Fakulta elektrotechnická

Katedra telekomunikační techniky

Bakalářská práce

Využití (veřejných) IoT sítí pro dohled starších samostatně žijících osob

Marek Kubát

Studijní program:

Elektronika a komunikace

Vedoucí práce:

Ing. Petr Novák, Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kubát** Jméno: **Marek** Osobní číslo: **491907**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Využití (veřejných) IoT sítí pro dohled starších samostatně žijících osob

Název bakalářské práce anglicky:

Use of (public) IoT Networks for the Supervision of Older Single People

Pokyny pro vypracování:

Stále přibývá starších samostatně žijících lidí. Protože je jejich bezpečnosti velmi důležitá, nezbývá než na ně občas aspoň nějak vzdáleně dohlédnout. Dnes je tohoto dosaženo nejčastěji pomocí běžného internetu (např. IP-kamery, …). Se zvětšujícím se pokrytím různých IoT technologií lze uvažovat i o tomto typu přístupu, protože použitých senzorů a přenášených dat nebývá mnoho.
A) Zmapujte u nás veřejně dostupné IoT technologie / sítě (LoRa, SigFox, …) vhodné pro zmíněný účel (dostupnost, jednoduchost, podpora, výrobci, vlastní koncové zařízení, cena přístupu, …).
B) Prostudujte možnost tvorby vlastního domácího / experimentálního zařízení (se senzory) připojitelného přímo do takovéto již existující (veřejné) sítě.
C) Navrhněte a vytvořte jednoduchou verzi (vlastního) koncového zařízení připojitelného do již existující IoT sítě (sítí) sloužící jako domácí tzv. gateway. Tedy umožňující současný sběr hodnot i z několika lokálních jednoduchých / experimentálních senzorů pro sdružení jejich výstupu za účelem minimalizace přenášených dat a tím snížení nákladů na připojení do cílové IoT sítě.

Seznam doporučené literatury:

[1] Price Mark, C# 8.0 and .NET Core 3.0 - Modern Cross-Platform Development, Packt, 2019, ISBN 978-1788478120
[2] MacDonald Matthew, Pro WPF 4.5 in C#, APress, 2012, ISBN 978-1-4302-4366-3
[3] Noviello Carmine, Mastering STM32, LeanPub, 2017

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Novák, Ph.D. oddělení kognitivních systémů a neurovědy CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.01.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

Ing. Petr Novák, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Prohlášení:

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Praha 19. května 2022 Marek Kubát

Poděkování:

Rád bych poděkoval svému vedoucímu práce, panu Ing. Petru Novákovi, Ph.D, za velmi trpělivý, ochotný přístup a poskytnutí všech dostupných pomocných materiálů.

Marek Kubát

Abstrakt

Tato práce je zaměřena nejprve na přehled veřejně dostupných IoT sítí v České republice. Následně je realizován experimentální modul určený primárně pro dohled nad samostatně žijícími staršími lidmi a byl by do takovéto sítě případně připojitelný. K modulu jsou připojeny vybrané experimentální senzory přes nejčastěji používané sběrnice. Celý projekt byl vytvořen ve vývojovém prostředí STM32CubeIDE za použití jazyka C. Výsledná data jsou zobrazována v demonstrační aplikaci napsané v .NET Core / C#.

Klíčová slova:

IoT, Internet věcí, modul, komunikační kanál, senzor

Abstract

This work is mainly focused on the publicly accessible loT networks in the Czech Republic. Furthermore, it is an implemented experimental module primarily for the supervision of elderly people who live alone and could be connects to such a network. Selected experimental sensors are connected via the most frequently used buses. This whole project was created in the STM32CubeID environmental development using language C. The resulting data is displayed in a demo application written in .NET Core / C#.

Keywords:

IoT, Internet of Things, module, communication channel, sensor

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek kódů

Seznam zkratek

ADC Analog Digital Converter

ARM Advanced RISC Machine

FSK Frequency-shif keying

HAL Hardware Abstraction Layer

I2C Inter-Integrated Circuit

IoT Internet of Things

LAN Local Area Network

LPWAN Low Power Wide Area Network

LTE Long Term Evolution

PIR Passive Infrared

SCL Serial Clock

SDA Serial Data

SPI Serial Peripheral Interface

TCP/IP Transmission Control Protocol / Internet Protocol

TTL Transistor-Transistor-Logic

USART Universal Synchronous / Asynchronous Receiver and Transmitter

USB Universal Serial Bus

1 Úvod

Pojem *Internet věcí* (*Internet of Things, IoT*) se v dnešní době skloňuje a používá stále častěji. Internet věcí poprvé definoval britský inženýr Kevin Ashton v roce 1999, který se proslavil technologií pro identifikaci zboží (čárové kódy). Ovšem s myšlenkou o I=*Internetu věcí* přišla již v 60. letech 20. století společnost Coca-Cola. Ta zprovoznila automat umožňující samostatně monitorovat stav chlazení umístěných nápojů a rovněž informovat o jejich počtu. Takových pokusů by se však v historii našlo samozřejmě mnohem více.

Nicméně, dalším významným rokem byl rok 2008, kdy se ve Švýcarsku uskutečnila konference firmy Cisco. Tam zaznělo, že počet zařízení připojených do Internetu přesáhl počet obyvatel žijících na celém světě. Velký zlom nastal ovšem v roce 2011, kdy příchod protokolu IPv6 zajistil dostatek IP adres pro všechna zařízení.

V dnešní době už je Internet věcí rozšířen skutečně do mnoha odvětví. Počínaje průmyslem, přes zdravotnictví, finančnictví až po rodičovství a samozřejmě i domácí dohledy [1].

2 Cíle práce

Bakalářská práce je zaměřena na tvorbu experimentálního modulu se základními senzory nejčastěji používanými pro dohled nad samostatně žijícími staršími lidmi. Jsou tedy demonstračně použity různé komunikační kanály, jejich protokoly a na ně připojené koncové senzory. Tato práce byla brána jako vzor dle aktuálních potřeb projektů řešených vedoucím práce.

Jednotlivé cíle práce, podle zadání, jsou následující:

- A, Zmapujte u nás veřejně dostupné IoT technologie / sítě (LoRa, SigFox, ...) vhodné pro zmíněný účel (dostupnost, jednoduchost, podpora, výrobci, vlastní koncové zařízení, cena přístupu, ...).

  - V kapitole IoT sítě budou zmapovány a vybrány nejdostupnější veřejné IoT sítě. Jedná se zde o sítě Sigfox, NB-IoT a LoRa. U každé sítě jsou uvedeny jejich vlastnosti a samozřejmě jejich výhody a nevýhody.

- B, Prostudujte možnost tvorby vlastního domácího / experimentálního zařízení (se senzory) připojitelného přímo do takovéto již existující (veřejné) sítě.

  - Tato problematika je zpracována v další části práce, kde jsou popsány technické specifikace vybraných senzorů a komunikací, schéma zapojení a popis zvoleného vývojového prostředí.

- C, Navrhněte a vytvořte jednoduchou verzi (vlastního) koncového zařízení připojitelného do již existující IoT sítě (sítí) sloužící jako domácí tzv. gateway. Tedy umožňující současný sběr hodnot i z několika lokálních jednoduchých / experimentálních senzorů pro sdružení jejich výstupu za účelem minimalizace přenášených dat a tím snížení nákladů na připojení do cílové IoT sítě.

  - V kapitole Základní návrh, je uveden použitý typ hardwaru a ukázky části kódů, které vedou na správnou konfiguraci celého zařízení.

Tato bakalářská práce se nevěnuje následujícím tématům:

- Zabezpečení přenosu dat
- Spolehlivosti přenosu
- Celkové dokumentaci pro výrobu zařízení

Výstupem této práce je čistě experimentální zařízení pro ověření možnosti sběru dat z několika různých senzorů, připojitelných pomocí různých sběrnic. Nejedná se tedy o celkové a zcela hotové zařízení a kompletní dokumentaci.

3 Základní pojmy

Pokusme si během následujících řádků vyjasnit pojmy k tomuto tématu související a pochopit, co *Internet věcí* znamená, nebo jak je definován. Jedním z důležitých pojmů je *Síť*. V mnoha případech je to bráno jako představa Internetu, tzn. celosvětový systém propojených počítačových sítí pracujících nejčastěji na protokolu *TCP/IP*. Dá se ale vytvořit i *LAN* (lokální domácí síť), ve které mohou různé objekty (většinou nějaké malé počítače) spolu přímo komunikovat a posílat další zprávy do *Internetu*. Tím se dostáváme k dalšímu důležitému pojmu, kterým je *věc* resp. *objekt*. V případě *IoT* se za objekty nejčastěji považují senzory mající vlastní elektroniku a software s možností snímaní různých fyzikálních veličin. Tyto informace pak následně sdílí s ostatními objekty v síti, anebo je pouze přeposílají dále pro další zpracování, nebo vyhodnocení. Nicméně za další objekty mohou být už brány i složitější stroje, vozidla, domácí spotřebiče apd.

Je tedy patrné, že v případě *Internetu věcí* nepůjde pouze o samotné objekty, ale i o data, tedy informace, které si jsou schopny i mezi sebou přeposlat. Nyní je možné dát dohromady definici, která dle [2] zní:

Definice 1. *Internet věcí představuje systém, ve kterém si objekty vyměňují data přes síť Internet. Objekty mohou mít libovolnou topologii a uspořádání za účelem dosažení vyšších cílů.*

Nezbytnou součástí *Internetu věcí* a *senzorů* je *mikrokontrolér*. *Mikrokontroléry* jsou charakteristické tím, že mají program i data uložená ve vlastní paměti, integrované (většinou) v jednom pouzdru spolu s potřebnými I/O periferiemi. Existuje spousta rodin mikrokontrolérů dostupné se šířkou sběrnic v rozsahu 8 bitů až 32 bitů. Jednotlivé rodiny se mohou velmi lišit v počtu a kombinací různých interních periferií, velikostí pamětí, atd. Mezi největší světové výrobce mikrokontrolérů dnes patří například Microchip, Atmel (dnes již součást Microchip), Analog Devices, Freescale (dříve Motorola, dnes již NXP), NXP (Philips), ST Microelectronics, Texas Instruments a mnoho dalších.

Další definice nejvíce používaných pojmů nezbytné pro práci:

- Senzor - součástka, která snímá libovolnou fyzikální veličinu a je připojena na lokální komunikaci většinou mikroprocesoru.
- Lokální sběrnice - způsob komunikace mezi senzorem a mikroprocesorem obsaženým z modulu, toto většinou na velmi krátkou vzdálenost (I2C, SPI, viz kapitola Způsoby přenosu dat)
- Modul - celkové zařízení obsahující senzory a je připojené na externí komunikační kanál
- Externí komunikace – komunikace z modulu na větší vzdálenost do (většinou výkonného) koncového zařízení (USART, LAN, LORA)

4 IoT sítě

Následující kapitola se bude zabývat dostupnými *IoT sítěmi*. Bude kladen důraz na výhody, nevýhody a vhodnost použití právě pro domácí dohled.

4.1 Sigfox

První z rozšířených a dostupných IoT sítí je od francouzské firmy SigFox[1]. Tato síť slouží obvykle pro jednosměrnou komunikaci. To lze chápat takto, pokud bychom obdrželi od senzoru přes síť Sigfox informaci o požáru v domácnosti (tedy směrem ven), tak již nejsme schopni přes síť Sigfox spustit hlášení / akci v domácnosti (tedy směrem dovnitř).

Celá technologie spadá do takzvané *LPWAN* kategorie. Sigfox využívá bezlicenční pásmo 868 MHz pro Evropu a může poslat maximálně 140 zpráv o délce 12 bytů a přijmout pouze 4 zprávy o délce 8 bytů za den. V době přenosu však zpráva zabírá pouze 100 Hz z celkového pásma. To snižuje přenosovou rychlost. Tím, že se kombinuje nízká přenosová rychlost a malý objem dat, se velmi snižuje spotřeba energie daného zařízení. Díky tomu zařízení vydrží několik let, což je velká výhoda.

Sigfox rovněž velmi šikovně řeší spolehlivost a zabezpečení přenosu. Jsou odesílány vždy tři kopie té samé zprávy na třech různých frekvencích zvolené z pseudonáhodné sekvence, a proto v podstatě nemůže dojít ke ztrátě zprávy.

Dosah signálu u přímé viditelnosti mezi anténami může činit až 200 km. V krajině 30-50 km a v zastavěné oblasti 3-10 km.

Tato síť má i své nevýhody. První už byla popsána výše, a to je problém s jednosměrnou komunikací. Další nevýhoda pramení z nízké přenosové rychlosti, kdy je téměř nereálné přenášet fotky, či videa např. z bezpečnostních kamer. Poslední nevýhodou je zpoplatnění služby operátorem.

Vzhledem k výše popsaným výhodám a nevýhodám se tato síť jeví jako nevhodná pro účely domácího hlídače a to zejména kvůli omezení kapacity přenosu signálu.

Zde je na obrázku 1 ukázka cen a tarifů [3].

[1] https://sigfox.cz/cs

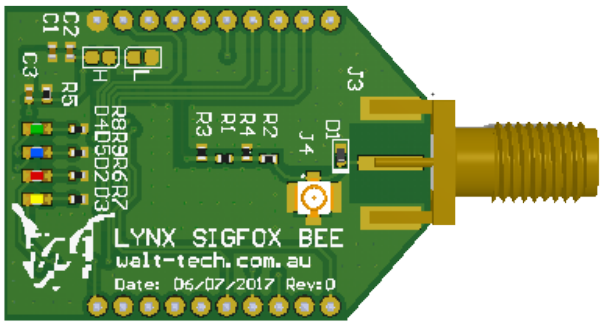
Základní ceník konektivity

Do 1000 zařízení

BASIC	PLUS	ULTRA
140 Kč / zařízení / rok	**215 Kč / zařízení / rok**	**247 Kč / zařízení / rok**
2 uplinky denně	70 uplinků denně	140 uplinků denně
1 downlink týdně	2 downlinky denně	4 downlinky denně

Obrázek 1: Ceník Sigfox

Na obrázku 2 je ukázka jednoho z mnoha modulů. Jedná se o modul LYNX BEE Sigfox Module RCZ1 [4], který je možné připojit k další rozšiřující desce. Cena takového modulu se pohybuje okolo $30.

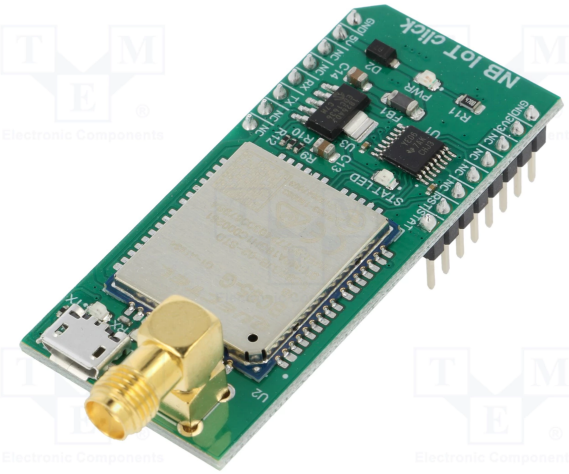


Obrázek 2: Samostatný modul pro tvorbu vlastního zařízení připojitelného do sítě Sigfox

4.2 NB-IoT

Narrow Band IoT je další ze sítí spadající do třídy *LPWAN*. Na rozdíl proti Sigfox poskytuje schopnost obousměrné a častější komunikace. Tato technologie využívá úzké pásmo *LTE*. Stačí tedy pouze upravit software na nynějších vysílačích. Díky tomu má téměř 100% pokrytí. Tato technologie spadá pod globálního operátora a služba je tedy vždy zpoplatněna. Což lze považovat asi za největší nevýhodu.

Jedním z možných modulů pro připojení je deska NB IOT CLICK [5], která je na obrázku 3.



Obrázek 3: Samostatný modul pro tvorbu vlastního zařízení připojitelného do sítě Narrow Band IoT

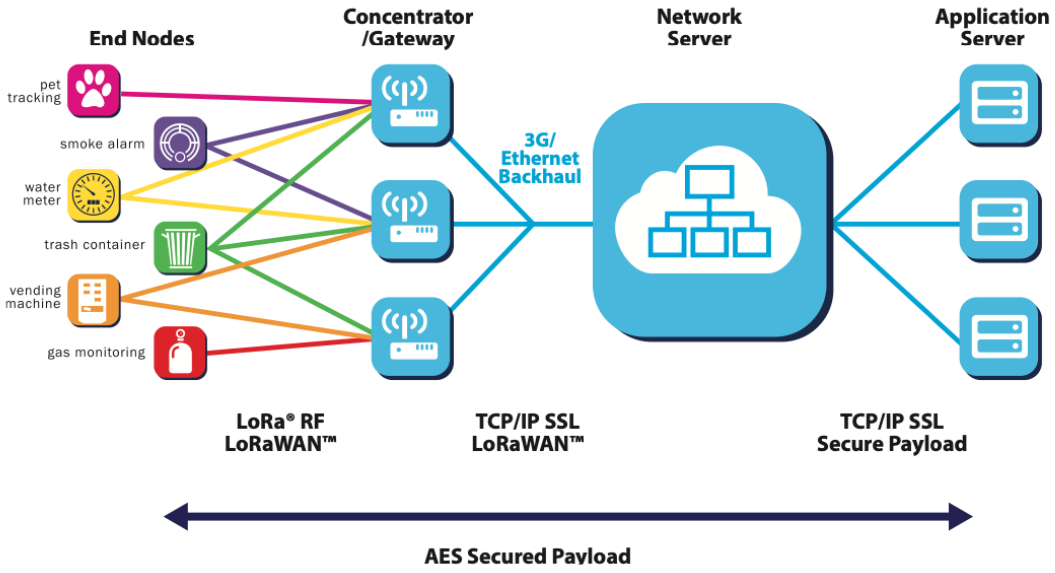
4.3 LoRa

Long Range[2] [6] je síť s ultra nízkými požadavky na spotřebu energie s obousměrnou komunikaci. Jedná se o standard fyzické vrstvy pro rádiovou komunikaci využívající jako Sigfox rovněž nelicencované pásmo 868 MHz a přenosová rychlost se pohybuje od 300 bit/s do 50 000 bit/s. Využívá modulaci *FSK*, díky které je vytvořen velký dosah. LoRa, narozdíl od Sigfoxu nebo NB-IoT vznikla spoluprací více subjektů, což nabízí výhodu, že tato technologie umí využívat zařízení od různých výrobců.

Jak je vidět na obr. 4, tak architektura je uspořádána do tzv. mesh a každý senzor je připojen na gateway (výchozí bránu). Odtud jsou příchozí pakety přeposílány do cloudového serveru.

IoT síť LoRa na českém území provozuje společnost České Radiokomunikace[3]. Cena za připojení je individuální a závislá na mnoha faktorech jako jsou počet připojených zařízení, počet odchozích a příchozích zpráv za den. Nicméně existuje základní tarifní balíček Pilot, jehož cena je 200 Kč/měsíc, viz obrázek 5. Ovšem lze také použít celosvětově dostupnou platformu The Things Network[4], odkud je možné připojit se k jíž existujícím gateway.

[2] https://lora-alliance.org
[3] https://www.cra.cz
[4] https://www.thethingsnetwork.org



Obrázek 4: Princip a struktura přenosu dat v síti LoRa

Obrázek 5: Tarif Pilot pro síť LoRa

4.4 Třídy LoRa

Zařízení pro LoRa technologie je možné rozdělit do následujících tříd:

4.4.1 Třída A

- Nejvíce úsporná zařízení
- podpora obousměrné komunikace pro všechna zařízení
- každý uplink je následovaný dvěma rámci pro příjem dat

4.4.2 Třída B

- otevírají mimořádná přijímací okna v danou dobu

4.4.3 Třída C

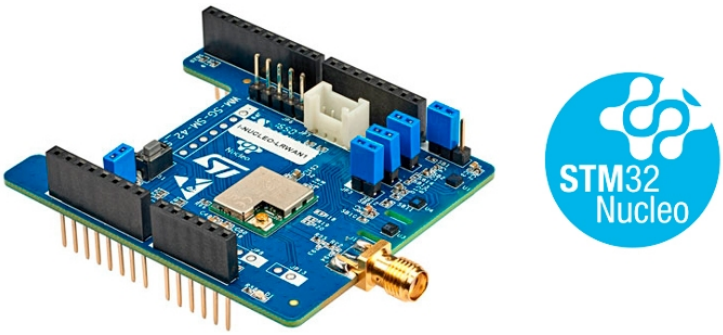
- přijímací okna jsou otevřená téměř nepřetržitě a zavírají se pouze při vysílání

Pro shrnutí jsou v tabulce 1 uvedeny technické specifikace LoRa.

Tabulka 1: Specifikace LoRa

technologie	Spread Spectrum
modulace	FSK
počet kanálů	10
velikost zprávy	256 bytů
rychlost přenosu	250 bps - 50 kbps
přenosové pásmo UP	125/250 kHZ
přenosové pásmo DOWN	868 MHz
počet zpráv za den	neomezený
vysílací výkon	25 mW / 14 dBm
citlivost	-140 dBm
zisk	165 dBm
výdrž baterie	5-15 let

Se zařízením LoRa se budeme zabývat i v další části této práce. Zde je ukázka na obrázku 6 právě rozšiřujícího modulu LoRa [7] k vývojové desce typu STM32-Nucleo. Vytvářený modul sbírá data ze senzorů a data jsou přenášena přes zvolené komunikační rozhraní do počítače pro jejich další zpracování.



Obrázek 6: Vývojový LoRa modul vhodný například pro desky STM32-NUCLEO

5 Základní návrh

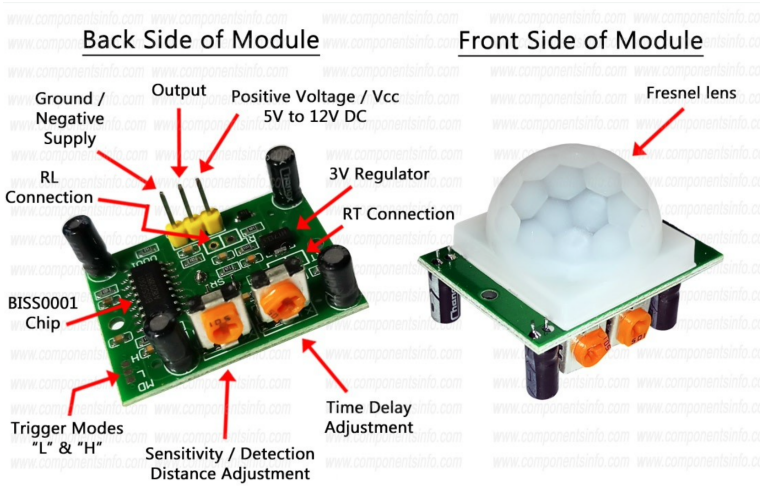
Následující kapitola popisuje základní návrh celého vytvářeného senzorového modulu, který byl inspirován [8], [9]. Základním návrhem je myšleno sběr dat ze senzorů a přes lokální sběrnice (I2C, SPI, ...) jejich poslání do centrály modulu, v našem případě mikrokontrolér L152RE. Dále odeslání dat (i částečně předzpracovaných) přes externí komunikaci USART / LAN / LORA do (většinou vzdáleného) počítače a zobrazení aktuálních hodnot v demonstrační aplikaci.

5.1 Použité senzory

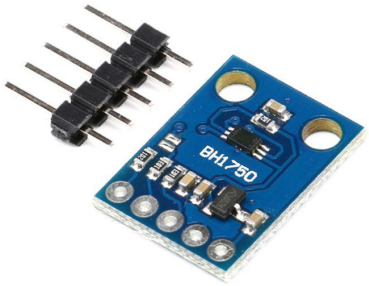
Níže budou popsány jednotlivé senzory, které byly vybrány jako vzorové pro tuto práci. Jedná se tedy o typické příklady, které by u realizace domácího hlídače nad staršími osobami mohly být nejčastěji zastoupeny. Vzorové senzory byly rovněž vybrány tak, aby zastupovali nejčastěji používané lokální sběrnice pro přenos dat ze senzorů do mikropočítače.

5.1.1 HC-SR501 - Senzor pohybu PIR

Prvním použitým senzorem, viz obr. 7, je *PIR* čidlo pro nejzákladnější detekci pohybu [10]. Výstup z tohoto čidla je digitální (ano/ne). Pokud čidlo zaznamená pohyb, signalizuje toto pomocí 3,3 V logiky, tedy logickou jedničkou a pokud žádný pohyb nezaznamená, je na výstupu logická nula (0V). Snímací rozsah je až 110° a 7 metrů.



Obrázek 7: Senzor HC-SR501 založený na principu PIR



Obrázek 8: Senzor BH1750 detekující intenzitu osvětlení

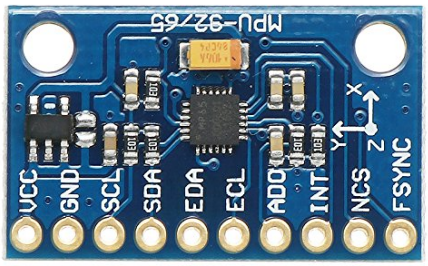
Obrázek 9: Senzor plamene pomocí IR

5.1.2 BH1750 - Senzor detekující intenzitu osvětlení

Tento senzor slouží k detekci intenzity osvětlení [11]. Senzor používá sběrnici *I2C*. Rozsah hodnot je 1-65535 lx. Napájecí napětí by nemělo přesáhnout 4,5 V.

5.1.3 Infračervený senzor plamene

Dalším důležitým senzorem, který by při dohledu nad samostatně žijícími lidmi neměl chybět je detektor plamene [12]. Senzor detekuje světlo o vlnové délce 760 nm. Detekční vzdálenost může být až 80 cm a snímací úhel až 60°. Výstupní hodnoty se dají vyčíst buďto přes digitální pin formou logické 1 a 0 (nastavena prahová úroveň), nebo přes analogový pin (přímý analogový výstup senzoru).



Obrázek 10: Senzor typu MPU obsahující akcelerometr, gyroskop a magnetometr

5.1.4 MPU9255 - Senzor pohybu

Posledním použitým senzorem je tří-osý akcelerometr [13] (obsahující i gyroskop a magnetometr, ty však nejsou využity). Výstupní data lze přenášet sériově jak přes *I2C*, tak *SPI*. Napájecí napětí je v rozmezí 2,4 - 3,5 V. Ten je vhodný například pro detekci pohybu některých předmětů (postel, křeslo, ...).

5.2 Způsoby přenosu dat

Senzory a další periferie komunikující s mikrokontrolérem mohou mít buď analogové (jednodušší senzory) nebo digitální rozhraní (složitější senzory).

V případě analogového výstupu zpracováváme data prostřednictvím interního nebo externího AD převodníku. V případě, že se jedná o digitální přenos dat, pak jsou v dnešní době nejrozšířenější tyto (lokální) sběrnice.

5.3 Lokální sběrnice pro senzory

5.3.1 SPI

Při použití této sběrnice lze data současně odesílat a přijímat. *SPI* je typu "single master", tedy na sběrnici je přítomný právě pouze jeden řídící obvod. Všechny ostatní jsou typu slave (podřízení). Pro komunikaci slouží čtyři vodiče.

- SCK - Serial Clock (hodinový signál, poskytuje master)

- SDO - Serial Data Out (datový výstup, z master do slave), MOSI
- SDA - Serial Data In (datový vstup, ze slave do masteru), MISO
- SS - Slave Select (výběr cílového obvodu typu slave)

K inicializaci SPI je nezbytné toto nastavení:

- typ MASTER - řídící režim, pin SCK nastaven jako výstup
- typ SLAVE - řízený režim, pin SCK nastaven jako vstup
- přenosová rychlost
- klidová úroveň signálů
- hrana hodinového signálu - výstup dat na sestupnou nebo vzestupnou hranu hodinového signálu
- okamžik vzorkování dat na sběrnici - uprostřed nebo na konci odesílaných dat

5.3.2 I2C

Proti *SPI* sběrnici je *I2C* typu "multimaster". Nevyužívá se zde adresace pomocí signálu chip select, ale každé připojené zařízení obsahuje svoji vlastní komunikační adresu. Pro komunikaci slouží pouze dva vodiče:

- SDA, data (obousměrná linka)
- SCL, hodiny (poskytuje právě řídící obvod / master)

5.4 Externí komunikace

5.4.1 USART

USART je sériové rozhraní, sloužící ke komunikaci mikrokontroléru s ostatními jednotkami. K dispozici jsou tři módy:

- Asynchronní (full - duplex)
- Synchronní - Master (half - duplex)

- Synchronní - Slave (half - duplex)

Hlavní parametry jsou:

- 8 nebo 9 datových bitů
- 1 nebo 2 stop bity
- podpora parity
- Fractional baud rate generator

Tento typ komunikace by byl vhodný například v případě existence několika samostatných vzájemně propojených senzorových modulů, kdy je pouze jeden senzorový modul připojen do sítě pro externí komunikaci (*LAN / LoRa*).

5.4.2 BlueTooth

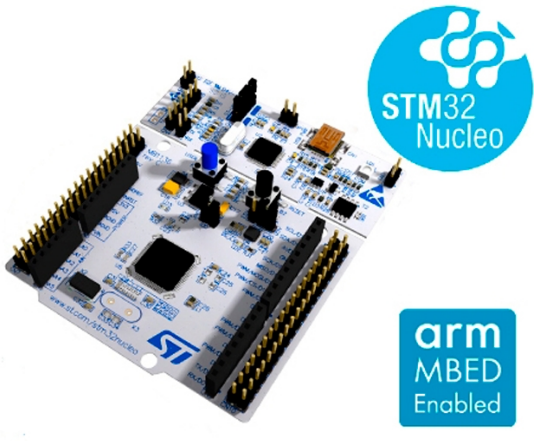
Používá se pro přenos dat mezi mobilními zařízeními nižší rychlostí, obvykle 1 – 2 Mb/s a na kratší vzdálenost (maximálně desítky metrů).

Tento typ komunikace v senzorovém modulu je využitelný zejména pro příjem informací / dat z tzv. nositelných senzorů (tep, okysličení, ...) nebo ze mobilních zařízení pro přivolání pomoci (help-tlačítka).

5.5 Použitý hardware pro vývoj modulu

Nucleo STM32-L152RE

Použitým mikrokontrolérem je STM32L152 [14] umístěný na vývojové desce nazvané STM32 Nucleo-L152RE. Tato vývojová deska je od již zmíněné společnosti ST Microelectronics. Jedná se procesor typu ARM Cortex-M3 spadající do kategorie Ultra-low-power. Procesor disponuje dvěma AD převodníky, dvěma rozhraní pro komunikaci na periferii I2C, 5x USART, 3x SPI a USB. Dále obsahuje FLASH paměť pro vlastní program o velikosti 512 kB. Rovněž je na vývojové desce k dispozici debugger a vestavěný programátor, připojitelný přes mini USB k počítači (tvorba a ladění aplikace).



Obrázek 11: Vývojová deska STM32-NUCLEO L152RE

5.6 STM32CubeIDE

Celý projekt je vytvářen v prostředí STM32CubeIDE od společnosti ST Microelectronics [5] (dostupný zdarma). Uvedený software je vhodný pro procesory typu ARM rodiny STM32 použité dále v této práci. Jako programovací jazyk jsem zvolil C [15].

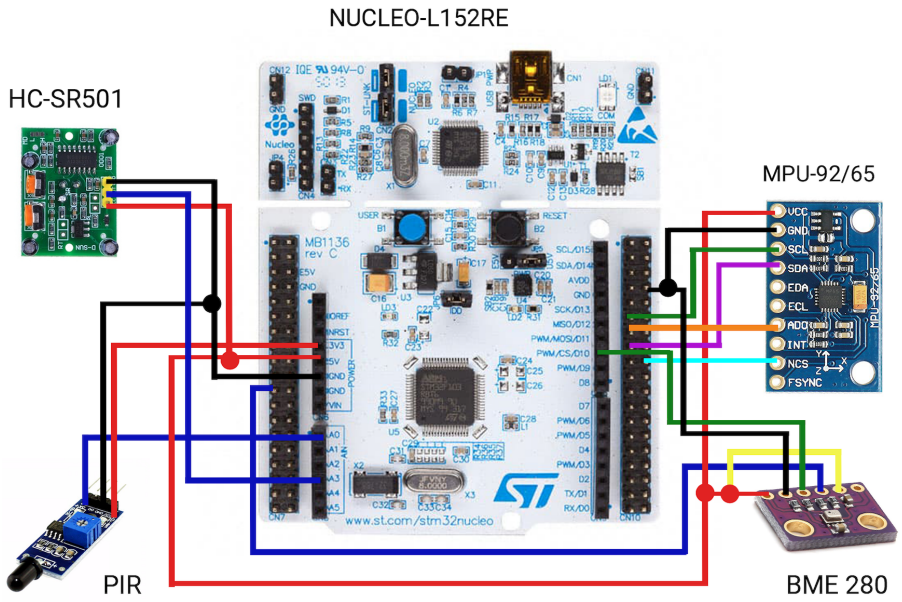
5.6.1 HAL knihovny

Při vytváření projektů pomocí STM32CubeIDE jsou využívány knihovny HAL [16] zajišťující dostatečnou univerzálnost vytvořeného programového kódu pro libovolný procesor typu ARM z rodiny STM32 od společnosti ST Microelectronics. Výhodou těchto knihoven je dokonalé vzájemné propojení mezi aplikační a hardwarovou vrstvu. Například pro inicializaci periférie používající SPI nemusíme dle datasheetu složitě konfigurovat registry procesoru, ale při tvorbě projektu (případně i v grafickém návrhu mikroprocesoru) zvolíme pouze příslušný pin a jím využívaný typ komunikace. Po vygenerování projektu je v souboru `main.c` již vytvořená inicializační funkce, a tu tedy pouze využijeme.

[5] https://www.st.com/content/st_com/en.html

5.7 Schéma zapojení senzorů v modulu

Na obrázku 12 je uvedeno připojení jednotlivých demonstračních senzorů k periferiím mikrokontroléru. Jedním z cílů této práce bylo rovněž poskytnout připojení senzorů přes různé typy nejčastěji používaných lokálních sběrnic. Proto detektor pohybu je připojen na digitální vstup a detektor plamene na analogový vstup. Senzor pro detekci osvětlení je připojen pomocí *I2C* a akcelerometr přes *SPI* sběrnici.



Obrázek 12: Propojení vývojové desky STM32-NUCLEO s vybraných typů senzorů.

6 Programové vybavení modulu

Program pro mikroprocesor je složen z několika hlavních částí:

- A) Zdrojové soubory pro senzory a lokální sběrnice
- B) Definice / vyplnění struktur pro použité senzory
- C) Zdrojové soubory pro externí komunikace
- D) Definice / vyplnění struktur pro použité komunikace
- E) Hlavní programová smyčka
- F) Celkové nastavení činnosti

Dále budou popsány jednotlivé části programového vybavení podrobněji doplněné vždy s ukázkami částmi kódů.

Jedním z důležitých souborů je `Def0Global.h`, ve kterém jsou zahrnuty všechny hlavičkové soubory. Tím se zajistí větší přehlednost a jednoduchost celého projektu. Tento soubor je zahrnut vždy v každém souboru.

6.1 Zpracování dat ze senzorů

6.1.1 Nastavení periferií a sběrnic

V projektu je vytvořený adresář AppSensors, ve kterém jsou zdrojové soubory .c a hlavičkové soubory .h k nastavení periferií pro vstupy dat a lokální sběrnice pro výše zmíněné typy senzorů.

Jak již bylo zmíněno, všechny použité senzory mají vlastní zdrojové soubory. Tyto zdrojové soubory obsahují vždy tři funkce. `Init()` , `ReadData()` a `GetValue()`. První zmíněná funkce `Init()` se volá pouze jednou, a to na začátku spuštění běhu programu. Jejím cílem je inicializovat daný senzor, periferie a lokální sběrnici. Je datového typu *EnBool*, což nám umožňuje použít návratovou hodnotu *EnTRUE* při úspěšné inicializaci, nebo *EnFALSE*, pokud něco selže.

Následující ukázka 1 je inicializace analogového vstupu použitého pro IR detektor plamene. Inicializační funkce `AnalogInit()` se díky programu STMCubeIDE vygeneruje sama, kde si také zvolíme příslušný port a pin, na kterém se má nakonfigurovat AD převodník.

```
1   ADC_HandleTypeDef hadc;
2   // interni ulozeni hodnoty prectene z externiho senzoru
3   uint32_t adc_val;
4   // inicializace AD prevodniku (senzor s analogovym vstupem)
5   EnBool AnalogInit()
6   {
7     ADC_ChannelConfTypeDef sConfig = {0};
8     //konfigurace AD prevodniku
9     hadc.Instance = ADC1;
10    hadc.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
11    hadc.Init.Resolution = ADC_RESOLUTION_12B;
12    hadc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
13      //  a dalsi inicializace
14     if (HAL_ADC_Init(&hadc) != HAL_OK)
15     {
16        return EnFALSE;
17     }
18    sConfig.Channel = ADC_CHANNEL_0;
19    sConfig.Rank = ADC_REGULAR_RANK_1;
20    sConfig.SamplingTime = ADC_SAMPLETIME_4CYCLES;
21    if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
22     {
23        return EnFALSE;
24     }
25      return EnTRUE;
26  }
```

Ukázka kódu 1: Funkce Init u analogového senzoru

6.1.2 Čtení hodnot ze senzorů

Poté co proběhne úspěšná inicializace daného senzoru, běh programu přejde do opakovaného volání funkcí `ReadData()` a `GetValue()`. Funkce `ReadData` vrací pouze datový typ *EnBool*. Funkce vyčte hodnotu z připojeného reálného senzoru a uloží ji do interní vyrovnávací paměti ve zdrojovém souboru příslušného senzoru, ovšem vrací nám pouze *EnTRUE* v případě, že vyčtení hodnoty bylo úspěšné a *EnFALSE* pokud ne. Poslední funkcí je již zmíněná funkce `GetValue`. Ta už vrací určitý datový typu, např. uint8_t, uint16_t a vrací tedy již skutečnou hodnotu pro další použití. Funkce `ReadData` tedy v podstatě pouze vyčte data ze senzoru po sběrnici, uloží je do interní proměnné ve zdrojovém souboru a signalizuje, zda data byla skutečně ze senzoru získána. Ovšem funkce `GetValue` interně použije již data přečtená ze senzoru, převede je na výstupní formát (upraví jednotky, přesnost, ...) a poskytne je pro další zpracování (například vyslání z modulu).

V ukázce 2 je typický příklad pro námi používaný analogový senzor. Nejprve voláme `EnBool AnalogReadData()`. Pokud vše proběhne správně, hodnota naměřená na pinu AD převodníku se uloží do proměnné *adc_val* a vrátí *EnTRUE*. Poté běh programu přejde do funkce `uint16_t AnalogGetValue()` a vrátí nám proměnnou *adc_val*. Toto rozdělení na dvě samostatné funkce je z důvodu celkové jednoduchosti možnosti snadné detekce selhání čtení hodnoty v periférii.

```
1   // cteni dat / hodnoty z realneho sensoru
2   EnBool AnalogReadData()
3   {
4     HAL_ADC_Start (&hadc) ; //start  ADC
5     HAL_ADC_PollForConversion (&hadc, 100); // pockej na konverzi
6     adc_val = HAL_ADC_GetValue (&hadc); // vycti hodnotu
7     HAL_ADC_Stop (&hadc); //stop adc
8     return EnTRUE; // pokud se to povedlo, vrat EnTRUE
9   }
10
11
12
13
14
```

```
15    // vraceni dat / hodnoty pro odeslani z modulu
16    uint16_t AnalogGetValue()
17    {
18        // zde muze byt uprava, korekce hodnoty pred jejim vydanim
19        return adc_val;
20    }
```

Ukázka kódu 2: Funkce ReadData a GetValue

Obdobným způsobem jsou vytvořeny trojice funkcí i pro další typy použitých senzorů / lokálních sběrnic.

6.2 Definice struktur pro senzory

Abychom byli schopni data ze senzorů periodicky zpracovávat je vhodné všechny senzory nějak z pohledu aplikace definovat. Do projektu jsem vložil dva nové soubory s názvem SensorMain.c(.h). Nejprve jsem v souboru SensorMain.h vytvořil strukturu s názvem `SensorInfo`, viz ukázka 3, popisující každý použitý senzor. Ta obsahuje šest prvků – ID (pořadové číslo senzoru), dataType (datový typ), odkaz na funkci Init, odkaz na funkci ReadData, odkaz na funkci GetValue a rovněž příznak zda je senzor povolen / aktivní.

```
1     typedef struct
2     {
3         uint8_t id;// poradove cislo sensoru 1,2,4.....
4         EnStatus status; //enable/disable
5         uint8_t dataType;  //datovy typ
6         EnBool (*PtrInit)(void);      // ukazatel na funkci Init
7         EnBool (*PtrReadData)(void);  // ukazatel na funkci ReadData
8         void (*PtrGetValue)(void);  // ukazatel na funkci GetData
9     } SensorInfo;
```

Ukázka kódu 3: Struktura SensorInfo

Některé základní datové typy použité při přenosu dat ze senzorů byly nadefinovány, ale lze je v budoucnu zcela libovolně rozšířit podle potřeby:

```
typedef enum
{
    EnDataTypeBool = 1,
    EnDataTypeUInt16 = 2,
    EnDataTypeFloat = 3,
    EnDataTypeUInt16x3 = 4,
} EnDataType;
```

Ukázka kódu 4: Nadefinové datové typy

Poté je v souboru SensorsMain.c vytvořen seznam struktur nazvaný `SensorsInfos` složený ze `SensorInfo`, kde každým řádek představuje nadefinovaný jeden použitý senzor. Zde je uveden příklad definic demonstračních použitých senzorů.

```
SensorInfo SensorsInfos[] =
{
//id, status, datovy typ, odkaz na Init, odkaz na ReadData, odkaz na GetVlue
{1, EnStatusTRUE, EnDataTypeBool, &DigiInit, &DigiReadData, (FuncPtrVoidToVoid)&
DigiGetValue},

{2,EnStatusTRUE, EnDataTypeUInt16, &AnalogInit, &AnalogReadData, (FuncPtrVoidToVoid)&
AnalogGetValue},

{3, EnStatusTRUE, EnDataTypeUInt16x3 , &MPUInit, &ReadAccelGyroMagData, (
FuncPtrVoidToVoid)&AccelGyroMagGetValue },

{4, EnStatusTRUE, EnDataTypeUInt16, &I2CInit, &I2CReadData, (FuncPtrVoidToVoid)
I2CGetValue},
};
```

Ukázka kódu 5: Seznam struktur SensorsInfos

Další krokem je již zmíněná inicializace každého z použitých senzoru. To je uskutečněno funkcí `SensorsAllInit()`, kde pomocí *for* cyklu procházíme postupně všechny definované senzory, tedy pro ně vytvořené struktury, a na každý voláme jeho inicializační funkci zaznamenanou v příslušné struktuře `SensorInfo`. Ukázka této funkce je až zde v ukázce 12 z důvodu dalších nutných vysvětlení.

6.3 Externí komunikace

Cílem práce je mimo jiné využít některé typy externích komunikací pro přenos získaných dat vytvářeným modulem na větší vzdálenosti. V projektu jsou plně použity dvě tyto komunikace – *USART* a *LAN*. Nicméně následující konfigurace umožní přidání další zcela libovolné vlastní komunikace. Definice komunikace je obdobná jako v případě definice senzorů. Nadefinujeme strukturu nazvanou `CommunInfo`, jejíž parametry budou id (ident komunikace), ukazatel na funkci Init a funkci Send určitého typu komunikace.

```
1  typedef struct
2  {
3      uint8_t id;        // poradove cislo komunikace 1,2,.....
4      EnBool (*PtrInit)(void);      // ukazatel na funkci Init
5      EnBool (*PtrSend)(void);  // ukazatel na funkci send
6  } CommunInfo;
```

Ukázka kódu 6: Struktura CommunInfo

Jako v případě definice datových typů nadefinujeme i komunikace. Opět je možné upravit dle potřeb.

```
1  typedef enum
2  {
3      EnCmnTypeUSART= 1,
4      EnCmnTypeLAN =  2,
5      EnCmnTypeLORA = 3,
6  } EnCmnType;
```

Ukázka kódu 7: Definice různých typů komunikací

Poté nadefinujeme seznam struktur `CommunInfos[]` obsahující všechny dostupné komunikační kanály.

```
CommunInfo CommunInfos[] =
{
      //typ komunikace, odkaz na Init, odkaz na odeslani
      { EnCmnTypeUSART, &CmnUSARTInit, &CmnUSARTSend },
      { EnCmnTypeLAN, &CmnLanInit, &CmnLanSend },
      { EnCmnTypeLORA, &CmnLORAInit, &CmnLORASend },
};
```

Ukázka kódu 8: Seznam struktur CommunInfos

Současně je definován odkaz na právě aktivní komunikaci pro její snadné využití v celém projektu.

```
CommunInfo *communActual = NULL;
```

Ukázka kódu 9: Odkaz na aktivní komunikaci

6.4 Formát datového paketu

Námi vysílaný datový paket je textového tvaru. To znamená, že budeme posílat pouze textové zprávy. Základní formát protokolu je zvolen následovně:

[AA TTT D =1;2459;0,5;1;0,9],

kde *AA* je adresa zařízení (pokud je použita), *TTT* je typ paketu (v našem případě DAT, jako datový), *D* je název skupiny hodnot v paketu. Zde vždy musí následovat znak =. Poté následují hodnoty ze senzorů oddělené středníkem. Celkový začátek a konec paketu označují hranaté závorky [ ].

6.5 Tvorba vysílaného paketu

Nyní se dostáváme k souboru s názvem AppMain. Ten obsahuje nejprve funkci `AppMainInit()`. Z této funkce voláme inicializaci zvolené komunikace a inicializaci všech definovaných senzorů. Funkce má následující tvar:

```
1   void AppMainInit()   //Inicializacni funkce volana z main.c pouze jednou
2  {
3       CommunInit(EnCmnTypeUSART); // inicializace vybrane externi komunikace
4       SensorsAllInit(); // inicializace vsech senzoru
5  }
```

Ukázka kódu 10: Funkce AppMainInit

Tyto dvě funkce mají velmi podobný tvar. Přičemž funkce `CommunInit` vypadá následovně. Pomocí *for* cyklu procházíme všechny typy definovaných komunikací, dokud nenarazíme na právě zadanou. Návrat je datového typu *EnBool* a pokud by tedy inicializace neproběhla korektně, dostaneme návratovou hodnotu *EnFalse*.

```
1   EnBool CommunInit(EnCmnType cmnType)
2   {
3     //prochazime vsechny typy komunikaci
4     for(int i=0; i<sizeof(CommunInfos)/sizeof(CommunInfo); i++)
5     {
6             //vyzvednuti informace o komunikaci
7             CommunInfo comm = CommunInfos[i];
8             //pokud je to pozadovana komunikace, inicializuj a vrat EnTRUE.
9             if(comm.cmnType == cmnType)
10             {
11                  // inicializace zadaneho typu komunikace
12                  comm.PtrInit();
13                  // uchovani odkazu na tento typ komunikace
14                  communActual = &comm;
15                  // zadana komunikace nalezena
16                  return EnTRUE;
17              }
18    }
19     // pozadovany typ komunikace nenalezen
20     return EnFALSE;
21  }
```

Ukázka kódu 11: Inicializace komunikace

Funkce `AllSensorsInit` má velmi podobný tvar, jen pomocí *for* cyklu procházíme všechny dostupné senzory, tedy pro ně definované struktury `SensorInfo` a na každý voláme jeho inicializační funkci zaznamenanou ve struktuře `SensorInfo`.

```
1   void SensorsAllInit()
2   {
3         //projiti vsech definovanych senzoru
4         for(int i=0; i<sizeof(SensorsInfos)/sizeof(SensorInfo); i++)
5         {
6             //vyzvednuti informace o senzoru
7             SensorInfo sensor = SensorsInfos [i];
8             //volani jeho inicializacni funkce
9             sensor.PtrInit();
10        }
11  }
```

Ukázka kódu 12: Inicializace všech senzorů

Druhou funkcí v souboru je `AppMainLoop()`, která je volána opakovaně a volá funkci `SensorsAllGetData()`.

```
1   void AppMainLoop().   //Funkce volana z main.c opakovane.
2   {
3     HAL_Delay(); //cekani mezi odesilanymi pakety
4     SensorsAllGetData(); //volani funkce pro ziskani dat
5   }
```

Ukázka kódu 13: AppMainLoop

Nyní už můžeme periodicky číst data z jednotlivých senzorů a vytvářet výstupní pakety, které budeme vysílat. Toto se děje právě ve funkci `SensorsAllGetData()`. Začátek vytváření paketu zajistíme zavolání funkce `PaketOutCreateAdrType(0x01, EnPaketTypeDAT)`, jejíž parametry jsou adresa a typ paketu. Poté do paketu vložíme název jeho datové části pomocí funkce `PaketOutAddName(„D“, ...)`. Už zbývá jen postupně vložit hodnoty z jednotlivých senzorů. To je řešeno pomocí přepínače *Switch*. Funkci `GetData`, přesněji řečeno kvůli její výstupní hodnotě, je nutno správně přetypovat. Po tomto přetypování už pouze získanou hodnotu zapíšeme do vytvářeného paketu do pomocí funkce`PaketOutAddNameInt`. Poté už

jen ukončíme paket voláním funkce `PaketPutEnd()` a odešleme přes vybranou komunikaci. Vzorový příklad funkce `SensorsAllGetData()` vypadá potom následovně:

```
1   void SensorsAllGetData()
2   {
3       // zacatek vytvareni paketu (asdresa, typ)
4       PaketOutCreateAdrType(0x01, EnPaketTypeDAT);
5       // prvni sensor D=...
6       PaketOutAddName("D\0x01");
7
8       // prochazime vsechny senzory
9       for(int i=0; i<sizeof(SensorsInfos)/sizeof(SensorInfo); i++)
10      {
11            SensorInfo sensor = SensorsInfos [i];
12                  //pokud bude status False, dej do paketu "Dis"
13                  if (sensor.status == EnStatusFALSE)
14                  {
15                      PaketOutAddNameString(NULL, "Dis\0x00");
16                      continue;
17                  }
18                    //pokud se nepovede precist data nebo je ulozit
19                    //napise se do paketu "Nul"
20                  if ((sensor.PtrReadData == NULL) || (sensor.PtrGetValue == NULL))
21                  {
22                      PaketOutAddNameString(NULL, "Nul\0x00");
23                      continue;
24                  }
25                  //pokud je ReadData false, napise se "NaN"
26                  if(sensor.PtrReadData() == EnFALSE)
27                  {
28                      PaketOutAddNameString(NULL, "NaN\0x00");
29                      continue;
30                  }
31
32
33
34
```

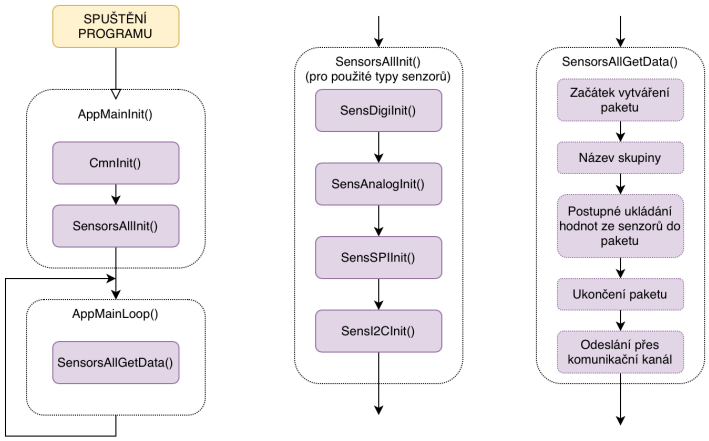
```
35
36          //prepinac pro prirazeni spravnych datovych typu
37          switch (sensor.dataType)
38          {
39
40                case EnDataTypeBool:
41                {
42                  uint8_t val =
43                  ((FuncPtrVoidToUInt8)(sensor.PtrGetValue))();
44                  // Funkce ve struct prijma void a vraci void.
45                  // Ted jsme to pretypovali na prijimani void a vraceni uint8.
46                  PaketOutAddNameInt(NULL, val);
47                  break;
48                }
49                case EnDataTypeUInt16:
50                {
51                  uint16_t val = ((FuncPtrVoidToUInt16)(sensor.PtrGetValue))();
52                  PaketOutAddNameInt(NULL, val);
53                  break;
54                }
55
56                // Datovy typ pro akcelerometr. Data jsou ulozena ve trech promennych.
57                // Bereme vzdy postupne, prevedeme na float a upravime
58                case EnDataTypeUInt16x3:
59                {
60                  int16_t val1 = ((FuncPtrUInt8ToInt16)(sensor.PtrGetValue))(1);
61                  float f1 = (float)val1/16384.0f;
62                  PaketOutAddNameFloat(NULL, f1, 2);
63
64                  int16_t val2 = ((FuncPtrUInt8ToInt16)(sensor.PtrGetValue))(2);
65                  float f2 = (float)val2/16384.0f;
66                  PaketOutAddNameFloat(NULL, f2, 2);
67
68                  int16_t val3 =((FuncPtrUInt8ToInt16)(sensor.PtrGetValue))(3);
69                  float f3 = (float)val3/16384.0f;
70                  PaketOutAddNameFloat(NULL, f3, 2);
71                  break;
```

```
72                  }
73              default :  {break;}
74            }
75        }
76        // ukonceni paketu
77        PaketOutEnd();
78        // odeslani odpovedi / paketu
79        if (communActual != NULL)
80        {
81        communActual->PtrSend();
82        }
83    }
```

Ukázka kódu 14: Ukázka funkce SensorsAllGetData

6.6 Princip vykonávání programu

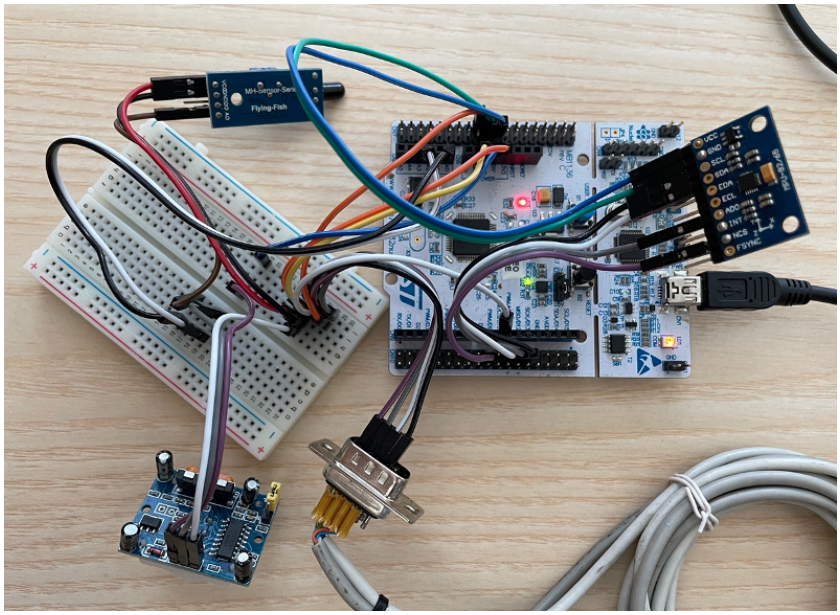
Výše byly jednotlivě popsány důležité části projektu. Pro ucelení a ujasnění celého běhu programu byl vytvořen následující diagram, obr. 13, právě pro naše použité demonstrační senzory.



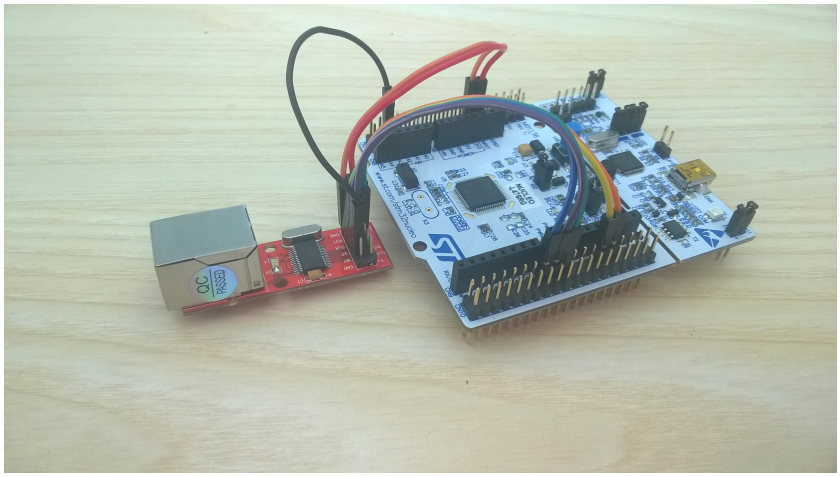
Obrázek 13: Principiální činnost programu ve vytvořeném senzorovém modulu

7 Výsledná realizace

Uspořádání projektu po hardwarové stránce bylo velmi jednoduché. Senzory byly propojeny přes nepájivé pole na zvolené porty mikrokontroléru. Celé zhotovené zapojení je na obrázku 14.



Obrázek 14: Reálná ukázka vytvořeného experimentálního senzorového modulu.



Obrázek 15: Vývojová deska s připojeným LAN modulem (ENC28J60)

7.1 Testovací zobrazení dat

Nejjednodušší a nejpřímější vyčtení / zobrazení paketu, jehož tvorba byla popsána výše je přes jakýkoli volně dostupný sériový terminál (tedy v případě použití komunikace typu *USART* a textového protokolu), například dostupnou aplikaci CoolTerm zobrazující přímo pakety přicházející z vytvářeného senzorového modulu. Ukázka těchto paketů je na obrázku. Jak je vidět, tak pakety odpovídají struktuře popsané v kapitole Datový paket. První je název / typ paketu *DAT*, to znamená, že se jedná o datový. *D* je název skupiny (představující data) a po znaku = následují informace o detektoru pohybu, poté hodnota z analogového senzoru na detekci plamene, další tři čísla jsou hodnoty z akcelerometru a posledním prvkem je chybová hláška *NaN*, což znamená, že poslední senzor (detektor osvětlení) neposkytl korektní data.

Ukázka takto vyslaných paketů je na obrázku 16.

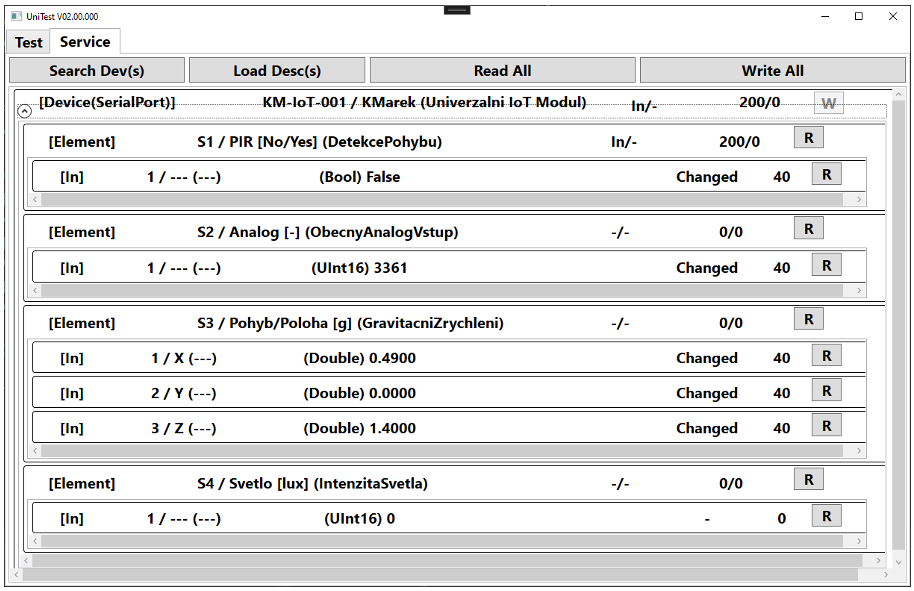
```
[DAT D=0;3623;0.9;0.35;1.11;NaN][DAT D=0;3597;0.9;0.35;1.10;NaN][DAT
D=0;3562;0.9;0.34;1.11;NaN][DAT D=0;3503;0.9;0.35;1.10;NaN][DAT D=0;3506;0.9;0.35;1.11;NaN][DAT
D=0;3505;0.9;0.35;1.10;NaN][DAT D=0;3504;0.9;0.35;1.10;NaN][DAT D=0;3507;0.9;0.35;1.10;NaN][DAT
D=0;3510;0.7;0.35;1.11;NaN][DAT D=0;3515;0.9;0.35;1.10;NaN][DAT D=0;3521;0.9;0.36;1.10;NaN][DAT
D=0;3523;0.9;0.35;1.11;NaN][DAT D=0;3523;0.9;0.35;1.11;NaN][DAT D=0;3637;0.8;0.35;1.9;NaN][DAT
D=1;3682;0.8;0.29;1.15;NaN][DAT D=0;3481;0.6;0.31;1.12;NaN][DAT D=0;3380;0.6;0.30;1.13;NaN][DAT
D=0;3453;0.6;0.30;1.12;NaN][DAT D=0;3624;0.6;0.30;1.13;NaN][DAT D=0;2806;0.6;0.30;1.13;NaN][DAT
D=0;3559;0.6;0.30;1.13;NaN][DAT D=0;3474;0.6;0.30;1.13;NaN][DAT D=0;3455;0.5;0.30;1.12;NaN][DAT
D=0;3450;0.4;0.31;1.11;NaN][DAT D=0;3451;0.6;0.29;1.14;NaN][DAT D=1;3453;0.5;0.30;1.13;NaN][DAT
D=0;3454;0.5;0.30;1.13;NaN]
```

Obrázek 16: Datové pakety

Pro externí komunikaci po *LAN* lze použít zcela stejný textový protokol (jako v případě *USART*), ovšem samozřejmě ještě zabalený do transportního *TCP/IP* protokolu. Pro vytvoření *LAN* komunikace byl použit již hotový modul s ENC28J60 využívaný velmi často v projektech typu Arduino. Pouze pro účely ověření možnosti využití *LAN* komunikace poskytl vedoucí práce částečně vytvořené zdrojové soubory pro tento typ komunikace. Tato komunikace byla tedy ověřena pomocí poskytnutých komponent, protože její skutečné vytvoření / zprovoznění není nikterak snadné.

Pro zobrazení přijatých dat (přes *USART* a *LAN*) v uživatelsky přijatelné formě používám demonstrační aplikaci napsanou v .NET Core / C# [25]. Obsah a formát aplikace je velmi jednoduchý a přímý. Každý připojený senzor tvoří jednu pod-tabulku a jeho data představují řádky této pod-tabulky. Formát pro každý senzor je stejný. V levé části je vždy Element, jakožto senzor. V prostřední části se vyskytuje číslo senzoru, neboli jeho ident, poté jeho název (zadaný do zobrazovací aplikace), jakého datového typu je výstupní hodnota a co měří

nebo snímá. V pravé části je kolik celkově přišlo datových paketů a zda byla změna přijaté veličiny. Ukázka z této aplikace je na obrázku 17



Obrázek 17: Demonstrační aplikace pro kontrolu dat přicházejících ze senzorového modulu

8 Rozšířený návrh

Po základním návrhu jsem se pokusil vytvořit i návrh rozšířený, který v sobě nese dvě varianty a které byly čistě experimentální pro ověření zda je možno tímto stylem senzorový modul skutečně rozšířit.

8.1 Propojení s Thunderboard Sense 2

Jednou z dostupných *IoT* technologií může být multi-senzorová deska Thunderboard Sense 2 [21] od společnosti Silicon Labs[6]. Deska nabízí řadu senzorů jako je například senzor teploty a vlhkosti, tlaku, kvality ovzduší a další. Dále RGB LED, mikrofon, radiové rozhraní a i uživatelské tlačítko. Tato deska je vhodná pro své uživatele i z dalšího důvodu a to je volně dostupná mobilní aplikace [22]. Její ukázku vidíme na obrázku 19. V aplikaci vidíme hodnoty z jednotlivých senzorů, můžeme rozsvěcet LED a mnoho dalších věcí. Další výhodou této desky je, že s ní lze pracovat jako s jiným mikrokontrolérem. Lze ji tedy přeprogramovat pro libovolný vlastní účel. S tímto modulem jsme pracovali ve škole původně v jiném předmětu, a tudíž jsem se rozhodl jej rovněž zakomponovat do této práce. Využil jsem jej jako bezdrátové tlačítko, kde stisk tlačítka (na modulu) vysílal přes Bluetooth signál "HELP"a bylo to vytvořeno jako další senzor do projektu "senzorového modulu".

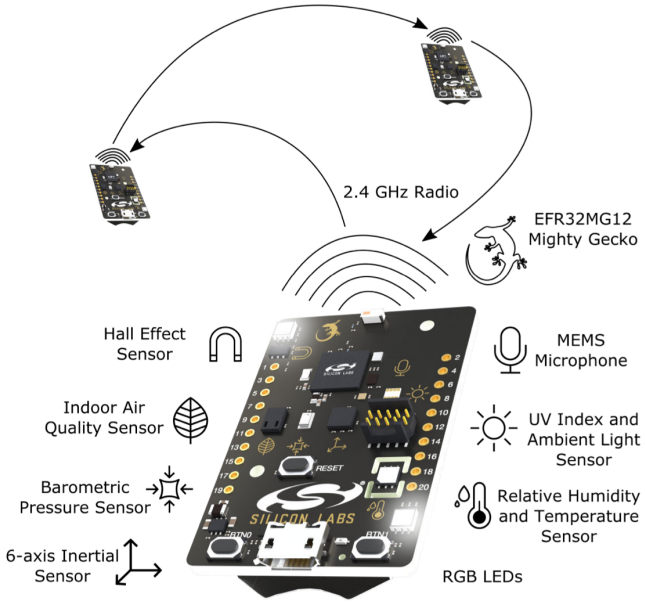
Thunderboard byl spárován pomocí technologie Blutooth s rozšiřujícím Bluetooth modulem X-NUCLEO-IDB05A1, který je přímo od STM a lze jej velmi jednoduše připojit na námi používanou vývojovou desku STM32-Nucleo [14]. Výstupem tohoto rozšířeného návrh bylo, že při stisku tlačítka na Thunderboardu se na Nucleu rozsvítila zelená LED. Rovněž je paket zasílaný na externí komunikaci rozšířen o další hodnotu se stavem 0 / 1 indikující stav tlačítka "HELP".

Datový paket měl potom obdobný tvar jako paket v základním návrhu. Opět začátek a konec paketu je ohraničený znaky [ ], typ paketu je *DAT*, tedy datový. Název skupiny je *T*, jako Thunderboard. Poté následuje znak = a stav tlačítka. Logická nula indikuje nestisknuto a tedy, že vše je v pořádku.

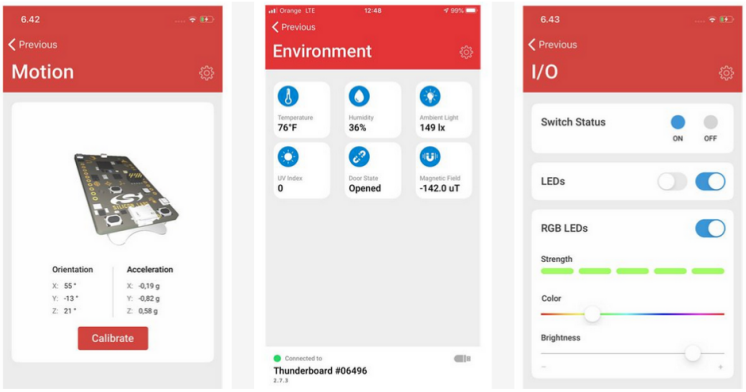
[6] https://www.silabs.com/development-tools/thunderboard/thunderboard-sense-two-kit

Logická jednička znamená stisknutí a tedy signalizaci "HELP".

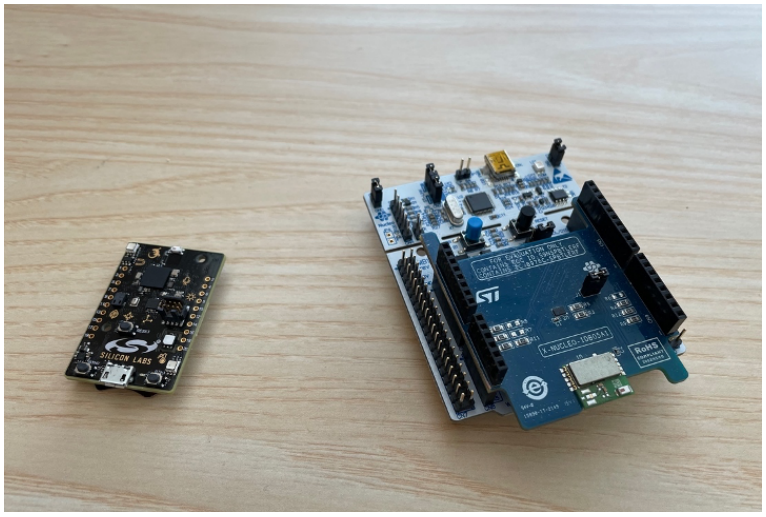
[DAT T = 0; KLID] [DAT T = 1; HELP].



Obrázek 18: Modul Thunderboard Sense 2 použitý jako simulace HELP tlačítka



Obrázek 19: Ukázka volně dostupné aplikace pro Thunderboard Sense 2



Obrázek 20: Vlevo Thunderboard, Vpravo Bluetooth modul nasazený na Nucleo

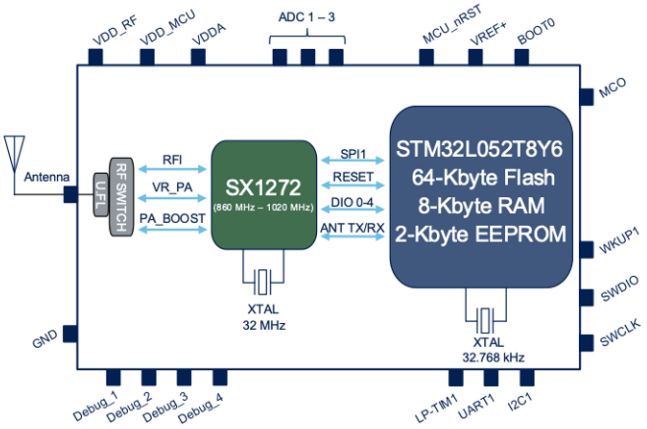
8.2 Využití přenosu dat pomocí LoRa

Pro vývojové desky typu STM32-Nucleo existuje přímo modul *LoRa*, který lze s Nucleem velmi jednoduše využít pro přenos dat. Tento modul bude zde použit jako externí komunikační kanál (pro přenos dat do externího / vzdáleného zařízení), nikoliv tedy jako senzor. Smyslem tohoto rozšířeného návrhu bylo vyzkoušet a ověřit činnost rozšiřující desky I–NUCLEO–LRWAN1. Vnitřní uspořádání modulu včetně konektorů je následující a znázorněné na obrázku 21. Modul obsahuje interní procesor nesoucí název STM32L052T8Y6, jehož paměť dosahuje velikostí 64 kB Flash, 8 kB RAM a 2 kB EEPROM.

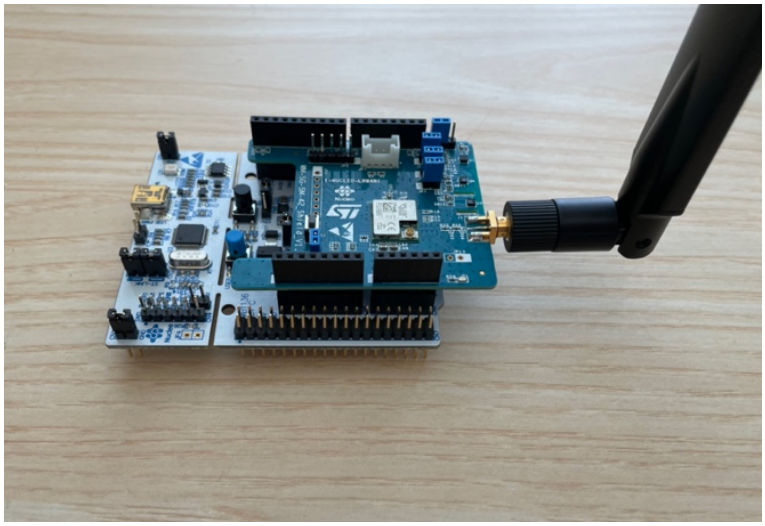
Poté se činnost modulu ověřila vypsáním nakonfigurovaných parametrů:

```
VERSION: 2.1.7, May 15 2022
             LOG: OFF
          AT ECHO: ON
         BAUDRATE: 115200bps
          MACADDR: B827EBFFFE87BD11
         ETHERNET: DHCP
             DNS1: 114.114.114.114
             DNS2: 8.8.8.8
       NTP SERVER: 1.ubuntu.pool.ntp.org
      EUI PADDING: {3, FF}, {4, FF}
       GATEWAY ID: 192.168.4.5
          LORAWAN: Public
   LORAWAN SERVER: thethingsnetwork.org
  UPLINK UDP PORT: 1780
DOWNLINK UDP PORT: 1780
         CHANNEL0: 471500000, A, SF7/SF12, BW125KHz  (LORA_MULTI_SF)
         CHANNEL1: 471700000, A, SF7/SF12, BW125KHz  (LORA_MULTI_SF)
         CHANNEL2: 471900000, A, SF7/SF12, BW125KHz  (LORA_MULTI_SF)
         CHANNEL3: 472100000, A, SF7/SF12, BW125KHz  (LORA_MULTI_SF)
         CHANNEL4: 472300000, A, SF7/SF12, BW125KHz  (LORA_MULTI_SF)
         CHANNEL5: 472500000, B, SF7/SF12, BW125KHz  (LORA_MULTI_SF)
         CHANNEL6: 472700000, B, SF7/SF12, BW125KHz  (LORA_MULTI_SF)
         CHANNEL7: 472900000, B, SF7/SF12, BW125KHz  (LORA_MULTI_SF)
         CHANNEL8: OFF                               (LORA_STANDARD)
         CHANNEL9: OFF                               (FSK)
```

Ukázka kódu 15: Výpis nakonfigurovaných parametrů



Obrázek 21: Vnitřní uspořádání LoRa modulu



Obrázek 22: Modul LoRa nasazený na vývojové desce Nucleo

Bohužel z důvodů nutnosti zaplacení přístupového tarifu a poněkud málo času ke konci semestru, nebylo toto rozšiřující řešení zcela implementováno do projektu a tedy ověřeno. Nicméně v projektu je připravený adresář CmnTypeLORA, kam bude možné doplnit konfigurační soubory. Základní činnost modulu LoRa se povedlo ověřit, modul s programovým kódem spolupracuje, a tedy jeho připojení na reálnou LoRa síť by neměl být velkým problémem.

9 Závěr

Cílem této práce bylo nejprve zmapovat veřejně dostupné *IoT sítě*. Byly vybrány tři nejrozšířenější v České republice a popsány jejich důležité vlastnosti z pohledu této práce. Následně bylo nutné navrhnout a zrealizovat vlastní senzorový modul, který by mohl být v budoucnu do takové sítě připojitelný. Tento senzorový modul byl navrhován s ohledem na možnost použití jako jednoduchý dohledový modul / systém pro starší a zejména samostatně žijící osoby v domácnosti.

Za tímto účelem byly zvoleny vzorové demonstrační senzory využitelné právě v těchto případech a to zejména senzor plamene, senzor světla a pohybu. Při jejich použití byly rovněž současně vybrány vzorové lokální sběrnice pro jejich připojení na mikrokontrolér. V budoucnu lze samozřejmě použít (skoro) libovolné senzory a lokální sběrnice. Velkým přínosem této práce je navržená jednotná práce nejen se senzory, ale zejména z nich získanými daty pro další manipulaci. Použité senzory jsou do projektu přidávány pomocí zcela definovaným postupem a tím je vytvořený modul do budoucna snadno rozšiřitelný. Dobře definována je nejen strana (vstupních) senzorů, ale rovněž i (výstupních) externích komunikací. Taktéž pouhým přidáním vhodně definovaných částí, zle začlenit do modulu v podstatě libovolnou vlastní komunikaci.

Práce však zachází poněkud dále a nabízí rozšířenou variantu modulu. Ta obsahuje *BlueTooth* modul (vývojová deska pro NUCLEO) pro možnost příjmu bezdrátových zařízení, například osobní tlačítko přivolání pomoci. Rovněž práce pojednává o možnosti připojení celého vytvořeného senzorového modulu do *IoT sítě LoraWAN*. Tato část byla podrobněji rozpracována a experimentálně vyzkoušena, avšak z časových důvodů nebylo připojení vytvořeného modulu do reálné *IoT sítě* vytvořeno.

Na první a obecný pohled se může zdát, že název práce „Využití *IoT technologií* ...“ není v práci dostatečně využit / zmíněn. Avšak pod pojmem *IoT* nezahrnujeme pouze skutečnou komunikaci jednotlivých senzorů přes internet, ale také jakoukoli činnost / nasazení senzorů, které jsou do internetu nějak (i nepřímo) připojeny, nebo jej aspoň nějak pro svou činnost využívají. V práci se tedy jedná zejména o vytvoření senzorového modulu, jehož cílem je právě přenos dat po internetu a to jakýmkoli způsobem, buď běžným (domácí přípojka), nebo speciálním určeným v podstatě pouze pro potřeby *IoT* (radiové *LoRa*).

Cíle práce byly tedy splněny ve všech bodech zadání. Navrhnutý modul je schopen činnosti jak po softwarové, tak hardwarové stránce. Komunikace (se senzory, nebo dálkové) tvoří vždy velkou část práce při tvorbě takových zařízení. Jedná se tedy o zařízení jako velmi vhodný základ pro rychlé a snadné vlastní použití při experimentech v oblasti dohledu v domácím prostředí.

Literatura a zdroje

[1] KOĎOUSKOVÁ, Barbora. *Internet věcí (IOT): Definice,příklady využití, produkty.* [online]. [cit. 2. 12. 2021]. Dostupné z: https://www.rascasone.com/cs/blog/iot-internet-veci-definice-produkty-historie.

[2] POHANKA, Pavel. *Internet věcí.* [online]. [cit. 1. 12. 2021]. Dostupné z: http://i2ot.eu/internet-of-things.

[3] Sigfox. *Ceník základní konektivity.* [online]. [cit. 5. 2. 2022]. Dostupné z: https://sigfox.cz/cs/o-nas/cenik-vop.

[4] RedLinx. *LYNX BEE Sigfox Module RCZ1.* [online]. [cit. 1. 5. 2022]. Dostupné z: https://redlinx.net.au/index.php/shop/46/8/wireless-m2m-iot-products/m2m-sensors/lynx-bee-sigfox-module-rcz1-ufl-detail.

[5] TEM. *NB IOT CLICK MIKROELEKTRONIKA.* [online]. [cit. 1. 5. 2022]. Dostupné z: https://www.tme.eu/cz/details/mikroe-3294/rozsirujici-desky/mikroelektronika/nb-iot-click/.

[6] LORA, Alliance. *What is Lorawan.* [online]. [cit. 2. 12. 2021]. Dostupné z: https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf.

[7] ST Microelectronics. *B-L072Z-LRWAN1.* [cit. 2. 12. 2021]. Dostupné z: https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html

[8] JUŘÍK, David. *Experimentální senzorové moduly pro tvorbu hraček a domácího monitorování.* [online]. [cit. 1. 5. 2022]. Dostupné z: https://dspace.cvut.cz/handle/10467/87605.

[9] JÍNA, Marek. *Přenos dat ze senzorů při tvorbě experimentálních hraček a rehabilitačních pomůcek.* [online]. [cit. 1. 5. 2022]. Dostupné z: https://dspace.cvut.cz/handle/10467/99235.

[10] HC-SR501 PIR MOTION DETECTOR. *Product Description.* [online]. [cit. 3. 4. 2022]. Dostupné z: https://www.mpja.com/download/31227sc.pdf.

[11] Hadex. *Měřič osvětlení-luxmetr, modul GY-302 s BH1750* [online]. [cit. 1. 5. 2022]. Dostupné z: https://www.hadex.cz/m541-meric-osvetleni-luxmetr-modul-gy-302-s-bh1750/?gclid=CjwKCAjwu_mSBhAYEiwA5BBmfz63ow6dBv67Z_RSfF0T6DKAj7WSiEbqC8BcqC4Se9zyUgnBov72MhoClmoQAvD_BwE.

[12] Drátek.cz. *Senzor Plamene Infračervený Detekční Modul.* [online]. [cit. 1. 5. 2022]. Dostupné z: https://dratek.cz/arduino/1520-senzor-plamene-infracerveny-detekcni-modul.html?gclid=CjwKCAjwo8-SBhAlEiwAopc9W9AOQcDH3_a_EDG7fVYJAKxqcytHfSwp-5BE-8_DTPYPPu6F7BJ-ehoCYGAQAvD_BwE.

[13] Standford. *MPU-9255 Product Specification.* [online]. [cit. 1. 5. 2022]. Dostupné z: https://stanford.edu/class/ee267/misc/MPU-9255-Datasheet.pdf.

[14] ST Microelectronics. *Nucleo-L152RE.* [online]. [cit. 6. 4. 2022]. Dostupné z: https://www.st.com/en/evaluation-tools/nucleo-l152re.html.

[15] NOVIELLO, Carmine. *Mastering STM32.* LeanPub, 2017.

[16] ST Microelectronics. *Description of STM32L1 HAL and low-layer drivers.* [online]. [cit. 1. 5. 2022]. Dostupné z: https://www.st.com/content/ccc/resource/technical/document/user_manual/97/4d/5f/9a/ed/e4/4e/66/DM00132099.pdf/files/DM00132099.pdf/jcr:content/translations/en.DM00132099.pdf.

[17] SCHAFFEROVÁ, Magdaléna. *Sigfox – nejpomalejší síť, kterou by měl chtít každý.* [online]. [cit. 2. 12. 2021]. Dostupné z: https://www.zooco.io/blog/sigfox-nejpomalejsi-sit-kterou-by-mel-chtit-kazdy/.

[18] IoT portal. *Sigfox versus NB-IoT, která IoT síť zvítězí?.* [online]. [cit. 2. 12. 2021]. Dostupné z: https://www.iot-portal.cz/2020/04/16/sigfox-versus-nb-iot-ktera-iot-sit-zvitezi/.

[19] Electronics Notes. *SigFox for M2M & IoT.* [online]. [cit. 1. 5. 2022]. Dostupné z: https://www.electronics-notes.com/articles/connectivity/sigfox/what-is-sigfox-basics-m2m-iot.php.

[20] IoT portal. *Lorawan*. [online]. [cit. 2. 12. 2021]. Dostupné z: https://www.iot-portal.cz/2016/02/29/lorawan/.

[21] Silicon Labs. *UG309: Thunderboard Sense 2 User's Guide*. [online]. [cit. 22. 3. 2022]. Dostupné z: https://www.silabs.com/documents/public/user-guides/ug309-sltb004a-user-guide.pdf.

[22] Silicon Labs. *Sltb004a: Explore the capabilities of thunderboard sense*. [online]. [cit. 23. 3. 2022]. Dostupné z: https://www.silabs.com/development-tools/thunderboard/thunderboard-sense-two-kit.

[23] Mectechnicalservices. *HC SR501 PIR Motion Sensor Module*. [online]. [cit. 3. 4. 2022]. Dostupné z: http://www.mectechnicalservices.com/modules-c-17/hc-sr501-pir-motion-sensor-module-p-36.

[24] HAMAN, Martin.*Kontrolní a řídící modul s IoT*. [online]. [cit. 2. 12. 2021]. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=175260.

[25] PRICE, Mark. *C# 8.0 and .NET Core 3.0 - Modern Cross-Platform Development*. Packt. 2019. ISBN 978-1788478120.