 PocketBase

Důvod proč tuto službu používáme je ten, že automaticky generuje REST API z námi nastavených schémat SQL databáze, navíc se zde dají nastavit také pravidla pro API což nám ulehčilo mnoho práce s kódování backendu, navíc má také SDK pro Javascript, čehož využíváme.

Příklad API pravidla pro čtení z tabulky: *@request.auth.id = user.id*
Tímto pravidlem jsme nastavili, že se načtou jen data která vytvořil uživatele, který je momentálně přihlášen v koncové aplikaci

Schéma SQL databáze:



Obrázek 47-schéma SQL databáze

4.3 Implementace protokolu

I přes to, že veškerá komunikace s databází je zprostředkována skrze API vygenerovanou PocketBase, muselbýt vytvořen http endpoint na který operátor posílá data z našich koncových zařízení. Zprávu ze zařízení operátor na svém serveru vloží do JSON formátu, který metodou POST přepošle. Na našem endpointu se data rozkódují a uloží do databáze. Zdokumentovaný endpoint se nachází na Githubu: https://github.com/libormiller/scout-web/tree/master/src/routes/api/dataFromDevices

5. Závěr

Z pohledu hardwaru jsme úspěšně celý prototyp vyrobili, osadili a následně dokázali opravit menší chyby, které zde nastaly. Veškerá komunikace mezi komponenty na desce probíhala úspěšně.
Komunikace mezi GPS modulem a mikrokontrolerem byla úspěšná. Po fixu modulu k GPS síti jsme byli schopni vyčíst data o poloze, relativní rychlosti a času, které obsahuje NMEA protokol, přes který GPS modul komunikuje. Čas fixu modulu na GPS síť záležel na oblasti, kde se zařízení právě nacházelo. Pro připojení bylo optimální mít zařízení s výhledem na širé nebe, kde nejsou žádné překážky. Nejlépe se zařízení připojovalo v širé přírodě, nejhůře v zastavěné oblasti. Jakmile se připojilo, byla komunikace bezproblémová. Doba fixu na GPS síť v průměru byla v rozmezí 2-5 minut od zapnutí. První fix trval déle, a to přibližně 10 minut. Bylo to způsobeno kvůli stažení veškerých dat o poloze GPS satelitů, které modul vyžaduje k provozu.
Připojení na NB-IoT proběhlo v pořádku. Dokázali jsme úspěšně odeslat data přes UART pro nastavení modulu a následně je přijali na vzdáleném webovém rozhraní, kde jsme byli schopni zprávu vyčíst. Modul se připojil k síti téměř okamžitě, bez větších problémů. Na rozdíl od GPS byl schopen se připojit i v budově bez výhledu na širé nebe.
Další rozvoj projektu bude ležet v hardwarových přestavbách. Cílíme na co nejmenší velikost zařízení i s pouzdrem, což je ale limitováno baterií. Proto hodláme využít moduly, které budou mít více funkcí v jednom.

6. Reference

1. **Texas Instruments.** ti.com. *TPS6302x High Efficiency Single Inductor Buck-boost Converter with 4-A Switches.* [Online] 2019. https://www.ti.com/lit/ds/symlink/tps63021.pdf?ts=1678118783382.

2. **Microchip Technology Inc.** microchip.com. *PIC32CM LE00/LS00/LS60.* [Online] 2022. https://ww1.microchip.com/downloads/aemDocuments/documents/MCU32/ProductDocuments/DataSheets/PIC32CM-LE00-LS00-LS60-Family-Data-Sheet-DS60001615.pdf.

3. **Arora, Rajan.** ti.com. *I2C Bus Pullup Resistor Calculation.* [Online] 2015. file:///C:/Users/Libor/Downloads/I2C%20Bus%20Pullup%20Resistor%20Calculation%20(1).pdf.

4. **Texas Instrumens.** ti.com. *High Speed PCB Layout Techniques.* [Online] https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/14/High-speed-layout-techniques-slyp173-1_5F00_18_5F00_16.pdf.

5. **Zaucha, David.** I2C Noise Glitch Filtering. *ti.com.* [Online] 2006. https://www.ti.com/lit/an/slea053/slea053.pdf?ts=1678126497724&ref_url=https%253A%252F%252Fwww.google.com%252F.

6. **Quectel.** https://www.quectel.com. *GNSS L76-L.* [Online] https://www.quectel.com/product/gnss-l76-l.

7. **Molex.** 2042860001. *mouser.com.* [Online] https://cz.mouser.com/datasheet/2/276/1/2042860001_ANTENNAS-1374318.pdf.

8. **molex.** APPLICATION SPECIFICATION. *molex.com.* [Online] https://www.molex.com/pdm_docs/as/2042860001-AS.pdf.

9. **Murata.** mouser.com. *reference specification DFE201610E-1100887.* [Online] https://cz.mouser.com/datasheet/2/281/reference_specification_DFE201610E-1100887.pdf.

10. **TDK.** product.tdk.com. *TFM201610ALMA.* [Online] https://product.tdk.com/system/files/dam/doc/product/inductor/inductor/smd/catalog/inductor_automotive_power_tfm201610alma_en.pdf.

11. **Microchip Technology Inc.** microchip.com. *MPLAB Snap In-Circuit Debugger User's Guide.* [Online] 2020. https://ww1.microchip.com/downloads/en/DeviceDoc/50002787C.pdf.

12. **BOSCH.** bosch-sensortech.com. *BMI270.* [Online] 2021. https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi270-ds000.pdf.

13. **Maxim Integrated.** analog.com. *MAX17260.* [Online] https://www.analog.com/media/en/technical-documentation/data-sheets/MAX17260.pdf.

7. Seznam obrázků

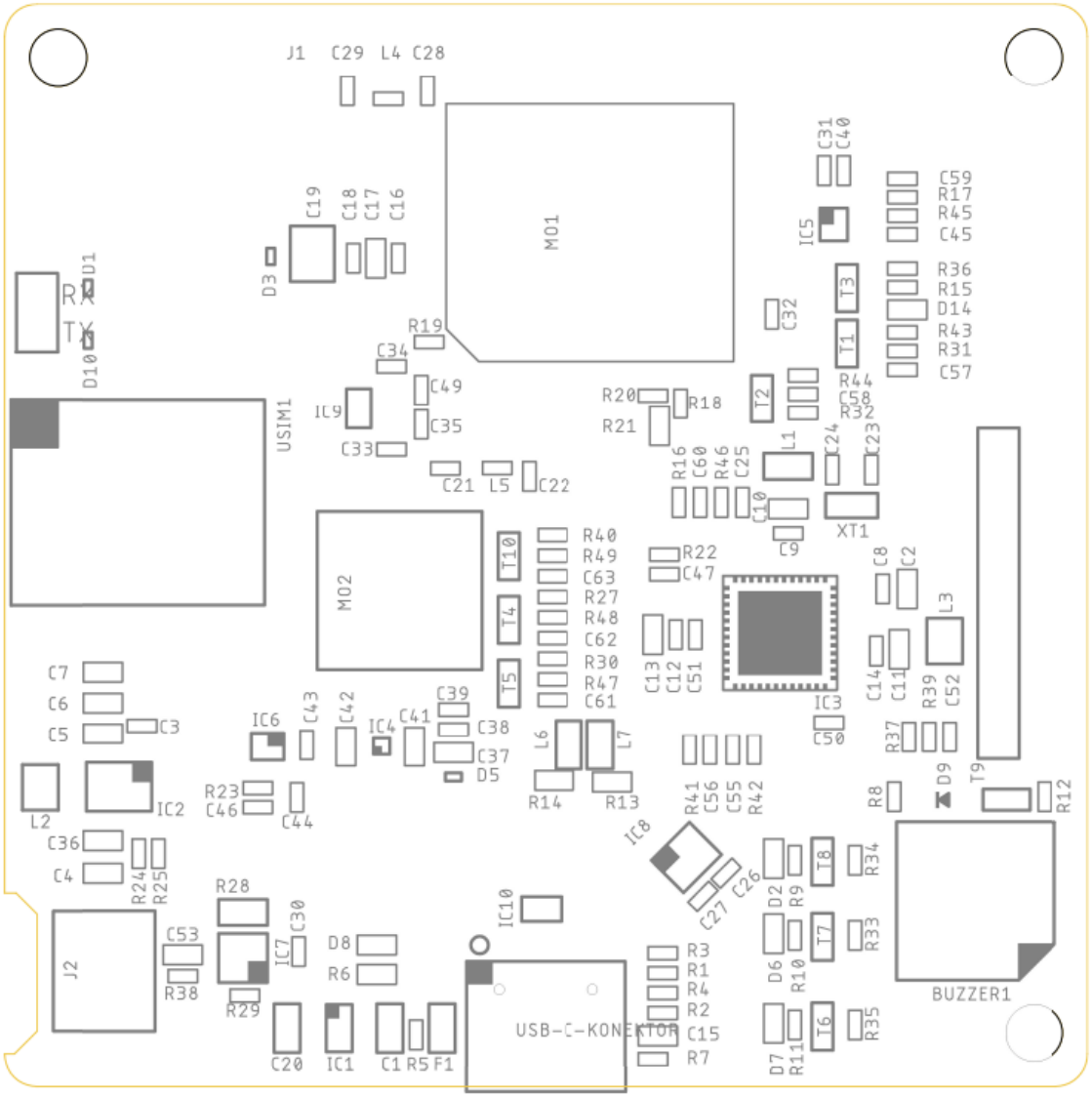
8. Přílohy

8.1 BOM List

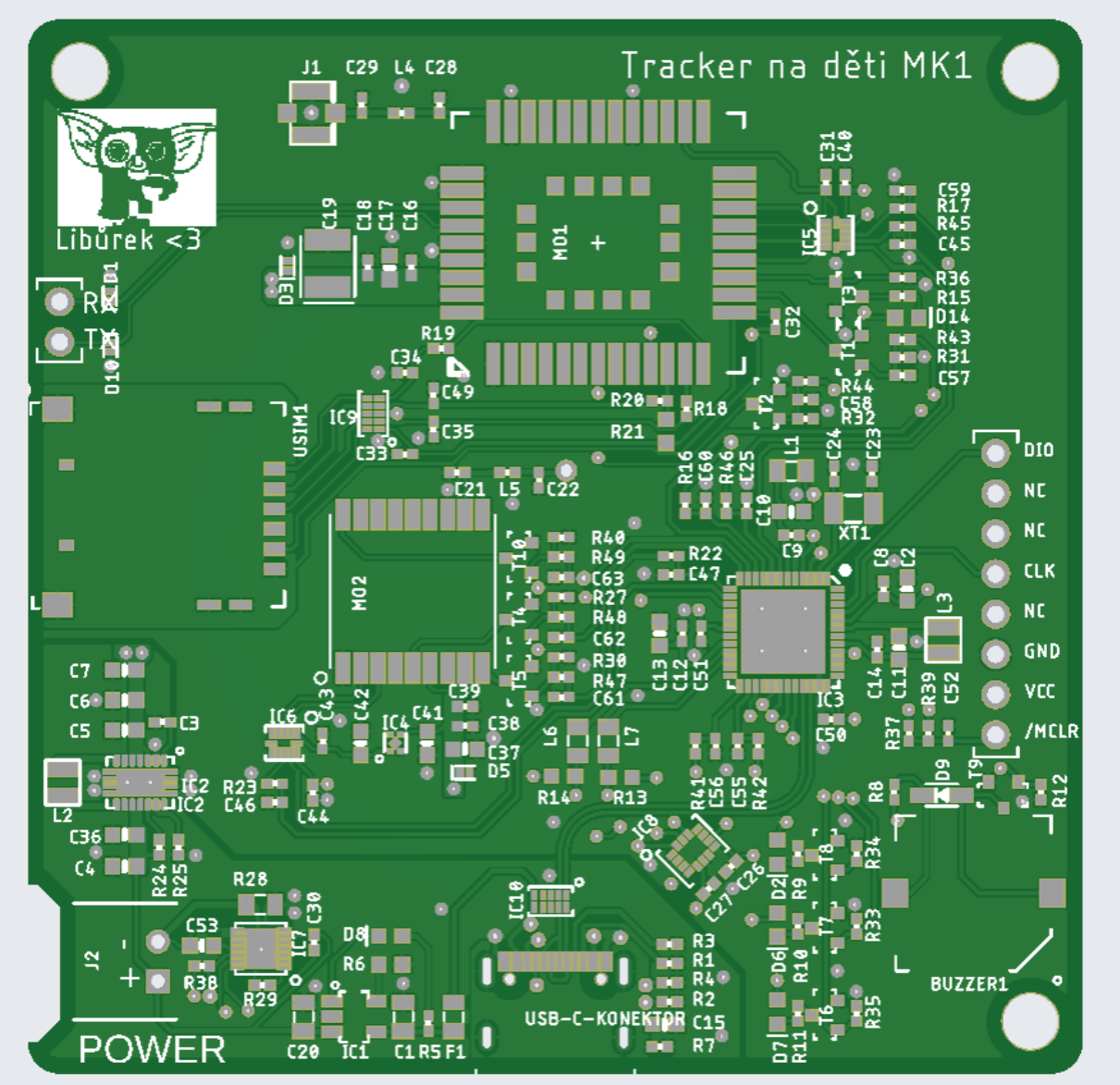
Mn.	Zařízení	Pouzdro	Označení ve schématu	Označení výrobce
1	Dioda spínací	SOD-323	D9	BAS16HT1G
1	0R01	805	R28	KRL1220E-M-R010-F-T5
1	1 uH	806	L2	TFM201610ALMA1R0MTAA
1	10 uH	806	L3	DFE201610E-100M=P2
1	100 pF	603	C17	CL10C101JB81PNC
1	100 uF	1210	C19	TPSB107K006R0400
1	10k NTC	402	R29	NCU15XH103F6SRC
1	2k	603	R6	RC0603JR-072KL
1	Krystal 32.768 kHz	CRYSTAL-SMD-3.2X1.5MM	XT1	FX135A-327
1	4.7 nF	603	C15	06035C472J4T2A
1	470 nF	603	C53	GCM188R71E474KA64D
1	ANTENA GSM patch		ANT1	204286-0001
1	Ferit 600R@100MHz	805	L1	BLM21SP601SN1D
1	BMI270	LGA-14	IC8	BMI270
1	Piezo měnič		BUZZER1	CMI-9705-0380-SMT-TR
1	Konektor 1X8		J3	10129378-908001BLF
1	Konektor 1x2		J4	10129378-902004BLF
1	LED červená	603	D6	LTST-C193KRKT-5A
1	LED žlutá	603	D7	LTST-C193KSKT-5A
1	MAX17260	TDFN-14	IC7	MAX17260
1	MCP73831	SOT-23-5	IC1	MCP73831T-4ADI/OT
1	Pojistka 750mA	805	F1	MF-PSMF075X-2
1	PIC32CM5164LE00048	QFN-48	IC3	PIC32CM5164LE00048
1	QUECTEL-BC66-NA		MO1	BC66NADA
1	QUECTEL-L76L		MO2	L76L-M33
1	Konektor S2B-XH-A-1		J2	S2B-XH-A-1
1	TPS63021	VSO-14	IC2	TPS63021DSJT
1	TPS7A0328DQNR	DQN0004A	IC4	TPS7A0328DQNR
1	Konektor U.FL		J1	CONUFL001-SMD-T
1	Konektor USB-C		USB-C	USB4105-GF-A
1	Konektor uSIM		USIM1	78646-3001
2	0R	603	L4, L5	CRCW06030000Z0EAC
2	4.7 uF	805	C1, C20	CL21A475KAQNNNE
2	4k7	603	R13, R14	RC0603FR-134K7L
2	5k1	402	R1, R2	CRCW04025K10FKEDC
2	Ferit 1k@100MHz	805	L6, L7	74279205
2	Pole ESD diod	DFN2510A-10	IC9, IC10	PESD4USB5B-TBRX
2	TXB0104QRUTRQ1	UQFN-N12	IC5, IC6	TXB0104QRUTRQ1

3	18 pF	402	C16, C23, C24	C1005C0G1H180J050BA
3	22 uF	603	C5, C6, C7	GRM188C80G226MEA0D
3	LED zelená	603	D2, D8, D14	LTST-C193KGKT-5A
4	1 uF	603	C11, C13, C41, C42	EMK107B7105KA-T
4	10k	603	R3, R4, R21, R38	RT0603FRE0710KL
4	200R	402	R9, R10, R11, R15	CRCW0402200RFKEDC
4	22R	402	R8, R18, R19, R20	RC0402JR-0722RL
4	240 pF	402	C25, C45, C46, C47	GRM1555C1H241JA01D
4	ESD dioda	DFN1006-2	D1, D3, D5, D10	PESD7V0L1BSLAZ
5	10 uF	603	C2, C4, C10, C36, C37	CL10A106MQ8NNNC
5	33 pF	402	C33, C34, C35, C39, C49	C0402C330K3HACTU
9	18 nF	402	C55, C56, C57, C58, C59, C60, C61, C62, C63	GCM155R71E183KA55D
10	Mosfet-N	SOT-23-3	T1, T2, T3, T4, T5, T6, T7, T8, T9, T10	T2N7002AK,LM
14	100k	402	R7, R12, R24, R25, R27, R30, R31, R32, R33, R34, R35, R36, R39, R40	CR0402-FX-1003GLF
15	1k	402	R5, R16, R17, R22, R23, R26, R41, R42, R43, R44, R45, R46, R47, R48, R49	RC0402JR-071KL
18	100 nF	603	C3, C8, C9, C12, C14, C18, C26, C27, C30, C31, C32, C38, C40, C43, C44, C50, C51, C52	C0603C104K5RAC3121

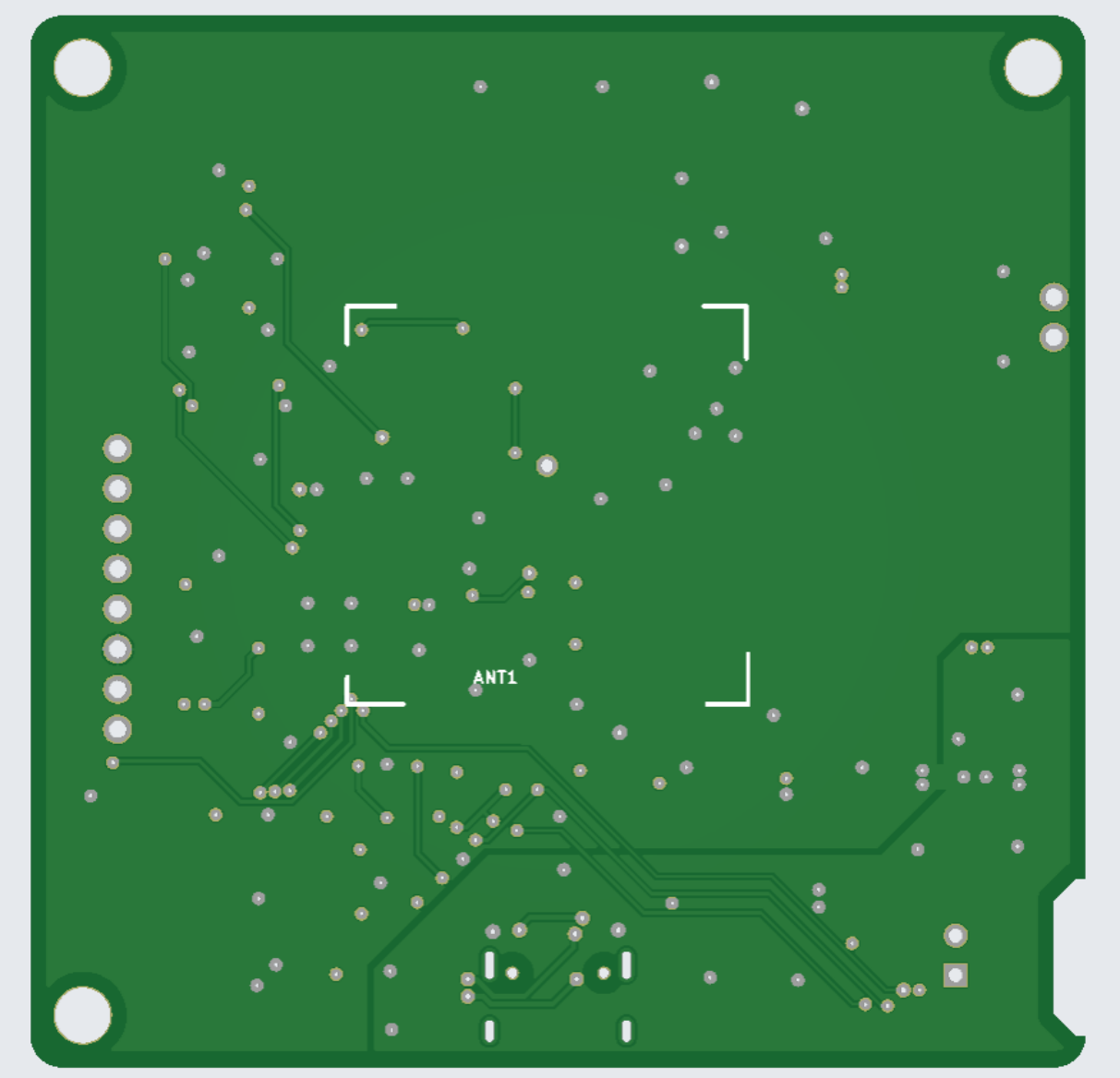
8.2 Rozložení součástek na PCB



8.3 Gerber data – horní srana



8.4 Gerber data – spodní strana



8.5 Fotka vyrobeného zařízení

8.6 Nákres pouzdra pro PCB určeného pro vývoj