
# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION


# SBĚR DAT S POMOCÍ BOSCH XDK A JEJICH BEZDRÁTOVÝ PŘENOS DO LABVIEW.
DATA ACQUISITION WITH BOSCH XDK AND WIRELESS DATA TRANSFER TO LABVIEW.

DIPLOMOVÁ PRÁCE
MASTER'S THESIS


AUTOR PRÁCE — Bc. Filip Vítek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Stanislav Pikula, Ph.D.
SUPERVISOR



BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

***Student:*** Bc. Filip Vítek ***ID:*** 203540

***Ročník:*** 2 ***Akademický rok:*** 2021/22

**NÁZEV TÉMATU:**

## Sběr dat s pomocí Bosch XDK a jejich bezdrátový přenos do LabVIEW.

**POKYNY PRO VYPRACOVÁNÍ:**

Cílem práce je využití zařízení Bosch XDK pro záznam dat z integrovaných snímačů a jejich bezdrátový přenos do aplikace v LabVIEW. Součástí řešení by měla být bezdrátová možnost konfigurace Bosch XDK. Zadání diplomové práce lze shrnout do následujících bodů:

1. Popište zařízení Bosch XDK a rozeberte dostupná komunikační rozhraní z hlediska použitelnosti pro zamýšlenou aplikaci z hlediska komunikační rychlosti a limitů vzdálenosti.
2. Proveďte návrh firmwaru pro Bosch XDK a aplikace v LabVIEW pro realizaci zadání.
3. Realizujte navržený firmware a aplikaci v LabVIEW.
4. Ověřte praktickými experimenty vlastnosti výsledného řešení jako datová propustnost komunikačního kanálu, dosah, výdrž baterie apod.

**DOPORUČENÁ LITERATURA:**

Vlach, Jaroslav, Josef Havlíček, and Martin Vlach. Začínáme s LabVIEW. BEN-technická literatura, 2008.

***Termín zadání:*** 7.2.2022 ***Termín odevzdání:*** 18.5.2022

***Vedoucí práce:*** Ing. Stanislav Pikula, Ph.D.

**doc. Ing. Petr Fiedler, Ph.D.**
předseda rady studijního programu

# Abstrakt

Tato práce se převážně zabývá návrhem a implementací firmwaru pro zařízení Bosch XDK 110 a stejně tak LabVIEW aplikace, které byly následně otestovány a také byly prověřeny některé vlastnosti. V zařízení jsou vyčítána data ze snímačů, která jsou následně odesílána bezdrátově do LabVIEW aplikace, kde jsou zobrazována, ukládána či dále jinak zpracovávána. Pro bezdrátovou komunikaci je používána WLAN s protokolem MQTT. V práci je prověřena výdrž zařízení při různých teplotách okolí, a při různých frekvencích vyčítání a odesílání dat. Výsledná implementace byla testována při měření enviromentálních dat v lednici, měření zrychlení po připevnění zařízení k autu nebo měření magnetické indukce a hluku elektromotoru.

# Klíčová slova

Bosch XDK 110, LabVIEW, Eclipse Mita, MQTT, Bezdrátová komunikace

# Abstract

This thesis is mostly concerned with designing and implementation of firmware for Bosch XDK 110 device and a LabVIEW application, which were both tested and also some of the features were explored as well. Data from the sensors in device are aquired and then wirelessly send into LabVIEW app, where that data are displayed, stored or processed in different way. WLAN is used for wireless communication with MQTT protocol. In thesis battery life was tested in different ambient temperatures and different frequencies of data sending and data acqusition. Final implementation was tested when measuring enviromental data in fridge, measuring acceleration when device was attached to a car or measuring magnetic induction and noise of electric motor.

# Keywords

Bosch XDK 110, LabVIEW, Eclipse Mita, MQTT, Wireless communication

# Bibliografická citace

VÍTEK, Filip. Sběr dat s pomocí Bosch XDK a jejich bezdrátový přenos do LabVIEW.. Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/142470. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Stanislav Pikula

# Prohlášení autora o původnosti díla

| | |
|---|---|
| **Jméno a příjmení studenta:** | *Filip Vítek* |
| **VUT ID studenta:** | *203540* |
| **Typ práce:** | *Diplomová práce* |
| **Akademický rok:** | *2021/22* |
| **Téma závěrečné práce:** | *Sběr dat s pomocí Bosch XDK a jejich bezdrátový přenos do LabVIEW* |

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 10. května 2022 ----------

podpis autora

# Poděkování

Děkuji vedoucímu diplomové práce Ing. Stanislavu Pikulovi, PhD za odbornou a pedagogickou pomoc a další cenné rady, které byly nápomocné při vytváření této závěrečné práce.

V Brně dne: 10. května 2022 ----------

# Obsah

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# Úvod

Zařízení Bosch XDK 110 použité v této práci je vývojový nástroj od formy Bosch obsahující různé senzory pro snímání okolního prostředí. Poskytuje výhodné vlastnosti pro použití například při vývoji nových zařízení či jejich testování a nalézá velké uplatnění v IoT (Internet of Things) aplikacích. Lze jej snadno přenést a umístit do měřeného prostředí či upevnit na zkoumaný objekt a díky baterii a možnosti bezdrátové komunikace či slotu na paměťovou kartu, kde mohou být data ukládána, nemusí být zařízení trvale drátově připojeno. Součástí je i možnost stažení si vývojového prostředí XDK-Workbench pro programování firmwaru, které mimo jiné obsahuje knihovny, funkce a různé příklady algoritmů pro ovládání hardwaru XDK 110.

LabVIEW je prostředí kde lze graficky programovat a lze jej s výhodou použít při měření, testování či zpracovávání dat a podobně. Tato práce je zaměřena na spojení zmíněných dvou nástrojů (LabVIEW a XDK 110), aby bylo možné výsledek univerzálně použít pro měření různých veličin, které je možné získávat pomocí dostupných snímačů v zařízení.

Práce je rozdělena do pěti základních částí. První kapitola v tomto dokumentu popisuje zařízení Bosch XDK 110 včetně snímačů a komunikačních rozhraní. Výběrem způsobu bezdrátové komunikace, protokolu nebo jazyka pro programování firmwaru se zabývá druhá kapitola. V kapitole číslo 3 je popsán návrh výsledného softwaru, tedy firmwaru pro XDK 110 a LabVIEW aplikace. Ve čtvrté kapitole je pak uvedena praktická část, například zprovoznění zařízení a bezdrátové komunikace, vývoj LabVIEW aplikace a další. V poslední kapitole, zaměřené na testování, je možné nalézt zjišťování výdrže zařízení nebo měření okolních podmínek v lednici, jako praktickou ukázku možného použití.

# 1. POPIS BOSCH XDK 110

V této závěrečné práci, jak již bylo zmíněno, je použit vývojový kit Bosch XDK 110. U tohoto zařízení se udává, že může být použito v teplotním rozsahu -20 °C až +60 °C. V případě nabíjení baterie je pak doporučená teplota omezena na 0 °C až +45 °C. Rozmezí vlhkosti je od 10 % do 90 % a stupeň ochrany zařízení je IP 30. Napájecí napětí je 5 V stejnosměrných a maximální proud je 500 mA.

Pro přímé ovládání či vizuální komunikaci jsou zde dostupná dvě tlačítka a několik LED diod. Diody jsou v barvách červená, zelená, oranžová a žlutá s tím, že programově jdou ovládat pouze červená, oranžová a žlutá a zbylá zelená indikuje stav nabíjení.


Obrázek 1.1 Bosch XDK 110 [1]

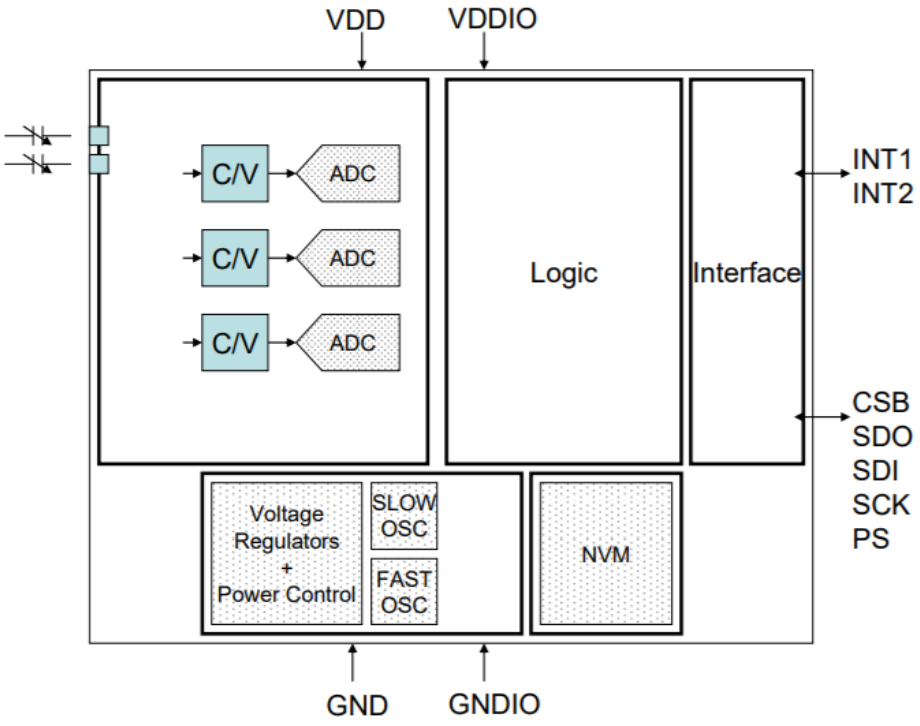
## 1.1 Snímače

V zařízení jsou použity pro snímání okolních podmínek či jiných fyzikálních veličin různé snímače. Mezi tyto senzory patří akcelerometr, gyroskop, magnetometr, senzor teploty, vlhkosti a tlaku, či senzory okolního hluku a osvětlení. V následujících podkapitolách budou tyto senzory popsány blíže, převážně pak jejich základní vlastnosti či principy.

### 1.1.1 Akcelerometr BMA280

Pro snímání zrychlení je použit tříosý MEMS akcelerometr s typovým označením BMA280. Mezi jeho základní vlastnosti patří rozsah, který je nastavitelný v hodnotách ±2 g, ±4 g, ±8 g a ±16 g. Nastaveným rozsahem je ovlivněna i citlivost snímače, a to s těmito hodnotami ve stejném pořadí: 4096 LSB/g, 2048 LSB/g, 1024 LSB/g a 512 LSB/g. A stejně tak je ovlivněno i rozlišení snímače jako 0.244 mg/LSB, 0.488 mg/LSB,

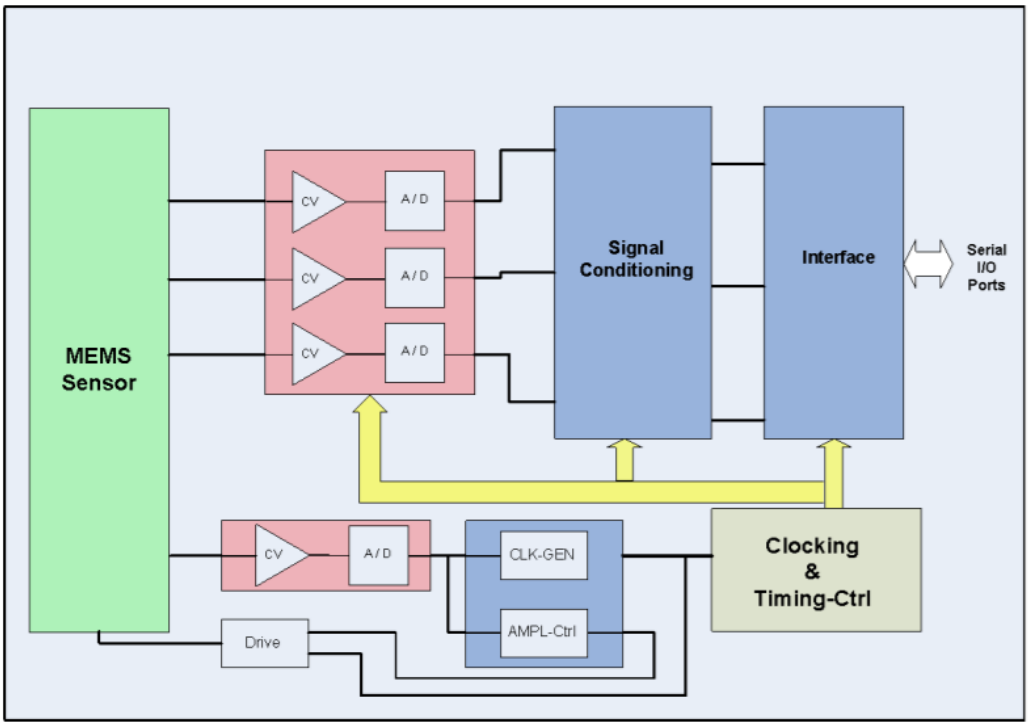
0.977 mg/LSB a 1.953 mg/LSB. Mezi další důležité parametry patří šířka pásma, která se pohybuje dle nastavení od 7.81 Hz až do 500 Hz a maximální frekvence vzorkování dat je 2000 Hz při nefiltrování dat. Digitální rozlišení tohoto senzoru je 14 bitů a výstup je taktéž digitální. Za zmínku stojí i teplotní rozsah, ve kterém je možné snímač použít, a to od -40 °C do +85 °C. Přítomen je také senzor teploty se stejným rozmezím jako je výše zmíněný pracovní teplotní rozsah akcelerometru. [2]




Obrázek 1.2 Blokové schéma BMA280 [1]

### 1.1.2 Gyroskop BMG160

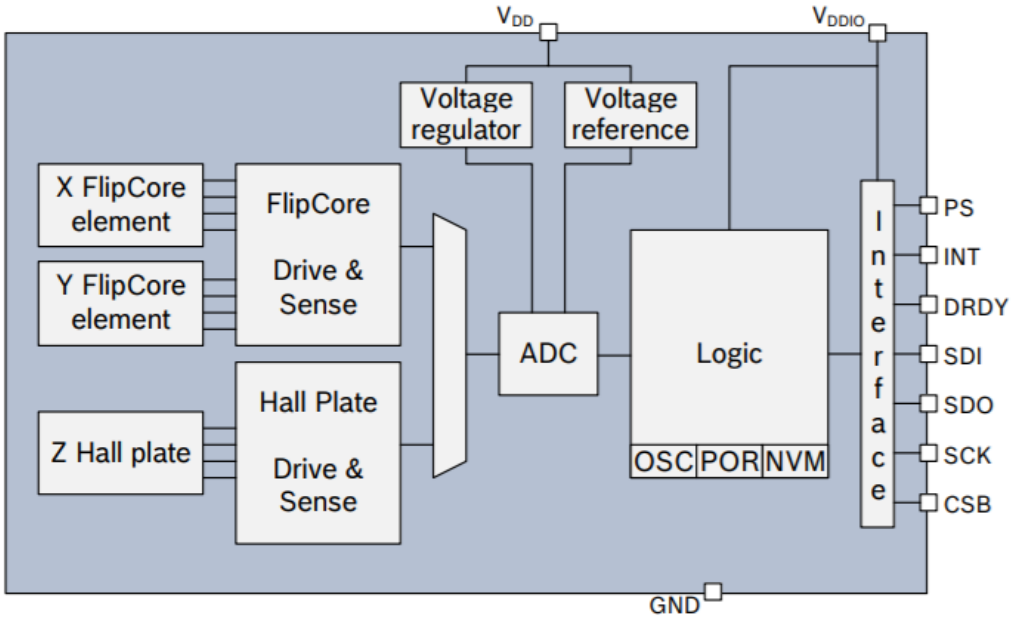
Stejně jako akcelerometr, je i gyroskop od firmy Bosch. Tento tříosý snímač má 5 rozsahů od 125 °/s do 2000 °/s, které jsou programově nastavitelné. Opět je voleným rozsahem ovlivňována i citlivost snímače, a to od 16.4 LSB/°/s až po 262.4 LSB/°/s. Nastavitelná je i šířka pásma, a to v hodnotách od 12 Hz do 230 Hz včetně možnosti nefiltrování dat. Vzorkovací frekvence převodníku může být od 100 Hz až do 2000 Hz. Digitální rozlišení je v tomto případě 16 bitů. Stejně jako u předchozího snímače je i zde přítomen senzor teploty s rozsahem -40 °C až +85 °C, což je totožné s pracovním rozsahem teploty gyroskopu. Typické použití tohoto snímače je v mobilních telefonech, pro stabilizaci obrazu, v herním průmyslu či pro navigaci. [3]




Obrázek 1.3 Blokové schéma BMG160 [2]

### 1.1.3 Magnetometr BMM150

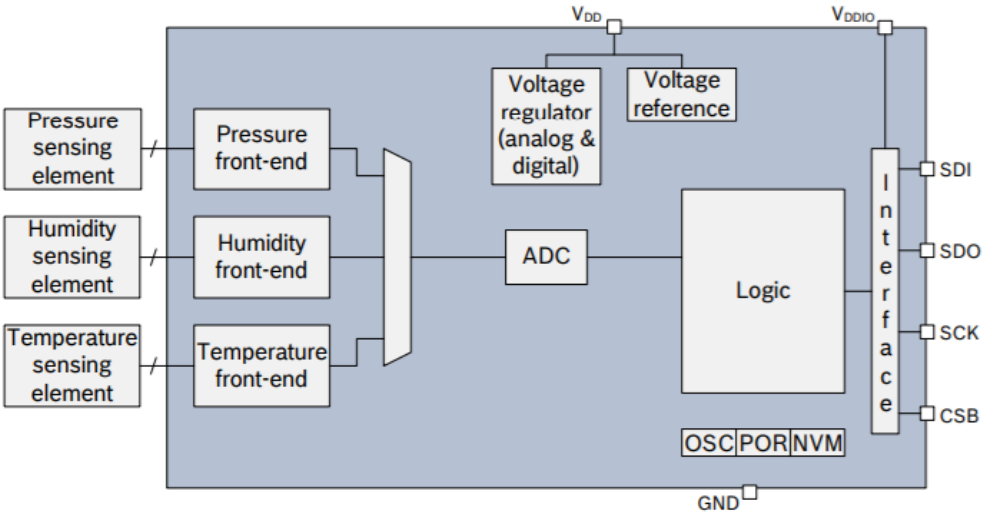
Pro snímání magnetického pole je v tomto zařízení umístěn geomagnetický senzor BMM150. Jako u předchozích, je i tento snímač tříosý s rozsahy ±1300 µT pro osy X a Y a ±2500 µT pro osu Z s rozlišením 0.3 µT. Vzorkovací frekvence se zde pohybuje v různých módech od 10 Hz až po 300 Hz. Teplotní rozsah, ve kterém může být snímač použit, je stejně jako v předchozích případech v hodnotách od -40 °C po +85 °C. Bitové rozlišení je zde různé v závislosti na osách. Pro osy X a Y je zde rozlišení 13 bitů a pro zbylou osu Z je zde rozlišení 15 bitů. Pro teplotní kompenzaci je zde také měřen odpor Hallova senzoru a zde se jedná o informaci obsahující 14 bitů. I zde je tedy digitální výstup za pomocí rozhraní SPI a I2C. Používá se pro informaci o magnetickém kurzu, jako elektronický kompas, pro rozšířenou realitu a další. [4]




Obrázek 1.4 Blokové schéma BMM150 [3]

### 1.1.4 Senzor teploty, vlhkosti a tlaku BME280

Senzor vlhkosti BME280 je i vybaven senzorem barometrického tlaku a teploty. Rozsahy tohoto snímače jsou 0 % až 100 % pro senzor vlhkosti, 300 hPa až 1100 hPa pro senzor tlaku a nakonec -40 °C až +85 °C pro senzor teploty. U senzoru relativní vlhkosti je rozlišení 0.008 %, hystereze ±1 % a nelinearita 1 %. Přesnost snímače tlaku se pohybuje v závislosti na zvoleném módu od ±1 hPa až ±1.7 hPa. Je vhodné také zmínit rozlišení, které je 0.18 hPa. U teplotního snímače je přesnost opět dána nastaveným módem od ±1 °C do ±1.5 °C s rozlišením 0.01 °C. Digitální rozlišení jednotlivých prvků je 16 bitů pro měření vlhkosti, 16, 18 nebo 20 bitů pro měření tlaku v závislosti na IIR filtru, stejně jako 16, 18 či 20 bitů pro měření teploty. Snímač má vzorkovací frekvenci až 182 Hz. Typické použití například při ovládání chytré domácnosti jako topení, ventilace či klimatizace. Dále pro domácí meteostanice, hodinky, kamery a jiné. [5]

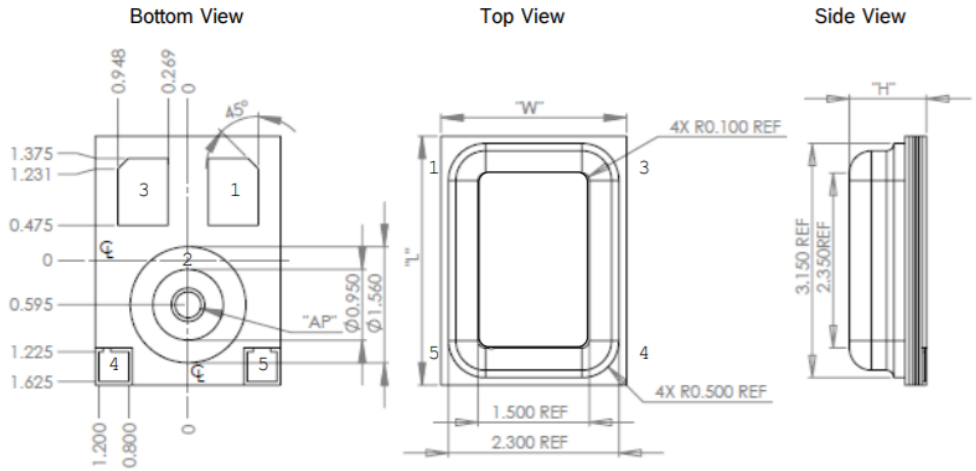



Obrázek 1.5 Blokové schéma BME280 [4]

### 1.1.5 Senzor okolního hluku AKU340

Použitý MEMS mikrofon má poměr signálu k šumu (SNR) 62 dB při frekvenci 1 kHz a frekvenční rozsah ±3 dB od 60 Hz do 12.5 kHz. Citlivost se pohybuje od -36 do -40 dBV/Pa a výstupní impedance je 150 Ω. Výstup ze senzoru je analogový a také je dobré zmínit že je všesměrový. [6]

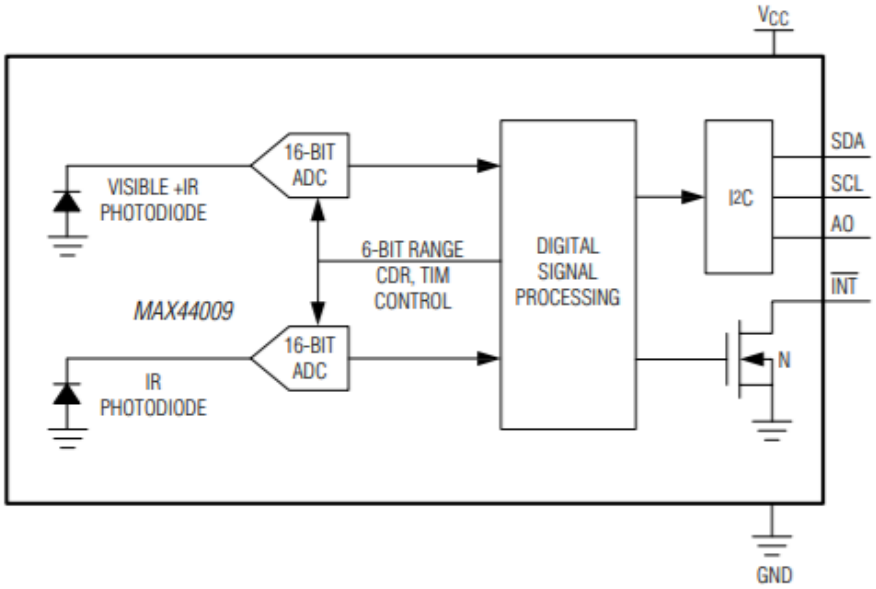
Kovové provedení obalu umožňuje dobrou ochranu proti elektromagnetickému či radiofrekvenčnímu rušení, a tedy je vhodný pro použití v bezdrátových zařízeních. Tento snímač se například používá pro chytré telefony, tablety, drátová a bezdrátová sluchátka a podobně. [6]




Obrázek 1.6 Pouzdro AKU340 [5]

### 1.1.6 Senzor okolního osvětlení MAX44009

Úroveň okolního osvětlení je snímáno pomocí senzoru s rozsahem od 0.045 luxů až 188 000 luxů. Hodnota 0.045 je i maximální citlivost v jednotkách lux/LSB. Digitální rozlišení dosahuje až 22 bitů a udávaný čas pro ADC konverzi je přibližně 100 ms. Výstup je tedy digitální a používá standard I2C. Výhodou tohoto snímače je také velmi nízký proud, který je 0.65 µA. Teplotní rozsah pro snímač je opět od -40 °C do +85 °C. Senzor se používá pro tablety či počítače a pro různá přenosná zařízení a bezpečnostní systémy. [7]



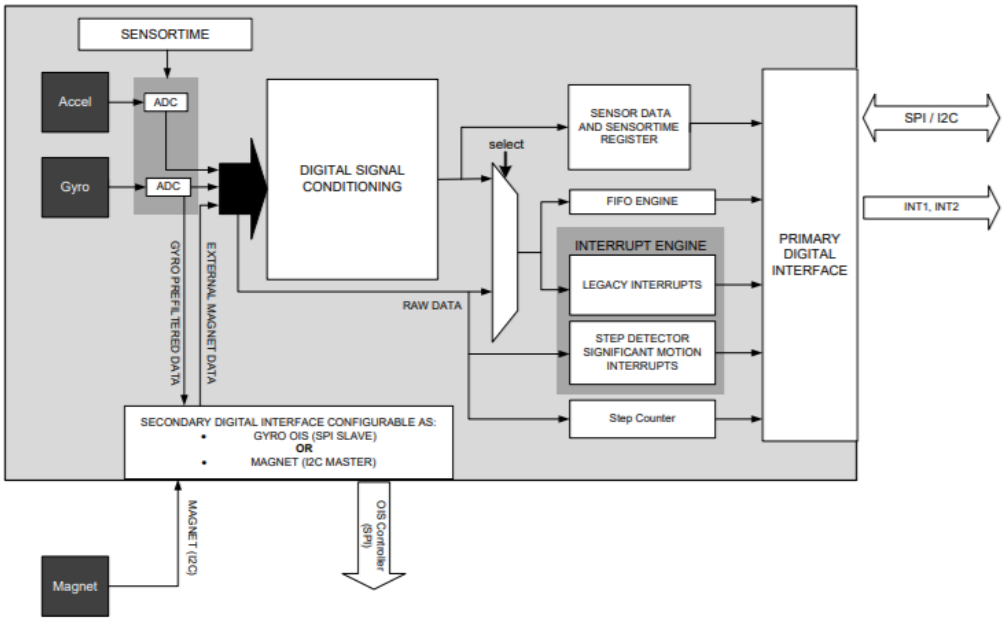

Obrázek 1.7 Blokové schéma MAX44009 [6]

### 1.1.7 Jednotka inerciálního měření BMI160

Tato jednotka je složena ze dvou snímačů, a to tříosého akcelerometru a tříosého gyroskopu. Je navržena po přesné měření s nízkou spotřebou například pro chytré telefony a hodinky, tablety, herní ovladače a další. Spotřeba je maximálně 925 µA a napájecí napětí se pohybuje od 1.71 V do 3.6 V. Snímače jsou hardwarově synchronizovány, včetně možnosti synchronizace externího snímače, například magnetometru, díky čemu jsou vhodné pro aplikace jako rozšířenou realitu nebo navigaci, kde je vyžadována vysoká přesnost dat. Součástí měřící jednotky je i paměť typu FIFO o velikosti 1024 bajtů schopná zvládnout i data z externího senzoru. Teplotní rozsah použití je od -40 °C do +85 °C jako i u zbylých snímačů. Je zde dostupný digitální výstup se standardem SPI nebo I2C. [8]

Akcelerometr má programově nastavitelné rozsahy ±2 g, ±4 g, ±8 g a ±16 g s citlivostmi od 16384 LSB/g do 2048 LSB/g. Digitální rozlišení je 16 bitů a rychlost vzorkování dat je od 12.5 Hz do 1600 Hz. [8]

Gyroskop má také programově nastavitelné rozsahy. Celkem je jich pět, a to v hodnotách od 125 °/s po 2000 °/s. Citlivost v závislosti na nastaveném rozsahu se pohybuje v rozmezí 16.4 LSB/°/s až 262.4 LSB/°/s. Frekvence vzorkování je v hodnotách od 25 Hz do 3200 Hz s digitálním rozlišením až 16 bitů. [8]




Obrázek 1.8 Blokové schéma BMI160 [7]

## 1.2 Komunikační rozhraní

V této podkapitole budou rozebrány možné způsoby, jak může zařízení komunikovat se svým okolím, ať už pomocí sběrnice (USB, rozšiřující deska s piny, JTAG), či bezdrátově (Bluetooth, WLAN, LoRa).

### 1.2.1 USB

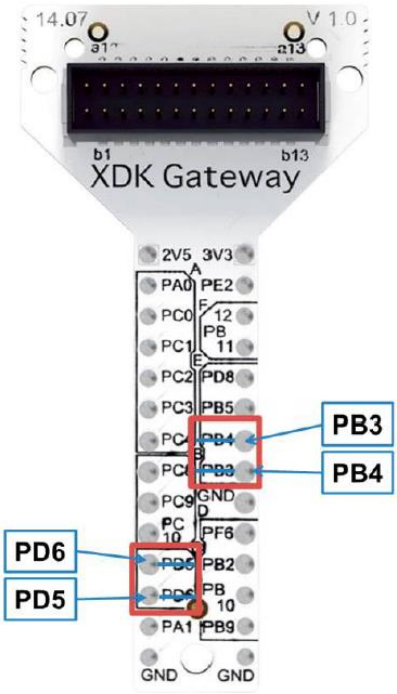
USB neboli Universal Serial Bus je univerzální sériová linka specifikovaná průmyslovým standardem. Verze USB 2.0 má maximální rychlost 480 Mbit/s. [9]

Vývojová deska Bosch XDK 110 obsahuje konektor Micro-USB pro připojení k PC pomocí kabelu dodávaného spolu se zařízením. Toto USB 2.0 může sloužit k přenosu dat, nabíjení či programování. Jak udává výrobce [1], aby nedošlo k poškození zařízení, je třeba používat jen certifikované USB konektory s kabely nepřesahující délku 3 m.

### 1.2.2 Rozšiřující deska s piny

V balení je také obsažena rozšiřující deska nazvaná „XDK Gateway“ s 26 piny umožňující jednoduše implementovat dodatečné funkce.

Při použití je třeba brát v potaz že piny pracují na napěťové úrovni 2.5 V a je dostupných několik nastavení velikosti proudu (0.5 mA, 2 mA, 6 mA, což je výchozí hodnota a 20 mA). Všechny piny sice zvládnou jednotlivě pracovat s úrovní 20 mA, je zde však omezení maximální velikosti proudu (50 mA), které může rozšiřující deska dohromady dodávat či přijímat a je tedy doporučeno používat výchozí hodnotu 6 mA. Mezi další užitečné informace může patřit, že napájecí linka 3V3 nemusí bez použití SD karty fungovat, je tedy doporučeno použít příkaz „sdc_init“. Piny jsou chráněny proti elektrostatickému výboji až do velikosti 4kV což zahrnuje sériový odpor o velikosti 40 ohmů. Důležité pak je zmínit výrobní chybu ve verzi 1.0, kde jsou mezi sebou prohozeny piny označené PB3 a PB4 a stejně tak piny PD6 a PD5 jak je ukázáno na obrázku 1.8. [1]




Obrázek 1.9 XDK Gateway [1]

### 1.2.3 JTAG

Vývojový kit může být programován buď přes USB, nebo je možnost použít i rozhraní se standardem JTAG. V takovém případě musí být dodrženo rozložení pinů, jaké je uvedeno v tabulce v [1] na straně 53.

### 1.2.4 Bluetooth

Bluetooth je standart pro bezdrátovou komunikaci mezi zařízeními vyvinut a publikován v roce 1999 a v roce 2002 přijat za standard IEEE 802.15.1. Dělí se na několik verzí, kdy verze 1 až 3 jsou označovány jako „Bluetooth Classic“ a dále novější verze 4 a 5. Tyto novější verze můžou být také v provedení BLE (Bluetooth Low Energy), tedy jak název napovídá, Bluetooth se sníženou spotřebou. Běžně používané frekvenční pásmo je 2.4 GHz a dosah se pohybuje v závislosti na verzi, nejčastěji mezi 50 a 200 metry. V závislosti na verzi se také pohybuje hodnota přenosové rychlosti od 1 Mb/s po 3 Mb/s. [10]

Použitá verze standardu v tomto zařízení je Bluetooth 4.0 BLE. Jak je možné zjistit v odstavci výše, jedná se o verzi se sníženou spotřebou, což ale také znamená, že přenosová rychlost se pohybuje na spodní hranici udávaného rozsahu, tedy 1 Mb/s. Praktický dosah je měřen v [11]. V tomto článku je testováno zařízení s různými anténami a vysílacím výkonem v průmyslovém prostředí a výsledek je dosah až 100 m. V případě hardwaru XDK 110 použitém v této práci je možné odhadnout přibližný dosah pomocí oficiálního nástroje od Bluetooth [12], do kterého se zadají kromě prostředí i parametry vysílacího zařízení zjištěné v [1] na straně 17, kde je mimo jiné uvedeno, že lze vysílací výkon programově nastavit až na hodnotu +3 dBm. Při této hodnotě je pak ve venkovním prostředí teoretický dosah mezi 68 m a 96 m. Při změně na prostředí kancelářské budovy, tedy prostředí s velkým množstvím překážek a pravděpodobně i jiného rušení, se teoretický dosah rapidně sníží na rozmezí od 16 m do 21 m.

### 1.2.5 WLAN

WLAN neboli „Wireless Local Area Network“ je síťovým standardem použitým u tohoto zařízení, též známým pod názvem Wi-Fi. Specifikace standardu 802.11 byla přijata IEEE (Institute of Electrical and Electronics Engineers) v roce 1997 [13]. Opět se dělí na několik verzí, jak je naznačeno v tabulce 2.11. V této tabulce je uvedeno jen pár základních verzí, ovšem obsahuje všechny, které je vhodné znát pro potřebu této práce.

V Bosch XDK 110 jsou použity standardy 802.11 b/g. Tyto dva standardy jsou navzájem kompatibilní, ovšem rychlost se pak při vzájemném použití snižuje dle pomalejšího z nich. Při standardu 802.11b je rychlost až 11 Mb/s a pro standard 802.11g je rychlost až 54 Mb/s. Oba disponují dosahem přibližně 140 m, který opět závisí na prostředí, podobně jako bylo zmíněno u Bluetooth v předchozí podkapitole a šířka pásma je také 2.4 GHz. [14]

Tabulka 1.1 Přehled standardů 802.11 [14]

| Standard | Frekvence [GHz] | Rychlost přenosu [Mb/s] | Dosah [m] |
|---|---|---|---|
| 802.11a | 5 | 54 | 120 |
| 802.11b | 2,4 | 11 | 140 |
| 802.11g | 2,4 | 54 | 140 |
| 802.11n | 2,4 / 5 | 248 | 250 |

Hodnoty rychlosti přenosu uvedené výše v tabulce 2.1 platí pro venkovní prostředí jen s několika překážkami [14].

### 1.2.6 LoRa

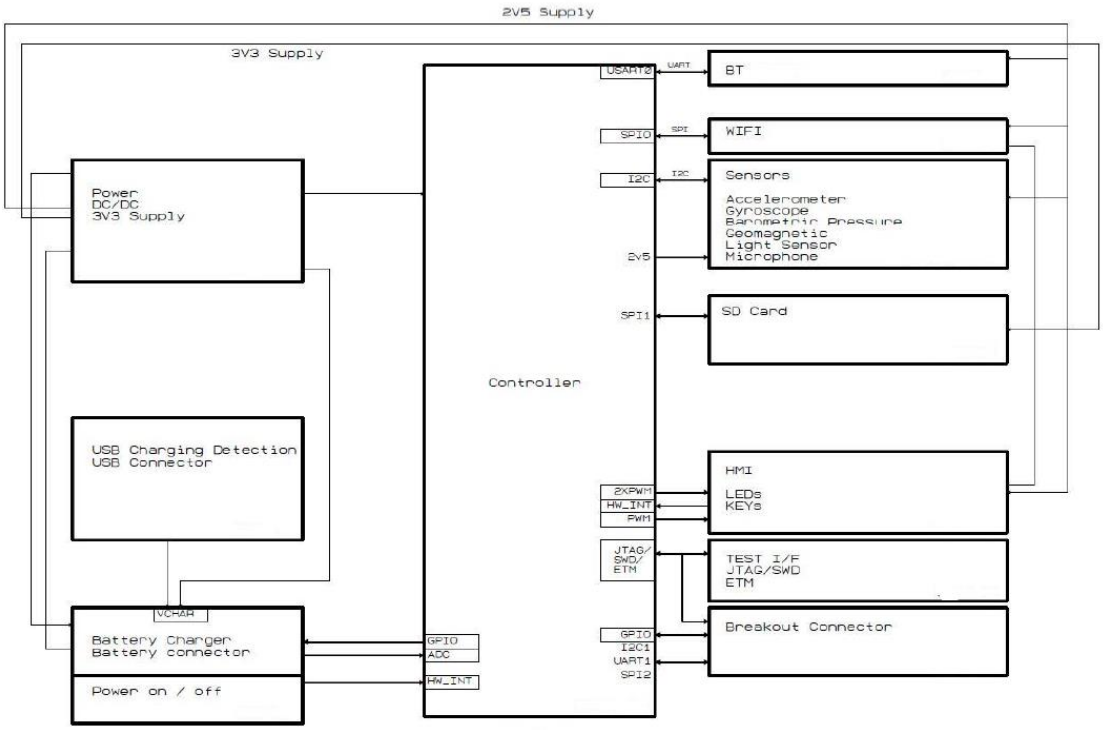
LoRa, zkratka pro „Long Range“, je technologie modulace odvozená od rozprostřeného spektra. Je to jedno z dalších možných řešení bezdrátové komunikace vedle nejrozšířenějších Bluetooth a WLAN. Hlavními výhodami této technologie je teoreticky dobrá odolnost proti rušení, větší dosah oproti předešle zmíněným řešením a zcela jistě také nízká spotřeba energie. Mezi nevýhody patří, v porovnání s ostatními, relativně nízký datový tok a je tak vhodná spíše pro občasnou komunikaci s menším objemem dat než pro nepřetržité přenášení většího objemu [15].

Technologie se skládá z výše zmíněné fyzické vrstvy LoRa a dále z vyšších vrstev LoRaWAN (Long Range Wide Area Network). Jsou využívána pásma o frekvencích 169MHz, 433 MHz a 868 MHz pro Evropu a 915 MHz pro severní Ameriku. [16]

Pro vývojové zařízení Bosch XDK 110 je také dostupná LoRa v případě dokoupení rozšiřujícího modulu, který doplní zařízení o tuto technologii. Tento modul je vybaven ochranou IP 30, jeho napájecí napětí je 2,5 V a frekvenční pásmo je 868.0-868.6 MHz.

Výrobce uvádí, že se jedná o řešení se zabezpečenou komunikací, která umožňuje průniky skrz struktury jako například budovy a také s možným dosahem až 40 km [16].


Obrázek 1.10 LoRa modul s Bosch XDK [11]




Obrázek 1.11 Blokové schéma XDK 110 [1]

# 2. VÝBĚR PROSTŘEDKŮ PRO KOMUNIKACI A IMPLEMENTACI

V této kapitole je popsán výběr prostředků, které budou použity pro návrh a následnou realizaci řešení. Ze zadání je již dané použití PC s LabVIEW. Zde bude rozebrán výběr vhodné bezdrátové komunikace, protokolu této komunikace a programovacího jazyka pro implementaci firmwaru XDK.

Při výběru bylo přihlíženo i na možné použití v IoT aplikacích. Například použití při diagnostice, kdy se zařízení připevní ke zkoumanému stroji a bezdrátově se v LABview budou sbírat data z akcelerometru nebo senzoru hluku.

## 2.1 Výběr bezdrátové komunikace

Jedním z bodů zadání je, že zařízení by mělo komunikovat bezdrátově. Vzhledem k tomu, že vývojová deska nabízí hned několik způsobů tohoto druhu komunikace, jako Bluetooth, LoRa nebo WLAN, je třeba učinit výběr na základě požadavků, zejména propustnosti.

### 2.1.1 Požadavek na propustnost

Při výběru způsobu bezdrátové komunikace je důležité zohlednit, jakou je potřeba mít propustnost. Z dostupných informací o snímačích, respektive z frekvence vzorkování a digitálního rozlišení, je možné určit přibližný teoretický maximální datový tok ze snímačů a jejich součtem i z celého zařízení. Dle výrobce [17] je možné měřit všemi senzory najednou, a tedy mohou být všechny rychlosti přenosu sečteny do jediné hodnoty propustnosti. Všechny hodnoty včetně výsledků jsou uvedeny v Tabulka 2.1.

V Tabulka 2.1 jsou uvedeny všechny snímače. Některé snímače se v tabulce opakují, a to z důvodu, kdy je v jednom pouzdře přítomno více snímačů, nebo když mají jednotlivé osy různé parametry. Senzor hluku je v dokumentaci uveden jako analogový a jako rozlišení bylo tedy použito maximální rozlišení ADC převodníku zařízení. To bylo zjištěno v knihovnách dostupných ve vývojovém prostředí, kde je maximální nastavitelné rozlišení převodníku 12 bitů. Maximální nastavitelná frekvence, s jakou lze získávat data ze snímače je 175 Hz (nastavení snímačů jsou popsány později v kapitole 4.5).

Tabulka 2.1 Datový tok

| Snímač | Frekvence vzorkování [Hz] | Digitální rozlišení [bit] | Počet os [-] | Propustnost [b/s] |
|---|---|---|---|---|
| BMA 280 | 2000 | 14 | 3 | 84000 |
| BMG 160 | 2000 | 16 | 3 | 96000 |
| BMM 150 (X,Y) | 300 | 13 | 2 | 7800 |
| BMM 150 (Z) | 300 | 14 | 1 | 4200 |
| BME 280 (STe) | 182 | 20 | | 3640 |
| BME 280 (STl) | 182 | 20 | | 3640 |
| BME 280 (SVl) | 182 | 16 | | 2912 |
| MAX 44009 | 10 | 22 | | 220 |
| BMI 160 (Akc) | 1600 | 16 | 3 | 76800 |
| BMI 160 (Gyr) | 3200 | 16 | 3 | 153600 |
| AKU 340 | 175 | 12 | | 2100 |

X,Y,Z – Označení os
STe – Senzor teploty
STl – Senzor tlaku
SVl – Senzor vlhkosti
Akc – Akcelerometr
Gyr - Gyroskop

Výpočet byl proveden dle vzorce

$$DF = f \cdot D \cdot N \,, \tag{2.1}$$

Kde DF je datový tok (Data Flow), f je vzorkovací frekvence, D je digitální rozlišení a N je počet os. Pro výpočet byly použity maximální hodnoty těchto parametrů a celkový teoretický datový tok by pak byl dán součtem dílčích hodnot v posledním sloupci Tabulka 2.1, který vychází jako 434.912 kb/s.

### 2.1.2 Porovnání propustností

V tabulce 2.2 jsou porovnávány propustnosti. V této kategorii dominuje WLAN s teoretickým datovým tokem až 54 Mb/s, následuje Bluetooth s 1 Mb/s a nejpomalejší je LoRaWAN s řádově stovkami až tisíci b/s. Dle zjištění z předchozí podkapitoly 3.1.1 by při datovém toku 434.912 kb/s nemusela LoRaWAN dostačovat, kdežto ostatní protokoly by měly s propustností stačit, dokonce i s menší či větší rezervou.

Tabulka 2.2 Datový tok

| Rozhraní | Datový tok |
|---|---|
| WLAN | 54 Mb/s [14] |
| Bluetooth | 1 Mb/s [10] |
| LoRaWAN | Stovky až tisíce b/s |

### 2.1.3 Porovnání spotřeby

Z hlediska spotřeby baterie se jeví technologie LoRa jako nejvhodnější, jak je patrné z Tabulka 2.3, kde jsou porovnány spotřeby jednotlivých řešení bezdrátové komunikace. Zde hodnoty ukazují, že BLE a LoRa mají podobnou spotřebu při několika uvedených módech, tabulka je však spíše orientační a závisí na použitém zařízení a dle závěru v [18] nebylo zařízení využívající LoRa technologii zcela optimalizováno. Lze si ale odnést že LoRa a BLE mají podobně malou spotřebu, zatímco u WLAN je spotřeba výrazně vyšší.

Tabulka 2.3 Spotřeba energie

| | Bluetooth [18] | WLAN [18] | LoRa [19] |
|---|---|---|---|
| IEEE Spec | 802.15.1 | 802.11 b | |
| Awake mode | 35 mA | 245 mA | 22 mA |
| Transmitting mode | 39 mA | 251 mA | 83 mA |
| Receiving mode | 37 mA | 248 mA | 35 mA |

### 2.1.4 Porovnání dosahu

Při pohledu na dosah je zcela bezkonkurenčně dominující LoRaWAN se vzdáleností až 40 km. Následuje WLAN s dosahem okolo 140 metrů a BLE s dosahem pro tento typ zařízení 50 m, jak je uvedeno v rubrice XDK110 v kapitole BLE dostupné na [20] .

### 2.1.5 Porovnání dalších parametrů

Ve výběru je nutno vzít v potaz i několik dalších parametrů. Jedním z nich je nutnost dokoupení rozšiřujícího LoRa modulu, aby bylo možné tuto technologii používat, s nezanedbatelnou pořizovací cenou 140 euro, čímž ji lze tímto nejspíše zcela vyřadit z další úvahy. Vzhledem k tomu, že zařízení bude muset komunikovat s LabVIEW, je třeba také zmínit že tento software nativně nepodporuje BLE.

### 2.1.6 Závěr volby bezdrátové komunikace

Vlastnosti jednotlivých způsobů komunikací jsou blíže popsány v kapitolách 2.1.2 až 2.1.5, kde jsou mezi sebou porovnány. Z této rešerše lze vyvodit jaká technologie bude tedy pro tento projekt nejvhodnější.

Výsledná LabVIEW aplikace by měla umět bez problému rychle vyčíst data, takže se dá tvrdit, že vyšší datový tok je přednější než spotřeba energie a tím pádem se jako použití bezdrátové komunikace jeví nejlépe WLAN.

## 2.2 Výběr komunikačního protokolu

Ve vývojovém prostředí XDK-Workbench je dostupných několik knihoven pro protokoly a další způsoby bezdrátové komunikace. Protokolů je velká řada a zde bylo převážně voleno z takových, ke kterým jsou dostupné knihovny z výše zmíněného vývojového prostředí pro usnadnění práce při zprovozňování.

Prvním protokolem, pro který je dostupná knihovna, je HTTP (Hyper Transfer Protocol). Jedná se o protokol aplikační vrstvy určený pro komunikaci převážně s webovými servery typu požadavek/odpověď. [20]

Dalším protokolem je UDP (User Data Protocol). Poskytuje postup pro odesílání zpráv s minimem protokolárních mechanismů a negarantuje přijetí zprávy. Většina internetových multimediálních služeb pracujících v reálném čase používá UDP, protože nepřináší zpoždění při retransmisi. Používá cyklickou redundanční kontrolu k ověření integrity paketů, pokud v některé části odhalí chybu, zahodí celý paket a příjemce pak dostává buď dokonalý, nebo zcela ztracený paket. [21] [22]

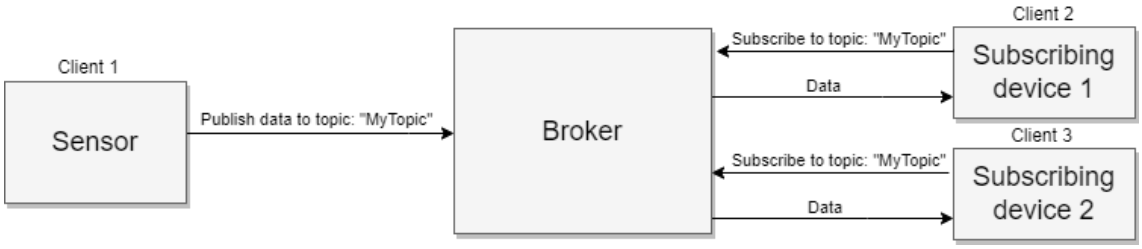
Posledním zde uvedeným protokolem je MQTT (Message Queing Telemetry Transport). Reprezentuje M2M (Machine to Machine) protokoly založených na publish/subscribe komunikačním vzorci. Účelem tohoto protokolu je použití v zařízeních s omezenými paměťovými možnostmi nebo omezeným procesním výkonem. [23]

K vytvoření firmwaru bude použit jazyk Eclipse Mita, jak je zmíněno později, v kterém jsou implementovány funkce pro protokoly MQTT a HTTP. Vzhledem k tomuto faktu lze z uvažování vyřadit protokol UDP a celkově se tedy jako nejvhodnější pro komunikaci jeví protokol MQTT, který bude následně v práci použit.

### 2.2.1 Popis protokolu MQTT

Protokol MQTT je standardně používán pro IoT (Internet of Things) aplikace. Je navržen, aby měl velmi malou zátěž a je vhodný pro zařízení s malou datovou propustností Vzhledem k tomu, že klient je jednoduchý a vyžaduje jen minimální zdroje, je možné ho použít i na malých mikrokontrolerech. V případech, kdy je nutné zajistit spolehlivé doručení zprávy, nebo je potřeba mít co nejúspornější komunikaci, je možné nastavit 3 stupně QoS (Quality of Service). Při nastavení 0 znamená, že je zpráva odeslána, respektive přijata nanejvýš jednou. Při QoS rovném 1 znamená, že zpráva je přijata

nejméně jednou a při nastavení na číslo 2 je zpráva přijata přesně jedenkrát. Jednou z dalších vlastností je i podpora pro nespolehlivé sítě, které se vyskytují často při použití v IoT aplikacích. Připojení se udržuje trvale a je tak zkrácen čas pro připojení klienta a brokeru. Součástí jsou i bezpečnostní prvky pro šifrování komunikace nebo ověřování klienta. [25]




Obrázek 2.1 Blokové schéma MQTT komunikace

Protokol funguje na principu publish/subscribe modelu. Základními prvky jsou klienti a broker zprostředkovávající komunikaci, který je taktéž někdy označován jako server. Protokol je typu příkaz/odpověď a každý příkaz je potvrzován. Potvrzovací příkazy jsou například „connack“ pro potvrzení připojení, „suback“ pro potvrzení odebírání a „puback“ pro potvrzení publikování zpráv. Chce-li klient zahájit komunikaci, musí se nejprve připojit k brokeru a poté může teprve publikovat nebo odebírat data. Během připojení také v určitém intervalu (běžně 60 sekund) klient odesílá tzv. „keepalive“ zprávu, která dává vědět brokeru, že daný klient je stále aktivní. Dalším pojmem v MQTT komunikaci je topik. Jedná se v podstatě o téma, na které chce daný klient zprávu odesílat nebo přijímat. Protokol tak nepoužívá adresy a zprávy nejsou zasílány pro konkrétní klienty, ale jsou publikovány na daný topik a odesílány všem kteří jej odebírají, pokud samozřejmě mají oprávnění. Hlavním úkolem brokeru je filtrovat zprávy na základě topiku a následně je distribuovat odběratelům. Je možné také zmínit, že každý klient může zprávy, jak publikovat, tak odebírat. [25]

Příklad komunikace je uveden na Obrázek 2.1, kde je znázorněno blokové schéma. Klient číslo 1 znázorňuje senzor, který publikuje měřená data na topik s názvem „MyTopic“. Klienti číslo 2 a 3 jsou zde v roli odběratelů, jedná se o odebírající zařízení a je možné si je představit pro příklad jako PC a chytrý telefon. Oba klienti odebírají na téma „MyTopic“ a tudíž jsou odesílaná data ze snímače pomocí brokeru posílána právě na tato zařízení. Tímto způsobem je možné například získávat data o teplotě a ta ukládat na PC, nebo je sledovat z telefonu.

## 2.3 Výběr jazyka pro firmware

První možností, jak programovat firmware pro XDK 110, je použití jazyka C. Co tento programovací jazyk nabízí zřejmě není třeba uvádět, ale je jistě výhodou, že jsou pro něj dostupné knihovny pro používané zařízení v rámci vývojového prostředí XDK-Workbench.

Další možností volby programovacího jazyka je Eclipse Mita [24], který nabízí hned několik výhod. Je zaměřený na programování IoT (Intenet od Things) aplikací a může tak usnadnit práci, zvýšit produktivitu a zpřehlednit či zkrátit kód. Je pro něj samozřejmě dostupný balík speciálně pro platformu Bosch XDK 110, kde lze s výhodou použít předpřipravené funkce pro ovládání a nastavování zařízení. Velkou výhodou je také usnadnění práce s RTOS (Real-time operating system), tedy operační systém reálného času, který je dostupný v rámci dříve zmíněného vývojového prostředí.

Zmíněný FreeRTOS je operační systém, který lze strukturovat jako soubor nezávislých úloh. Každá úloha je spuštěna nezávisle na jiných, ovšem vždy může být spuštěna jen jedna ve stejném okamžiku. Plánovač pak rozhoduje o tom, která spuštěna bude. Ten může tak opakovaně dané úlohy spouštět a zastavovat. Vzhledem k tomu, že systém je často používán pro reagování na reálné události, které jsou časově omezené, musí RTOS umět zareagovat včas. Proto jsou jednotlivým úlohám nastavovány priority, aby plánovač věděl, která úloha má přednost před jinou. Používaný FreeRTOS je blíže popsán v rubrice „FreeRTOS“ na webu [17].

Vzhledem ke zmíněným vlastnostem byl tento jazyk zvolen pro vytvoření firmwaru a bude tedy použit pro tuto práci a také bude blíže popsán v následující podkapitole.

### 2.3.1 Popis Eclipse Mita

Jak se lze více dočíst v předchozí kapitole, pro firmware byl zvolen programovací jazyk Eclipse Mita, který je zaměřen na IoT aplikace. V této kapitole bude zmíněný jazyk přiblížen z hlediska dostupných možností a funkcionalit.

Struktura programu se skládá ze dvou základních částí. První částí je nastavení senzorů, připojení a podobně a další částí jsou eventy, ve kterých se provádějí úkony jako čtení snímačů či odesílání dat. Klíčovým slovem „setup“ následovaným vlastním označením a konkrétním názvem nastavované periferie lze vytvořit strukturu, uvnitř které je možné danou periferii nastavit, jak je například uvedeno níže.

```
setup wifi : WLAN{
      ssid = 'MySSID';
      authentication = Personal(psk='MyPSK');
      ipConfiguration = Dhcp();
}
```

Zmíněné eventy jsou funkce vyvolávané periodicky či určitou událostí. Klíčovým slovem je „every“ a následuje buď délka periody včetně jednotek, či konkrétní událost jako stisknutí tlačítka, překročení prahové hodnoty na akcelerometru a další. Opět zde následuje příklad použití.

```
every 5 seconds {
	led.orange.write(true);
}
```

V jazyce je možné použít dva základní typy proměnných. První typ je klasická proměnná, do které je možné zapisovat a měnit hodnoty a deklaruje se klíčovým slovem „var“. Druhá je neměnná a nabývá hodnoty která se zadá při deklaraci a následně nemůže být změněna a používá se klíčové slovo „let“. Datové typy, kterých proměnné mohou nabývat jsou již podobné jako v jazyce C, například integer 8, 16 či 32bitový a stejně tak bezznaménkový integer. Další jsou double, float nebo boolean. Ze složitějších datových typů je dostupná například struktura, enum a je možné používat reference. Na rozdíl od jazyka C, kde pro string je nutné inicializovat pole typu char, zde se nachází přímo proměnná string, ovšem nelze ji indexovat.

Operace jsou taktéž obdobné s jazykem C, a to sčítání, odčítání, násobení, dělení, bitový posun, logická i bitová negace a podobně.

Stejně tak jsou na tom i kontrolní struktury a je možné tedy používat if-else, while, do-while a for.

Jako v mnoha jiných jazycích, i tady je možné použití pole. Tento typ struktury se používá běžným způsobem, podobně jako v jazyce C, tedy je možné přistupovat k jednotlivým prvkům a podobně. Deklarace pole je uvedena na příkladu níže.

```
var MyArray : array<uint8> = [1,2,3];
```

Dalším prvkem je použití funkcí. Ty mohou být tvořeny a používány ovšem s lehce odlišnou syntaxí než v jazyce C. Nejdříve je použito klíčové slovo „fn“ nebo „function“, za kterým následuje název dané funkce. Poté následují jednoduché závorky, uvnitř kterých jsou parametry funkce a za závorkou je uveden návratový typ. Příklady vytvoření funkcí jsou uvedeny níže. Pro funkce je možné použít i tzv. polymorfismus, tedy použití více funkcí se stejným názvem ale s jinými parametry.

```
fn helloWorld() {
	return "hello world";
}

function MultiplyByTwo(x : int32) : int32 {
	return x * 2;
}
```

Zajímavou funkcionalitou, který Eclipse Mita nabízí je „foreign function interface“ a ta umožňuje použití funkcí z knihoven napsaných v jazyce C. Nejdříve je nutné deklarovat hlavičku funkce která má být použita a přidat název knihovny, ve které se funkce nachází, jak je uvedeno na následujícím příkladu.

```
native unchecked fn atoi(str: string) : int32 header "string.h";
```

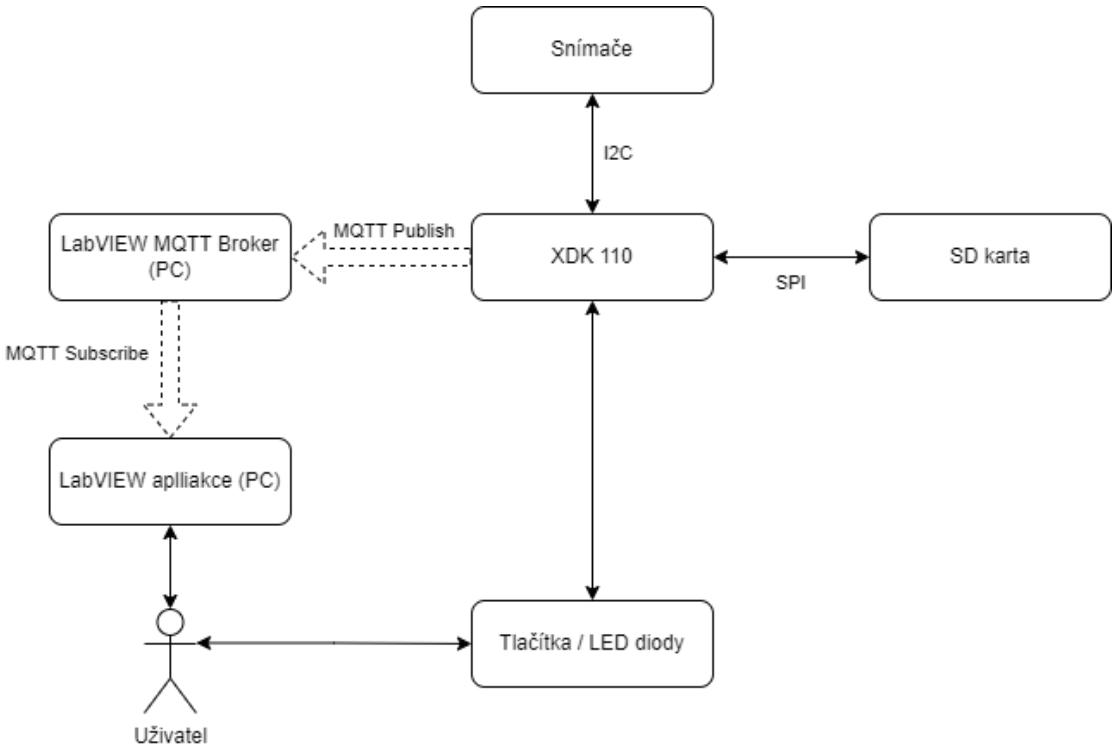
Poté je možné deklarovanou funkci běžně používat v prostředí jazyka Mita.

# 3. NÁVRH SOFTWARU

V této kapitole bude proveden návrh celkové funkčnosti firmwaru a LabVIEW aplikace a následná implementace je popsána v kapitole 4. Návrh bude také vycházet z výběrů provedených v kapitole 2.

## 3.1 Program

Výsledný software se bude skládat z firmwaru nahraném v zařízení Bosch XDK 110, který se bude starat převážně o sběr a ukládání dat ze snímačů či jejich odesílání. Druhou částí výsledného programu je LabVIEW aplikace umístěná na dalším zařízení, například PC, která bude vyčtená data přijímat, zpracovávat a následně i zobrazovat či vykreslovat.




Obrázek 3.1 Základní blokové schéma návrhu

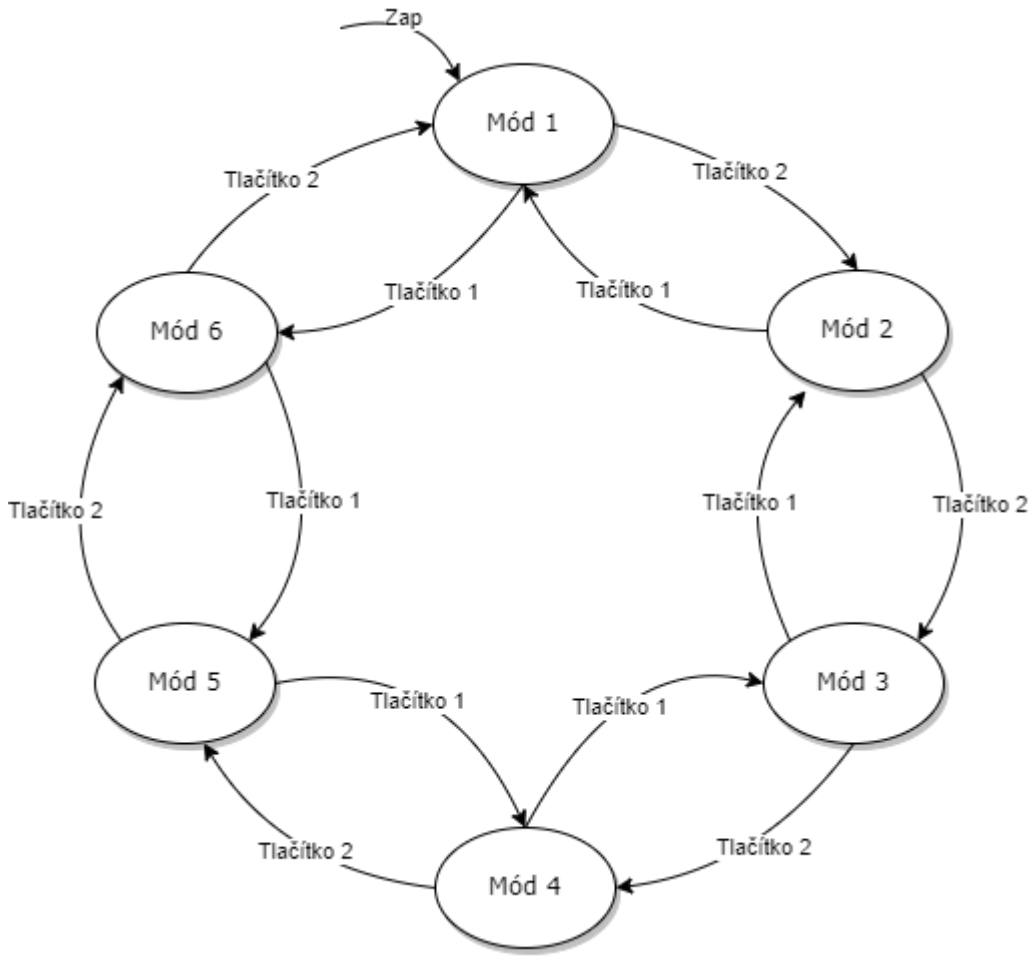
### 3.1.1 Funkce výsledného programu

Tato podkapitola se bude zabývat navržením celkové funkcionality výsledného softwaru, respektive co by měl program umět, co a jak bude možné nastavit či ovládat a podobně.

Jak bylo dříve zjištěno, v zařízení Bosch XDK 110 lze sbírat data ze všech snímačů najednou a pro bezdrátovou komunikaci byl zvolen způsob s vysokou datovou propustností, která by měla být dostatečná na zvládnutí přenášení dat ze snímačů. Na základě těchto znalostí se nabízí možnost, aby program uměl zobrazovat data získávána ze snímačů v reálném čase v aplikaci LabVIEW.

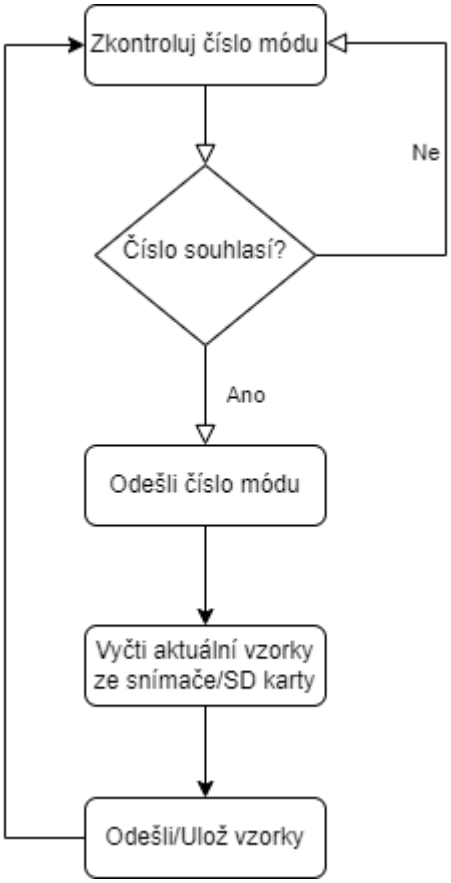
### 3.1.2 Firmware

Firmware by měl být koncipován jako několik módů, mezi kterými je možné přepínat pomocí dvou tlačítek dostupných na zařízení, jak je vyobrazeno na Obrázek 3.2. Každý jednotlivý mód by uměl vyčíst a odeslat data z některého ze snímačů, nebo je uložit na SD kartu, jak je znázorněno na Obrázek 3.3.




Obrázek 3.2 Módy firmwaru

Jak je zmíněno v kapitole 2.3.1, jazyk Eclipse Mita funguje na principu eventů a jednotlivé módy by tak byly vyvolávány periodicky a následně by se dle nastaveného módu kontrolovalo, zda se mají instrukce uvnitř provést či nikoliv. Perioda by odpovídala požadované frekvenci pro odesílání a vyčítání dat z konkrétního snímače.

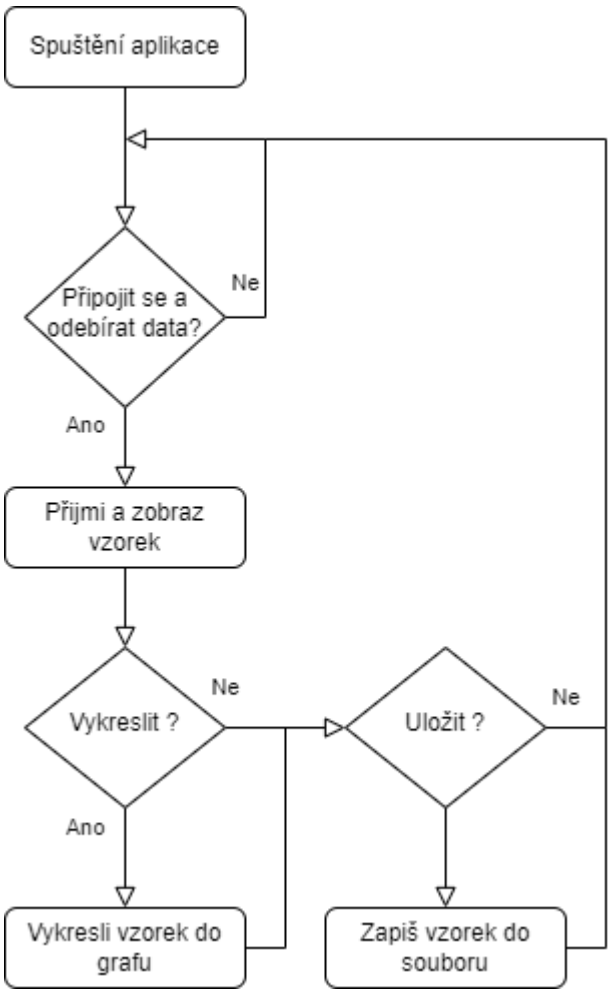


Obrázek 3.3 Obecný diagram módu

V módu jedna by byly vyčítány data z akcelerometru s vyšší frekvencí a následně v módu dva by obstarával data ze snímače okolního prostředí, tedy teplota, tlak a vlhkost a případně intenzitu okolního osvětlení s nižší frekvencí. V módu tři, který by byl vyvoláván opět s rychlejší frekvencí, by byly vyčítány osy magnetometru a podobně by to bylo v módu čtyři pro snímač hluku. Módy pět a šest by pak obstarávali ukládání na SD kartu, případně vyčítání a odesílání dat z ní.

### 3.1.3 Program v LabVIEW

V LabVIEW aplikaci by mělo být možné vyčítat veškerá data, která budou ze zařízení XDK 110 odesílána. Aplikace by měla být univerzální, tedy měla by se přizpůsobit aktuálně nastavenému módu, respektive vyčítanému snímači. Všechna přijímaná data by měla být zobrazována na front panelu. Měla by být také dostupná možnost přijímaná data ukládat do souboru nebo vykreslovat do grafu. Diagram návrhu LabVIEW aplikace je vyobrazen na Obrázek 3.4.




Obrázek 3.4 Diagram LabVIEW aplikace

# 4. IMPLEMENTACE

V předchozí kapitole byl proveden návrh výsledného softwaru. V rámci této kapitoly je popsána implementace tohoto návrhu, s průběžným testováním a následným vylepšováním či upravováním návrhu na základě získaných zkušeností z práce se zařízením. Klíčová bude především bezdrátová komunikace, případně pak vyčítání dat ze snímačů.

## 4.1 Oživení Bosch XDK 110

Po stažení a spuštění vývojového prostředí (XDK-Workbench) bylo pomocí USB zapojeno zařízení do PC. Následně bylo zapnuto spouštěcím přepínačem umístěném na boku pouzdra. Zařízení se spustilo, což indikovaly LED diody a následně bylo rozpoznáno i ve vývojovém prostředí.

V XDK-Workbench byl vybrán předpřipravený program od výrobce, který umožňoval vyčítat data ze snímačů na zařízení XDK 110 pomocí USB a následně je vypisovat v konzoli vývojového prostředí. Po vybrání programu byl pak následně pomocí tlačítka „Flash“ nahrán do zařízení a dále po nahrání i spuštěn. Po spuštění se do konzole začali periodicky vypisovat data z dostupných snímačů, dokud nebylo zařízení vypnuto, nebo z něj nebyl program vymazán.

## 4.2 Bezdrátová komunikace PC s XDK 110 přes UDP

V podkapitole 2.1 byla jako způsob bezdrátové komunikace vývojové desky s okolím zvolena WLAN. V této kapitole je uvedena rychlá možnost otestování bezdrátové komunikace přes protokol UDP.

### 4.2.1 Popis programu v XDK 110

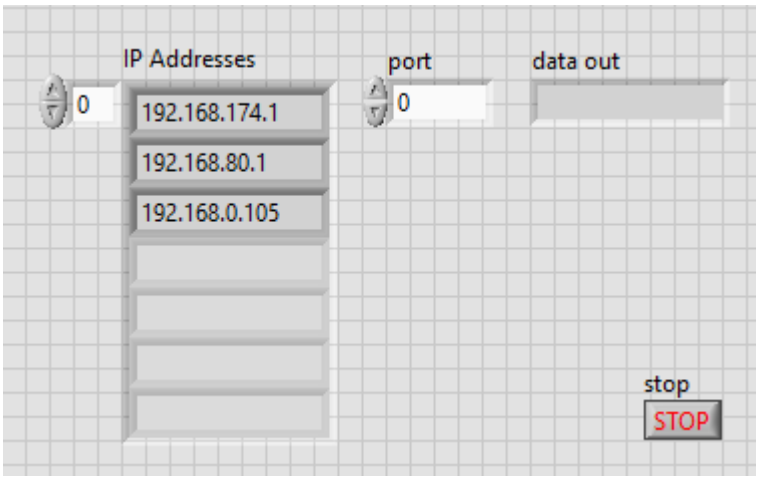
Z dostupných příkladů programů v jazyce C ve vývojovém prostředí XDK-Workbench, byl použit „SendDataOverUdp“, který se připojí k dostupné Wi-Fi síti a následně začne odesílat na specifikovaný server data (jak bylo později po zprovoznění zjištěno, program posílá cyklicky ascii znaky). Pro zprovoznění samotného programu je nutno v hlavičkovém souboru s názvem „AppController.h“ ještě doplnit několik informací, jak lze vidět na ukázce kódu v příloze A.1.

Nejprve je nutné doplnit WLAN SSID, neboli identifikátor bezdrátové sítě na místo textu „YourWifiNetwork“ a následně také přístupové heslo do této sítě, kterým je opět nahrazen text „YourWifiPassword“. Pokud se má zařízení připojit do podnikové WiFi sítě namísto osobní, je třeba ještě specifikovat uživatelské jméno a makro s názvem „IS_ENTERPRISE_WIFI_SELECTED“ je třeba nastavit jako „true“, v opačném případě je toto makro ponecháno jako „false“. Pro funkčnost už zbývá nastavit jen IP adresu serveru na který se mají odesílat data a také zadat číslo portu. V této chvíli by měl být

program schopen se po spuštění připojit k síti a začít odesílat data. V posledním řádku je i možnost nastavit vysílací frekvenci UDP v milisekundách, kde je jako výchozí hodnota nastaveno 1000 ms.

### 4.2.2 Popis LabVIEW aplikace

V programu LabVIEW byla vytvořena jednoduchá aplikace (Obrázek 4.2) na přijímání dat přes WiFi pomocí protokolu UDP. Aplikace umí zjistit i dostupné IP adresy pro usnadnění získání adresy zařízení na kterém běží LabVIEW. Ta, jak bylo zmíněno v předchozí podkapitole, musí být doplněna do programu v XDK 110. Po zjištění adresy se zadá v aplikaci port a stejně tak se vyplní i ve firmwaru pro zařízení XDK. Pokud jsou oba programy spuštěny, na front panelu (Obrázek 4.1) v LabVIEW v indikátoru „data out“ se začnou zobrazovat přijímané znaky.

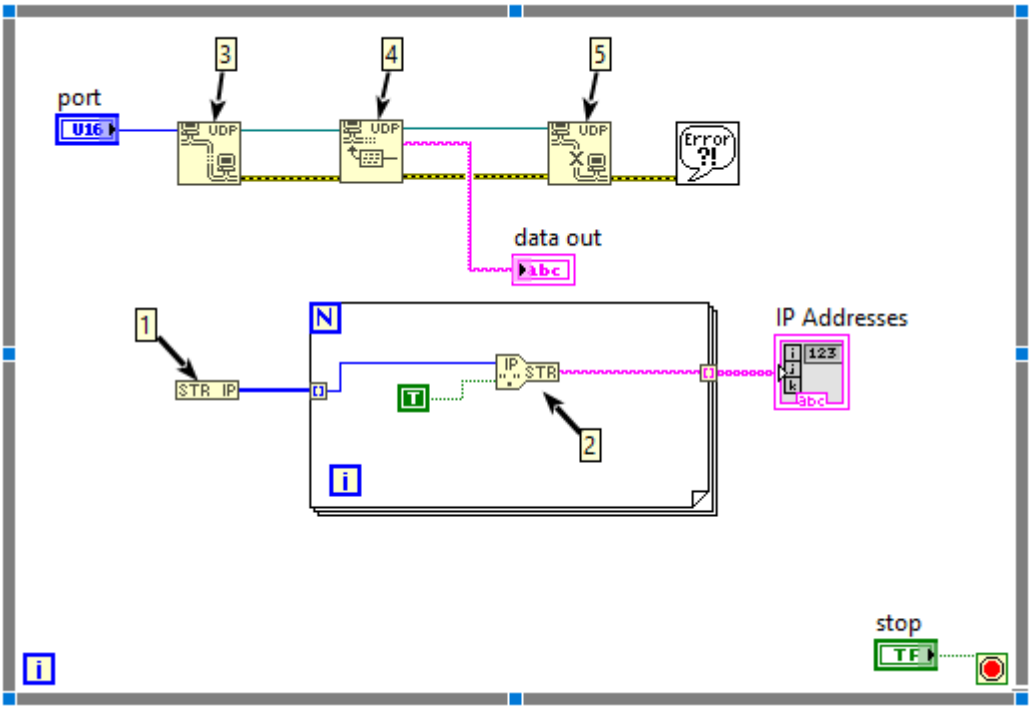



Obrázek 4.1 Front panel UDP

Blokový diagram zobrazuje Obrázek 4.2. První blok má zapojený jen vstup, na který se přivede číslo portu zadané uživatelem z front panelu. Mezi další vstupy patří také „net address“ který se připojuje ve chvíli kdy je dostupných více zařízení a je třeba přijímat data jen z jednoho. Dále je zde také vstup „service name“, který není pro tuto aplikaci důležitý a „timeout ms“, což je číselná hodnota v milisekundách, po jakou dobu se čeká, než funkce skončí s chybou (výchozí hodnota je 25 000 ms). Jako poslední je tady umístěn i vstup „error in“. Jako výstupy jsou zde „connection ID“, „port“ a „error out“.

V dalším bloku „UDP Read“ jsou vstupy „connection ID“ jenž je spojen se stejnojmenným výstupem z předchozího bloku, „max size“, který udává kolik bytů se má přečíst a nakonec „timeout ms“ a „error in“ stejné jako v předchozím případě. Výstupy „connection ID out“ a „error out“ jsou taktéž stejné jako v předešlém bloku, ovšem jsou zde ještě další výstupy jako „data out“, kde lze získat výstupní data a také „port“ a „address“ pro zjištění portu a adresy.

Posledním blokem v tomto řetězci je „UDP Close“ který má pouze dva vstupy „connection ID“ opět spojené s předešlým blokem a stejně tak i „error in“. Výstupy jsou také jen dva, a to „connection ID out“ a „error out“.




Obrázek 4.2 Aplikace UDP, části: 1) „String To IP“, 2) „IP To String“, 3) „UDP Open“, 4) „UDP Read“ a 5) „UDP Close“

## 4.3 Zprovoznění komunikace přes protokol MQTT

Jako protokol pro bezdrátovou komunikaci byl zvolen MQTT. V této kapitole bude popsáno zprovoznění tohoto způsobu bezdrátové komunikace.

### 4.3.1 Mosquitto Broker

V této podkapitole je uveden rychlý způsob, jak zprovoznit bezdrátovou komunikaci přes MQTT pomocí softwaru Eclipse Mosquittto s open source licencí dostupný z [25]. Je to snadný a nenáročný způsob, jak je možné vyčítat data ze zařízení XDK 110, pokud nebude k dispozici naprogramovaná LabVIEW aplikace.

Po instalaci je nutné vytvořit vlastní konfigurační soubor, obsahující příkazy `listener 1883` na prvním řádku, který přidá konkrétní port a `allow_anonymous true` na druhém řádku, který povolí že se na broker mohou připojit i anonymní zařízení a není tak potřeba specifikovat jméno a heslo pro připojení. V případě že je vhodné mít komunikaci zabezpečenou, je možné použít jako jednu z možností právě zmíněné jméno a heslo. Aby to bylo možné, je třeba vytvořit soubor, ve kterém bude uvedeno jméno a heslo na jednom řádku oddělené dvojtečkou. Těchto řádků může být uvedeno více dle toho, kolik je potřeba obsluhovat zabezpečených komunikací. Následně je v příkazovém řádku zadán příkaz `mosquitto_passwd -U passwordfile`, který v parametru uvedený soubor s heslem zašifruje. V konfiguračním souboru je třeba mít řádek `allow_anonymous false` a také cestu k souboru s heslem například `password_file c:\mosquitto\passwords.txt.` Konfigurační soubor musí mít příponu „conf“ a název může vypadat jako „myConfFile.conf“. Ovládání probíhá přes příkazovou řádku a broker se spouští příkazem `mosquitto -v -c myConfFile.conf`, kde se povolí „verbose“ mód a specifikuje již zmíněný konfigurační soubor, který se musí nacházet v instalačním adresáři. Zdali se broker spustil je možné zkontrolovat příkazem `netstat -a`, kde lze zkontrolovat, jestli na daném portu specifikovaném v konfiguračním souboru broker opravdu běží.

Software mosquitto umí také plnit roli výše zmíněného publishera nebo subscribera a je tak možné plnohodnotně otestovat komunikaci. Pro odebírání zpráv je zapotřebí spustit další příkazovou řádku s příkazem `mosquitto_sub -p 1883 -t test` v kterém je uveden port a také tzv. „topic“, tedy téma pro které mají být zprávy přijímány. Naopak pro publikování se do třetí příkazové řádky vloží příkaz `mosquitto_pub -h 127.0.0.1 -p 1883 -m "test message" -t test`. V příkazu je zadána IP adresa brokeru, dále port, následuje obsah zprávy a nakonec topik. Po zadání příkazu se zadaná zpráva vypíše v příkazové řádce, ve které běží subscriber.

### 4.3.2 Odesílání dat z XDK 110

V této podkapitole je popsáno zprovoznění MQTT klienta v roli publisher na samotném zařízení XDK 110. Jako první krok je třeba nastavit připojení k síti WLAN jak je ukázáno níže. Zde je třeba zadat ssid neboli název sítě a následně i heslo pro připojení. Další možností je zvolit konfiguraci IP, v tomto případě zvolena jako DHCP.

```
setup wifi : WLAN{
      ssid = 'My_ssid';
      authentication = Personal(psk='My_psk');
      ipConfiguration = Dhcp();
}
```

Následujícím krokem je nastavení protokolu MQTT. Níže je opět uvedena ukázka tohoto nastavení, kde se zadává nejdříve způsob transportu, tedy výše nakonfigurovaná WLAN. Dále také url adresa a ID, neboli název klienta pod kterým bude data odesílat. Jako příklad je zde uvedena také možnost „lastWill“, což je poslední zpráva na daný topik a s danou spolehlivostí odeslání, která bude odeslána po vynuceném odpojení. Předposlední je uvedeno zadání jména a hesla pro zabezpečenou komunikaci. Na posledním místě je součástí proměnná obsahující konkrétní topik, na kterém bude možné publikovat zprávy.

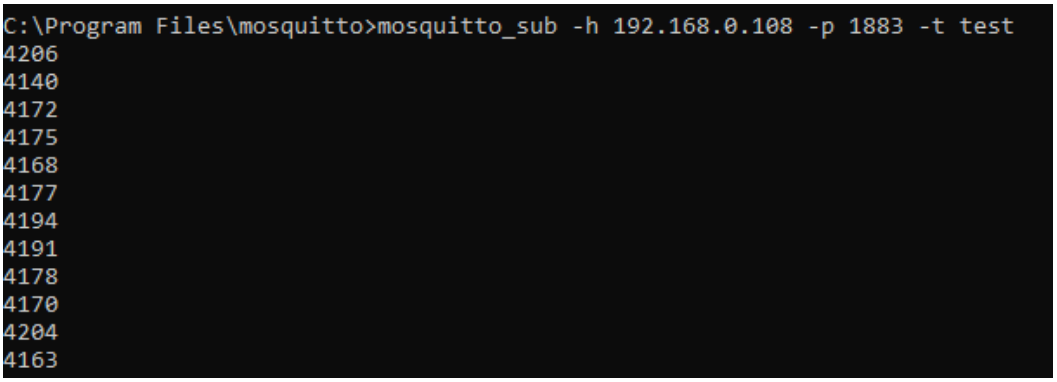
```
setup mqtt: MQTT {
  transport = wifi;
  url = "mqtt://192.168.0.108:1883";
  clientId = "XDK";
  lastWill = LastWill(topic = "test", message = "LastWill", qos = 0) ;
  authentication = Login(username ="MyUsername" , password = "MyPassword");

  var test = topic("test");
}
```

Mezi dalšími možnostmi nastavení lze zmínit také flag s názvem „cleanSession“, který udává, jestli má broker vytvořit dočasnou či trvalou relaci ve které bude uchovávat všechny zprávy pro klienta. Do prvku s názvem „keepAliveInterval“ se zadá čas v sekundách, tedy interval, během kterého se budou odesílat dva pingy do brokeru, aby obě entity navzájem věděly, že jsou stále aktivní.

Poté co je WLAN a MQTT nastaveno, je možné publikovat data, jak ukazuje následující příklad. Zde jsou každých 100 ms vyčtena data z akcelerometru a ihned odeslána.

```
every 100 milliseconds{
  backend.test.write(`${accelerometer.magnitude.read()}`);
}
```

Odesílání si je možno ověřit pomocí mosquitto subscriber jako na Obrázek 4.3. Jelikož jsou data přijímána z jiného zařízení, je nutné při spouštění mosquitto subscribera zadat i adresu, což je možné na zmíněném obrázku vidět.




Obrázek 4.3 Přijímání dat z XDK 110

Nutností je zde také zmínit problémy později zjištěné při testování bezdrátové komunikace. Výše uvedená funkce pro odesílání dat ze zařízení je nejspíše časově náročnější, a proto při vyšších frekvencích dochází po několika jednotkách či desítkách minut provozu pravděpodobně ke kumulování dat ve vyrovnávací paměti nebo podobnému jevu, následkem čehož je přerušen plynulý tok dat, který lze obnovit pozastavením programu nebo restartováním zařízení. Nastavená frekvence při testování byla maximální, tedy 1 kHz a v této frekvenci byla data, jak vyčítána z akcelerometru, tak odesílána ze zařízení. V LabVIEW aplikaci se pak po výše uvedené době data začala zobrazovat mnohem pomaleji a trhaně patrně přiškrcením toku dat v některém bodě. Při dalším testování byla frekvence vyčítání a odesílání dat postupně snižována a zařízení bylo pokaždé ponecháno desítky minut v provozu. Pokud byl plynulý tok dat opět přerušen, byla frekvence znovu snížena a zařízení restartováno. Nejvyšší frekvence, při které k tomuto jevu již vůbec nedochází byla určena jako 20 Hz. Aby byla vyloučena jako příčina problému LabVIEW aplikace, bylo přijímání dat testováno i pomocí softwaru Mosquitto, kde docházelo ke stejnému jevu. Pokud bylo zařízení připojeno pomocí sériové linky do PC a data byla na místo bezdrátového odesílání vypisována přes linku do konzole ve vývojovém prostředí, daný jev nebyl zpozorován a příčina je tak s největší pravděpodobností ve zmíněném odesílání přes MQTT. V další implementaci byla jako nejvyšší frekvence pro bezdrátové odesílání dat používána zjištěná frekvence 20 Hz.

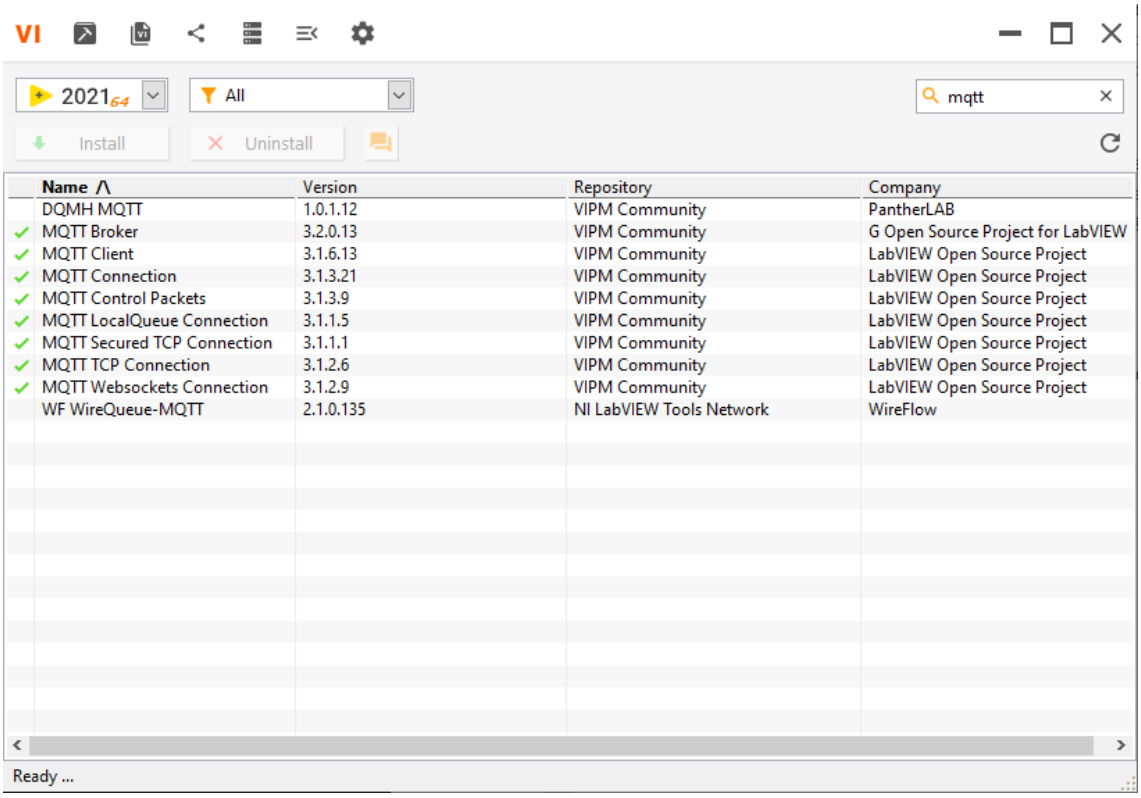
### 4.3.1 Přijímání dat do XDK 110

V použitém jazyce Mita dosud není implementováno přijímání dat přes protokol MQTT, jak uvádí samotný výrobce [26] v kapitole 3.5 „Subscribing to a topic“. Stejně tak není implementováno příjímání dat ani v dalším a posledním dostupném protokolu v jazyce Mita, a to http. Poslední možností, jak přijímat data pomocí bezdrátové komunikace se nabízí BLE, ovšem i u tohoto způsobu výrobce uvádí že funkcionalita pro přijímání dat není dosud implementována.

Jednou z teoretických možností, jak toto obejít je funkcionalita „Foreign Function Interface“, kterou zvolený jazyk nabízí. Jedná se o možnost použití funkcí, obsažených v knihovnách napsaných v jazyce C, přímo v Mita aplikaci. Toto je možné, jelikož se kód aplikace vždy následně překládá do jazyka C. Pokusy o propojení knihovny MQTT však byly kvůli její komplexnosti i přes vynaložené úsilí neúspěšné. Klient reprezentovaný zařízením XDK 110 se při žádném z pokusů o použití této knihovny nedokázal připojit k brokeru a bude tak nutné se dále obejít pouze s jednosměrnou komunikací.

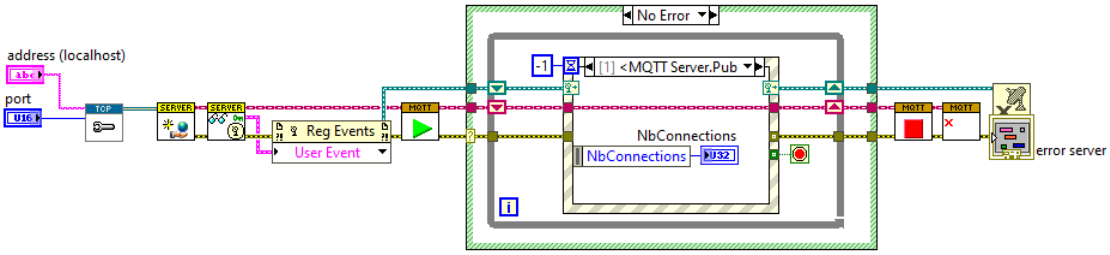
### 4.3.2 Protokol MQTT v LabVIEW

Aby bylo možné používat MQTT protokol v LabVIEW, je nutné do tohoto programu doinstalovat vhodné pluginy, jak naznačuje screenshot na Obrázek 4.4 zachycený v programu pro správu doplňků VI Package Manager (VIPM).



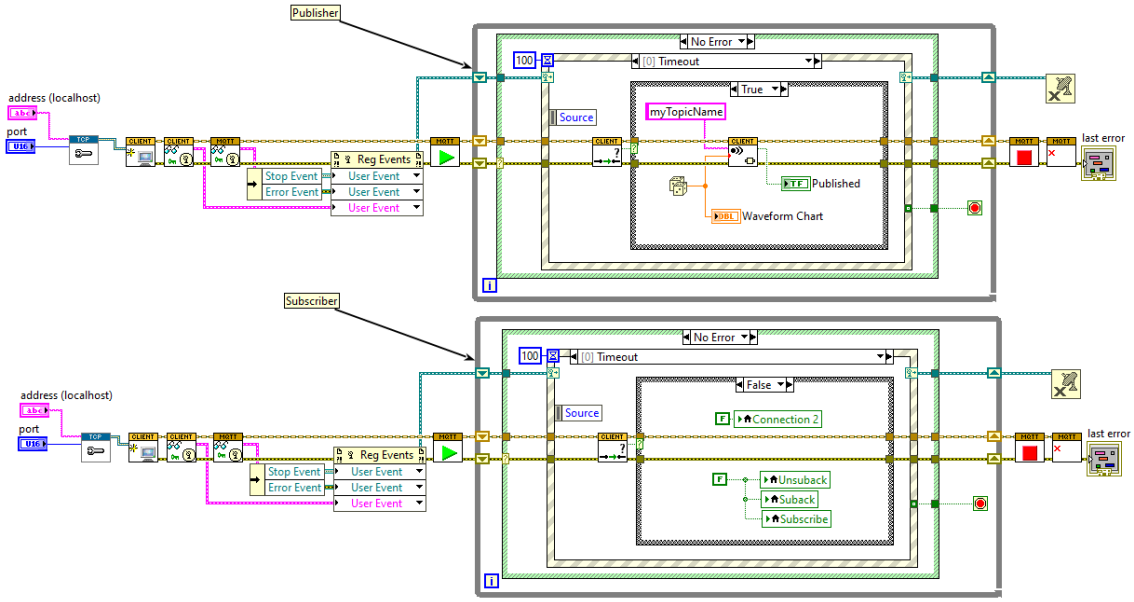

Obrázek 4.4 MQTT VIPM

Po nainstalování lze pak v knihovně nalézt předpřipravené VI aplikace týkající se již zmíněného protokolu jako subscriber, publisher či broker.



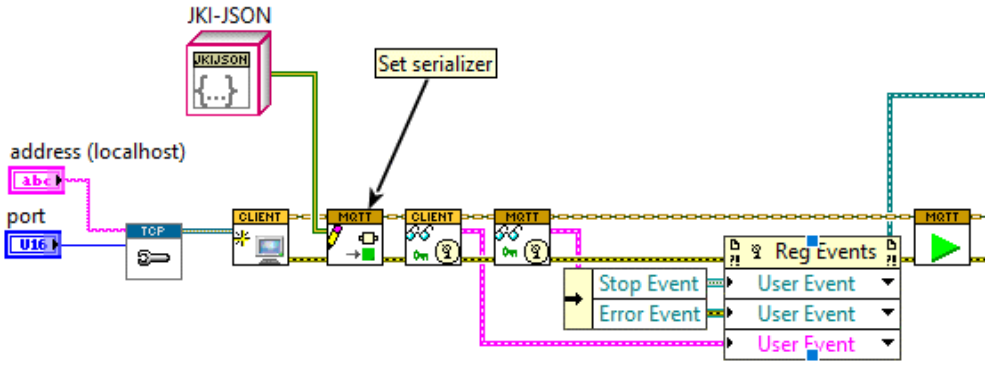

Obrázek 4.5 LabVIEW Broker

Na obrázku 3.8 je již zmíněný broker, do kterého je nutné na začátek doplnit blok „MQTT_TCP“ a k němu přidat adresu i port, aby bylo možné broker použít.



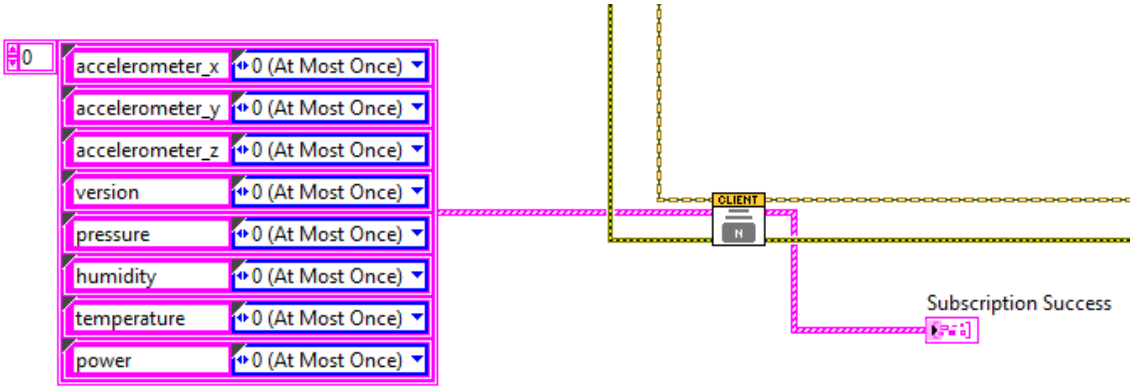

Obrázek 4.6 Příklad LabVIEW Publisher a Subscriber z použitého pluginu

Pro převedení na tvar dat vhodný pro LabVIEW je třeba použít „serializer“. Na tento účel byl přidán další plugin s názvem „OpenSerializer JKI JSON“, který obsahuje potřebné nástroje a konkrétně blok „JKI-JSON Serializer“. Tento blok se pak zapojí na vstup bloku z knihovny MQTT „Set Serializer“, který se přidá do řetězce mezi bloky „Create MQTT client“ a „Read public events“, jak je ukázáno na Obrázek 4.7.




Obrázek 4.7 LabVIEW Subscriber a Serializer

Pro vytvoření klienta pro odebírání zpráv byl použit příklad klienta dostupného v rámci knihovny, který byl upraven a rozšířen pro potřeby navrhnuté aplikace. Hlavní úpravu, aby bylo možné přijímat více topiků, umožnily bloky s názvem „Subscribe (array).vi“ a „Unsubscribe (array).vi“ ze zmiňovaného pluginu. Těmito se nahradí předešle použité bloky „Subscribe (scalar).vi“ a „Subscribe (scalar).vi“. Výhodou nově použitých je zmíněná možnost odebírat více topiků najednou. Vstupní signál do bloku již není jen string s názvem topiku, který se má odebírat, ale pole clusterů (Obrázek 4.8) obsahujících název topiku a také QoS (Quality of Service).

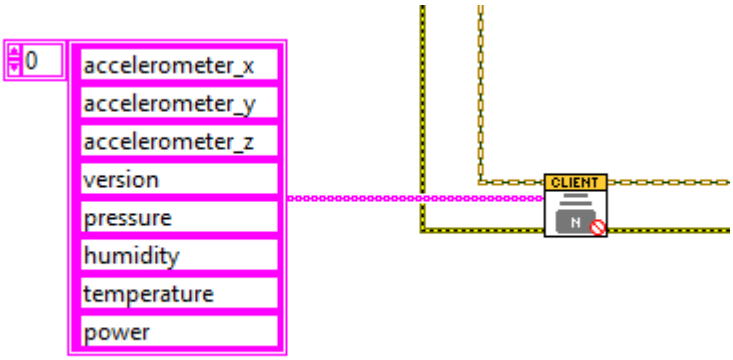



Obrázek 4.8 Příklad vstupního clusteru do Subscribe (Array).vi

Všechny hodnoty QoS jsou defaultně nastaveny na nulu a v této hodnotě byly při testování ponechány pro minimální zátěž komunikace.

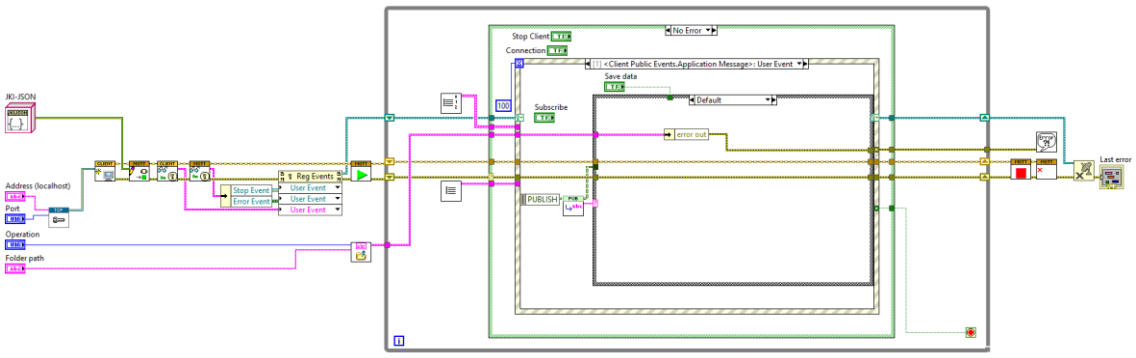
Výstup z bloku naznačuje úspěšnost přijetí zprávy. Vstupem druhého nově použitého bloku je již jen pole použitých topiků (Obrázek 4.9). Oba pak mají ještě navíc další vstup, který nastavuje timeout v milisekundách a který v defaultním stavu (nezapojený vstupní signál) má hodnotu 1000 milisekund.

Oba, tedy, jak vstupní cluster, tak vstupní pole do popisovaných bloků byly později uzavřeny do subvi pro větší přehlednost kódu aplikace.




Obrázek 4.9 Příklad vstupní pole do Unsubscribe (Array).vi

Výsledný klient je uveden na Obrázek 4.10, který v podstatě tvoří celou LabVIEW aplikaci, ovšem s dalšími úpravami a vylepšeními, která jsou popsána v následujících kapitolách.

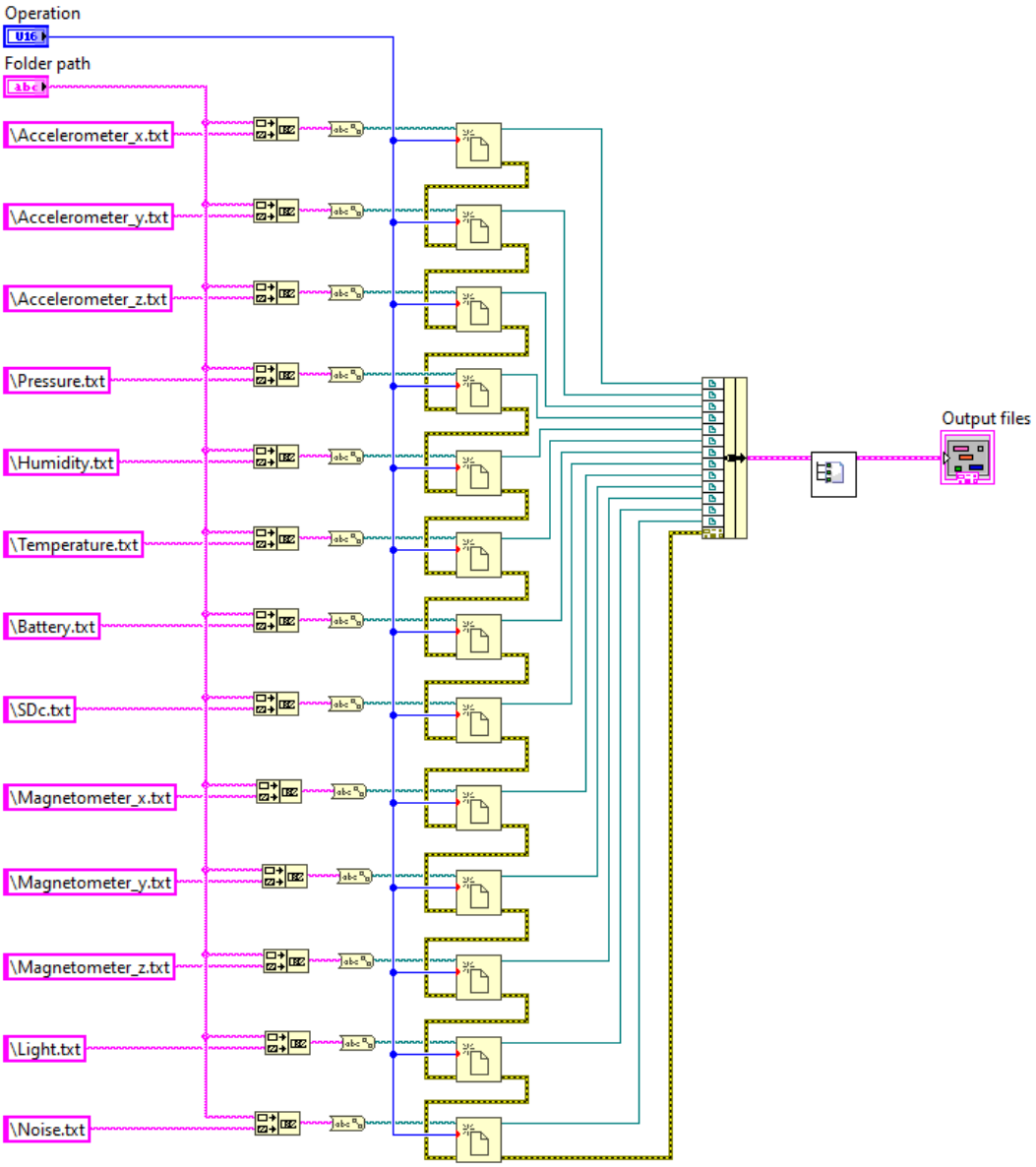


Obrázek 4.10 Výsledný MQTT klient LabVIEW aplikace

## 4.4 Ukládání dat z LabVIEW

Aby bylo možné přijímaná data nejen zobrazovat, ale i zaznamenávat, byla vytvořena možnost ukládat tato data do souboru.

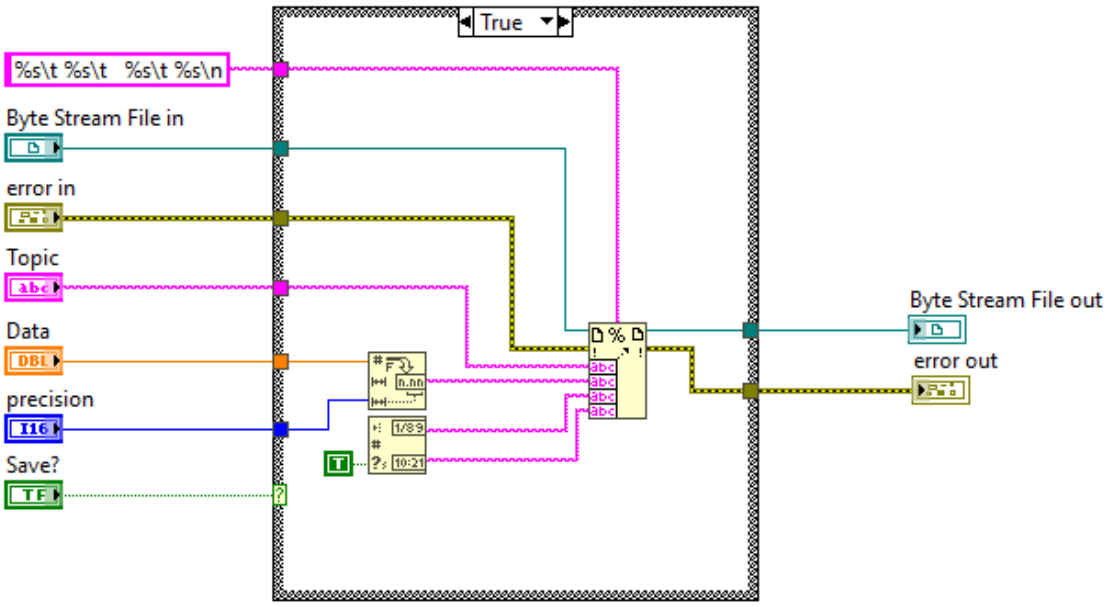
Uživatel má k dispozici tři ovládací prvky a prvním je tlačítko pro zapnutí či vypnutí ukládání. Další je zadání cesty do složky, kde mají být soubory uloženy. Soubory mají zadaný pevný předdefinovaný název i formát pod kterým se uloží do dříve zadaného adresáře. Posledním prvkem je volba operace, tedy zda se má soubor otevřít, vytvořit, nahradit či kombinace předešle zmíněných možností.




Obrázek 4.11 Files (SubVI).vi

Pro práci se soubory byly vytvořeny subVI. První se jmenuje „Files (SubVI).vi“ a vstupem jsou cesta do adresáře pro uložení souborů (string) a operace (enum), která se má provést. Uvnitř se ke stringu adresáře přidá název souboru a řetězec se převede na formát „path“ a signál se přivede do bloku „Open/Create/Replace file“, kterých je paralelně zapojených tolik, kolik je potřeba souborů. Na ně se také přivede druhý vstupní signál operace a výstupy z bloků jsou pak svázány do clusteru a následně vyvedeny na výstup.

Druhým subVI je „Write to file (SubVI).vi“. Prvním vstupem je proměnná typu boolean ovládaná pomocí tlačítka uživatelem, která udává, jestli se mají data ukládat či nikoli. Druhým vstupem je signál „Byte Stream File“ vyvedený z clusteru z předchozího bloku a korespondujícím s konkrétním souborem. Dále je ze zmíněného clusteru vyveden na vstup error a následují poslední dva vstupy, tedy řetězec s názvem odebíraného topiku a samotná data. Uvnitř subVI jsou pak v case struktuře data převedena na string a spolu s názvem topiku jsou i s časovou značkou, obsahující čas i datum, a ostatními vstupními proměnnými přivedeny na vstup bloku „Format into file“. Zde je také přiveden řetězec, značící, v jakém formátu mají být data uložena do souboru, což je také následně provedeno. Posledním vstupem je „precision“, který udává, s jakou přesností na desetinná místa má být hodnota uložena.




Obrázek 4.12 Write to file (SubVI).vi

## 4.5 Čtení dat ze snímačů

V této kapitole jsou popsány možnosti, jak nastavit dostupné snímače a případně způsoby nebo konkrétní funkce, jak data ze snímačů vyčítat.

### 4.5.1 Nastavení a čtení akcelerometru

Jak je blíže popsáno v kapitole 1.1.1, akcelerometr použitý v zařízení XDK 110 je tříosý. Je možné číst všechny tři osy akcelerometru a také celkovou magnitudu která odpovídá vzorci

$$M = \sqrt{x^2 + y^2 + z^2} \qquad (3.1)$$

Kde M je zmíněná magnituda a x, y, z jsou hodnoty zrychlení jednotlivých os.

Pro čtení dat z akcelerometru jej není nutné nastavovat na začátku programu a hodnoty parametrů se doplní těmi defaultními. V případě potřeby je ovšem možné nastavit rozsah v hodnotách 2G (defaultní hodnota), 4G, 8G a 16G. Dále lze nastavit šířku pásma filtru typu dolní propust jako 7.81 Hz, 15.63 Hz, 31.25 Hz, 62.5 Hz, 125 Hz, 250 Hz, 500 Hz a 1000 Hz. Mezi poslední dva prvky, které lze nastavit, patří práh, respektive prahová hodnota, pod kterou je zařízení bráno jako v klidu, tedy bez pohybu a hodnota nad kterou je zařízení bráno jako v pohybu a dle kterých jsou případně spouštěny konkrétní eventy. První dva zmíněné parametry mají vlastní makra, která zde lze s výhodou použít a další dva vyžadují typ uint32.

Jak bylo naznačeno v předešlém odstavci, existují eventy spojené s daným snímačem, které mohou být pro některé aplikace velmi užitečné. Prvním z takových je „any_motion“, který je spuštěn pokaždé, když je zaznamenán rozdíl mezi dvěma úspěšnými měřeními zrychlení, které je větší než nastavená, předešle zmíněná prahová hodnota. Obdobně funguje také „no_motion“, s tou změnou, že je spuštěn, pokud prahová hodnota není překročena mezi dvěma měřeními. Dalšími podobnými eventy jsou „low_g“ a „high_g“, kde oba porovnávají zrychlení s již několikrát zmiňovaným prahem. Zmínit lze také „single_tap“, což je spuštěno po vyšší aktivitě a následné neaktivitě, nebo „double_tap“, který se skládá ze dvou „single_tap“ eventů hned po sobě. Posledním zde zmíněným bude „flat“, jenž se spustí, když je zařízení položeno na plocho, jak název napovídá.

Pro čtení dat z akcelerometru pak lze použít níže uvedené funkce.

```
accelerometer.x_axis.read();
accelerometer.y_axis.read();
accelerometer.z_axis.read();
accelerometer.magnitude.read();
```

### 4.5.2 Nastavení a čtení senzoru okolního prostředí

Senzor okolního prostředí obsahuje čidla pro teplotu, tlak a vlhkost. Mezi konfigurace, které lze na tomto snímači provést, patří nastavení módu napájení. To může být nastaveno makrem „Normal“, tedy že měření probíhá automatizovaně cyklicky mezi aktivní periodou měření a pohotovostním režimem. Druhým nastavením je „Forced“, což znamená, že měření proběhne jednou dle ostatních nastavení, například filtru a pak se senzor vrátí do režimu spánku a data mohou být získána z datového registru. Související možností je také zadání doby trvání pohotovostního režimu, který se zadává ve formátu uint32. Posledním nastavením je „oversampling“, tedy převzorkování, což snižuje šum, ovšem za cenu zvýšení spotřeby energie a času potřebného pro měření.

Následně pak lze číst hodnoty teploty, tlaku a vlhkosti pomocí níže použitých funkcí.

```
environment.temperature.read()
environment.pressure.read()
environment.humidity.read()
```

### 4.5.3 Nastavení a čtení gyroskopu

U gyroskopu se nachází pouze dvě možnosti nastavení. První z nich je zvolení šířky pásma pro filtr typu dolní propust, kde lze pomocí maker nastavit hodnoty 12 Hz, 23 Hz, 32 Hz, 47 Hz, 64 Hz, 116 Hz, 230 Hz a 523 Hz což je i defaultní hodnota. Druhou možností je určit rozsah, který se opět pomocí maker dá zvolit jako 125 °/s, 250 °/s, 500 °/s, 1000 °/s a 2000 °/s (defaultní hodnota).

Následující funkce lze použít pro čtení dat z jednotlivých os.

```
Gyroscope_BMG160.x_axis.read();
Gyroscope_BMG160.y_axis.read();
Gyroscope_BMG160.z_axis.read();
```

### 4.5.4 Nastavení a čtení magnetometru

U tohoto snímače lze nastavit pouze jeden parametr, kterým je mód snímače. Jednotlivé módy a jejich vlastnosti jsou blíže popsány v Tabulka 4.1.

Tabulka 4.1 Módy magnetometru [4]

| Mód [-] | nXY [-] | nZ [-] | ODR [Hz] | Max ODR [Hz] | RMS [μT] | Spotřeba [mA] |
|---|---|---|---|---|---|---|
| Low power preset | 3 | 3 | 10 | >300 | 1.0/1.0/1.4 | 0.17 |
| Regular preset | 9 | 15 | 10 | 100 | 0.6/0.6/0.6 | 0.5 |
| Enhanced regular preset | 15 | 27 | 10 | 60 | 0.5/0.5/0.5 | 0.8 |
| High accuracy preset | 47 | 83 | 20 | 20 | 0.3/0.3/0.3 | 4.9 |

Kde Mód je název jednotlivých módů, nXY a nZ jsou počty opakování na daných osách (XY, nebo Z), ODR je „Output Data Rate“, tedy výstupní tok dat, Max ODR udává maximální ODR, RMS značí střední kvadratickou hodnotu šumu v osách x/y/z a poslední je spotřeba pro jednotlivé módy.

Ze snímače pak lze číst nejen jednotlivé osy, ale taktéž Hallův odpor. Funkce pro vyčítání jsou naznačeny níže.

```
magnetometer.x_axis.read();
magnetometer.y_axis.read();
magnetometer.z_axis.read();
magnetometer.resistance.read();
```

### 4.5.5 Nastavení a čtení senzoru intenzity osvětlení

Na tomto snímače lze nastavit manuální mód, což znamená že je možné manuálně nastavit integrační čas a také možnost použít režim vysokého jasu. V případě ponechání tohoto módu ve „false“ jsou tyto hodnoty doplněny automaticky. Dále je možné zadat již zmíněný integrační čas, po který sensor sbírá hodnoty osvětlení (defaultně je nastaveno 800 ms). Snímač je možné přepnout také do výše uvedeného režimu vysokého jasu, kdy do ADC převodníku jde pouze osmina celkového proudu. Poslední možností nastavení je „continuous mode“ což při hodnotě „true“ umožňuje kontinuální měření intenzity světla a hodnoty jsou ze snímače čteny dle zadaného integračního času. V hodnotě „false“ jsou hodnoty intenzity změřeny jen jednou za 800 ms a jedná se tak o úspornější režim.

Následně lze hodnotu intenzity okolního osvětlení vyčíst pomocí funkce uvedené níže.

```
light.intensity.read();
```

### 4.5.6 Nastavení a čtení senzoru hluku

U senzoru okolního hluku lze nalézt pouze dvě možnosti konfigurace. První je vzorkovací frekvence, která je defaultně nastavena na 2560 Hz (maximum je 45 kHz) a druhá je čas v milisekundách, jak dlouho má snímač čekat na další vzorky, pokud jich není dostatek (defaultně 100 ms). Výstupní vzorky hluku jsou vypočítávány z 256 vzorků získaných ze snímače, takže pro získání maximální použitelné frekvence získávání dat, je třeba nastavenou frekvenci vydělit 256. Je také vhodné zmínit že výstupem ze snímače je veličina bez jednotky a je třeba ji upravit či zkalibrovat.

Hodnoty ze snímače jsou čteny následující funkcí.

```
noise_sensor.noise.read();
```

## 4.6 Vyčítání stavu baterie zařízení

Aby bylo možné zjišťovat stav nabití zařízení XDK 110, byla implementována funkce pro zjišťování stavu akumulátoru, která je uvedena níže.

```
every 5 minutes {
      let ps = XDK110.powerStatus.read();
      where(ps) {
          is(PowerStatus.Battery -> level) {
            mqttXDK.power.write(`${level}`);
          }
          is(PowerStatus.Corded) {
                  mqttXDK.power.write(`100`);
          }
      }
```

Jedná se o periodický event, který se volá každých 5 minut. Nejdříve je do proměnné vyčten stav baterie pomocí uvedené dostupné funkce a následně je tato hodnota odeslána. Protože ve stavu, kdy je akumulátor zcela nabitý či zcela vybitý, není výstupem funkce úroveň nabití, je tato hodnota doplněna ručně. Dalším problémem je, že mezi zmíněnými dvěma stavy nelze rozpoznat, který nastal a je tak nutné brát tuto informaci v potaz při zpracovávání dat.

## 4.7 Zapisování a čtení dat z SD karty

Jak bylo uvedeno v kapitole 4.3.2, maximální používaná frekvence odesílání dat je 20 Hz. Ovšem aby bylo možné zapisovat data ve vyšších frekvencích, je nutností ukládat data přímo do zařízení, a k tomuto účelu byl využit dostupný slot na SD kartu. Další výhodou je také to, že nemusí být dostupná žádná síť nebo jiný způsob komunikace pro okamžité odesílání dat, ale je možné data ukládat a odeslat později po opětovném připojení k síti.

Vzhledem k tomu, že funkce pro zapisování do souboru po každém volání daný soubor otevře a zavře, je pak časově náročná na provedení. Z tohoto důvodu je v programu vytvořena proměnná typu string, která slouží jako vyrovnávací paměť. Do této proměnné je vždy uloženo 100 nasbíraných vzorků dat, které jsou následně najednou zapsány jedním voláním funkce. Proces zapisování je tímto velmi výrazně zrychlen a je tudíž možné zaznamenávat data i s vysokými frekvencemi.

V současné verzi firmwaru jsou do SD karty zapisovány pouze hodnoty z akcelerometru, současně ovšem není zapisována časová značka a jedná se tak pouze o surová data. Ze snímače zrychlení jsou do souboru ukládány vyčítané hodnoty magnitudy zrychlení.

Při čtení dat z SD karty dochází ke stejnému problému jako při zápisu, tedy že funkce po každém zavolání soubor otevře a zavře. Bylo by možné použít opět proměnnou typu string jako vyrovnávací paměť, ovšem tuto proměnnou není možné indexovat a nebylo by tak možné získat konkrétní vzorky zvlášť. Získat konkrétní vzorky je pro odeslání potřebné, protože v LabVIEW aplikaci se po odzkoušení mnoha různých konfigurací klientů MQTT na obou stranách nepodařilo přijímat jak string, tak ani pole hodnot. Daří se přijmout pouze vždy jedno číslo, a proto je vždy ze souboru vyčítán pouze jeden vzorek následně převedený ze stringu na číslo a poté odeslán.

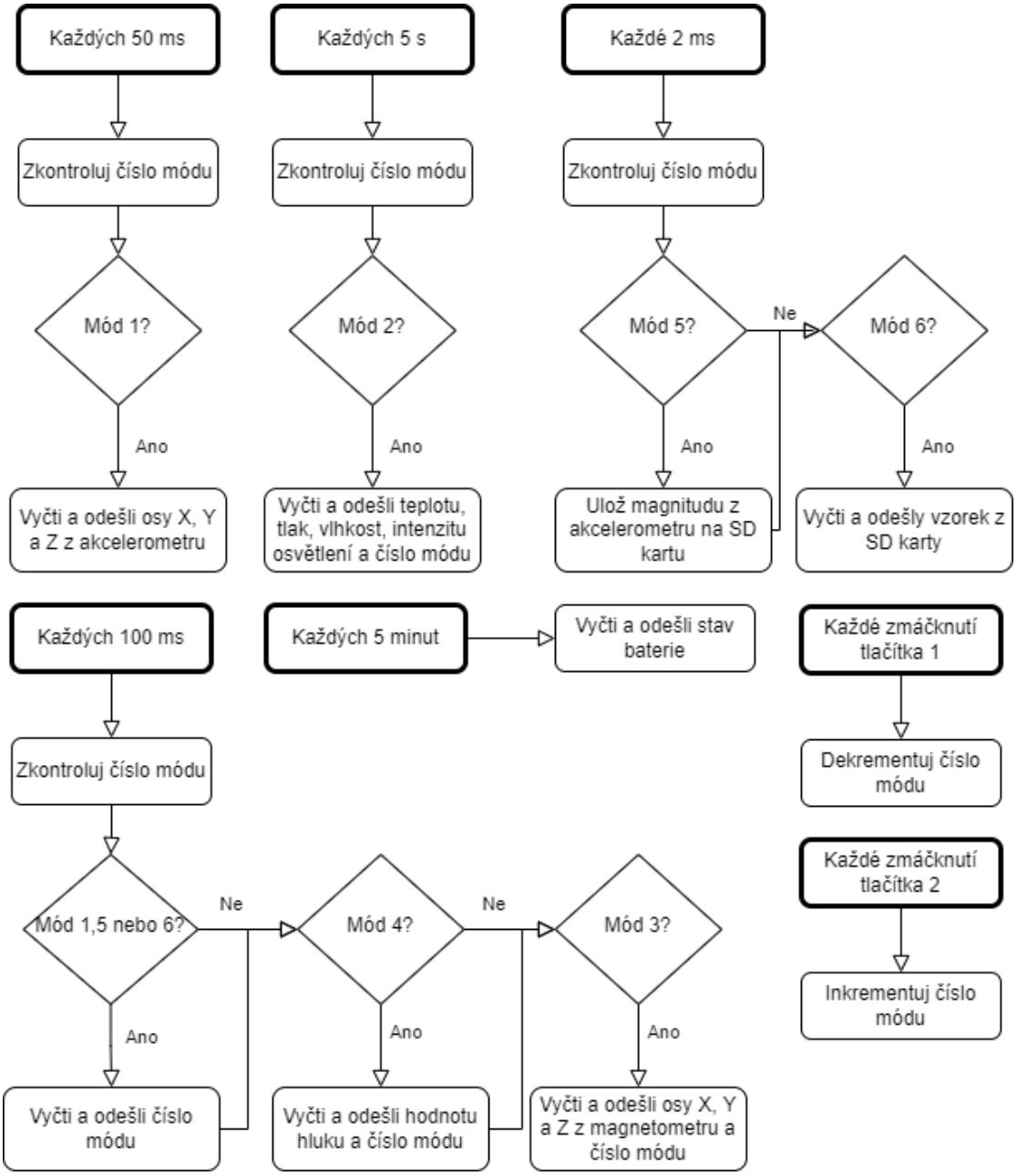
## 4.8 Módy firmwaru

Vzhledem k tomu, jak bylo popsáno v kapitole 4.3.1, že nelze do zařízení posílat ovládací signály, bylo nutné přijít s jiným způsobem ovládání. To je nutností, jelikož čtení a odesílání všech snímačů najednou by bylo značně náročné, jak na výdrž akumulátoru, tak i na výpočetní kapacitu. K tomuto účelu byly použity programovatelná tlačítka nacházejících se na horní straně XDK 110. Pomocí těchto tlačítek se přepíná mezi jednotlivými módy popsanými níže.

Struktura programu pak odpovídá návrhu provedeném v kapitole 3 a je tedy implementováno 6 módů, které odpovídají nějakému snímači, nebo práci s SD kartou.

Tabulka 4.2 Módy firmwaru

| Číslo módu | Perioda vyvolávání | Indikace módu | Vstup dat | Výstup dat |
|---|---|---|---|---|
| 1 | 50 ms | | Akcelerometr | MQTT |
| 2 | 5 s | | Senzor teploty, tlaku, vlhkosti, intenzity osvětlení | MQTT |
| 3 | 100 ms | | Magnetometr | MQTT |
| 4 | 100 ms | | Senzor hluku | MQTT |
| 5 | 2 ms | Červená LED | Akcelerometr | SD |
| 6 | 2 ms | Oranžová LED | SD | MQTT |



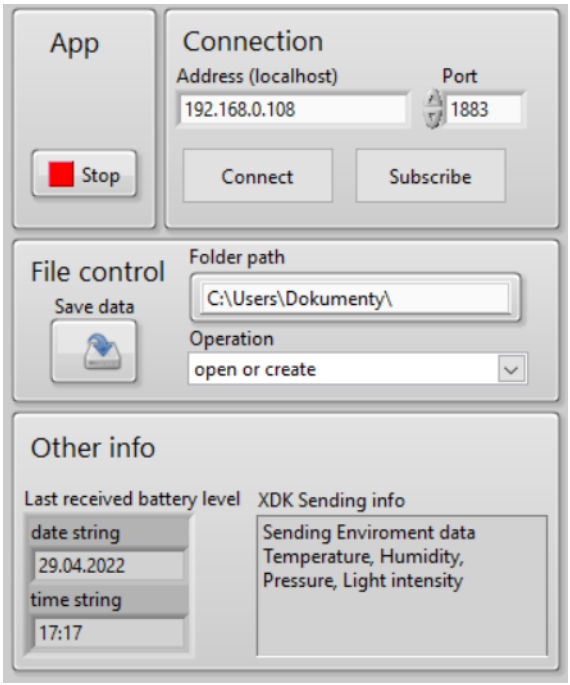

Obrázek 4.13 Diagram firmwaru

V prvním módu jsou vyčítána a odesílána data z akcelerometru, a to pro všechny tři osy. Toto se děje periodicky každých 50 milisekund, dokud není zmáčknuto některé z tlačítek. S periodou 100 milisekund je pak odesíláno číslo aktuálního módu. Při stisku pravého tlačítka číslo dvě, dojde k přepnutí do druhého módu, ve kterém jsou s periodou 5 sekund odesílána data ze snímačů teploty, tlaku, vlhkosti a okolního osvětlení zároveň s číslem právě nastaveného módu. Při dalším stisku stejného tlačítka dojde k odesílání údajů z magnetometru (osy X, Y a Z) s periodou 100 milisekund a opět se se stejnou periodou odesílá i informace o aktuálním módu. Stejně tak dochází k vyčítání a odesílání hodnot získaných ze senzoru hluku v následujícím módu. V předposledním jsou pak vyčítána a ukládána data na SD kartu a tento mód je indikován zapnutím červené LED diody na zařízení. Poslední je mód vyčítání a odesílání uložených dat z SD karty, což je indikováno svícením oranžovou LED diodou. Při dalším stisku tlačítka číslo dvě se opět přejde na začátek a lze tak módy přepínat dokola. Tlačítko číslo jedna má opačnou funkci než předešlé tlačítko a posunuje se mezi jednotlivými módy na druhou stranu. Některé čísla právě nastavených módů jsou odesílána v pomalejší periodě v jiném eventu než data, a to kvůli šetření energie a výpočetní kapacity, protože není nutné odesílat číslo módu například každé 2 ms. Diagram výsledného firmwaru je vyobrazen na Obrázek 4.13 a přepínání mezi módy je v kapitole návrhu na Obrázek 3.2.

## 4.9 Ovládání výsledné LabVIEW aplikace

Výsledná LabVIEW aplikace je ovládána z front panelu, jak je zvykem. V této kapitole bude panel a jeho ovládací prvky přiblíženy pro lepší pochopení veškerých funkčností.

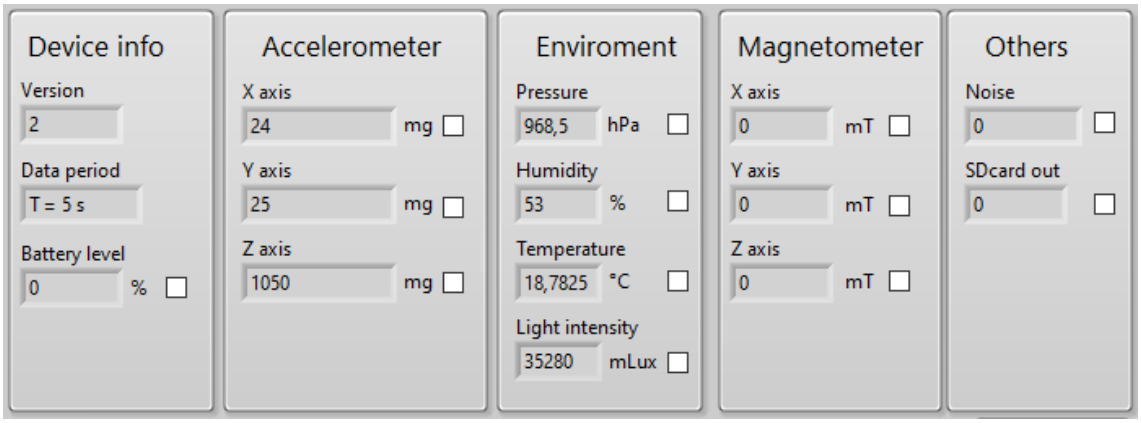
Na Obrázek 4.14 je ukázána levá část front panelu. V levém horním rohu je část pro ovládání aplikace, kde lze pouze po spuštění zastavit aplikaci stisknutím tlačítka „Stop“. V pravé horní části je nastavení a ovládání konektivity aplikace. Zde se zadá IP adresa do prvku s názvem „Adress (localhost)“ a poté také port do vedlejšího kontrolního prvku „Port“. Adresa a port které se zadávají patří pro MQTT Broker, ze kterého mají být data odebírána. Nachází se zde i dvě tlačítka, které lze použít po zadání adresy a portu. Prvním je tlačítko s názvem „Connect“, pomocí kterého se lze k Brokeru připojit a druhé má název „Subscribe“ pomocí kterého se spustí odebírání na programem nastavené topiky.




Obrázek 4.14 Front panel (první část)

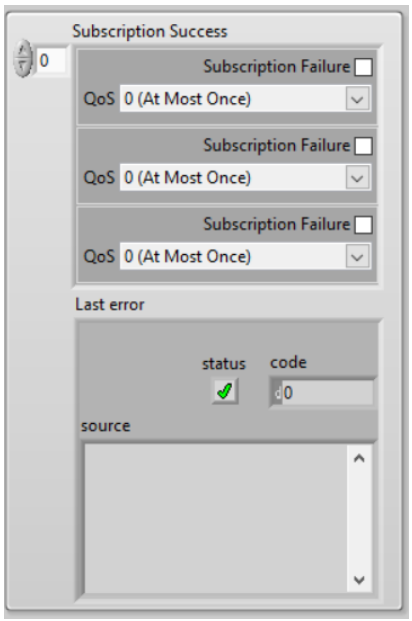
Uprostřed levé části front panelu se nalézá ovládání ukládání dat. V prvním ovládacím prvku „Folder path“ se zadává cílová cesta k adresáři do kterého mají být požadovaná data ukládána. Zadává se pouze cesta a názvy souborů s příponami, do kterých se data zapisují, jsou předdefinovány programováním. Následně se zadá operace, která má být se soubory provedena, tedy otevřít, vytvořit, přepsat či kombinace předešle zmíněných. Posledním ovládacím prvkem v této části je tlačítko „Save data“, které se při stisku rozsvítí zeleně, což indikuje zapnutí ukládání a dalším stiskem se ukládání vypne. Takže je možné kdykoliv za běhu programu ukládání zapnout nebo vypnout.

Ve spodní části je blok „Other info“, ve kterém je vlevo uveden čas a datum, kdy byla naposledy přijata hodnota o stavu baterie. Tato informace pomáhá pro lepší přehled o stavu akumulátoru, protože perioda přijímání úrovně nabití je 5 minut a je tak možné sledovat, kdy byla hodnota naposledy aktualizována. Jak bylo zmíněno dříve, nelze rozpoznat je-li zařízení plně nabito či vybito, a tak i tato informace může pomoci identifikovat stav akumulátoru. Další informací, která je zobrazována v tomto bloku, je textový údaj o aktuálně přijímaném módu, konkrétně ze kterého snímače nebo snímačů jsou právě data přijímána.




Obrázek 4.15 Front panel (druhá část)

V prostřední části front panelu (Obrázek 4.15) se nachází indikátory pro zobrazování přijímaných dat a dalších informací. První blok s názvem „Device info“ ukazuje různé informace o zařízení, a to konkrétně verzi, respektive číslo aktuálně přijímaného módu, periodu, se kterou jsou data odesílána ze zařízení a také úroveň nabití akumulátoru. Následuje blok s údaji o akcelerometru pro všechny tři osy, vedle kterého se nachází enviromentální údaje. Mezi nimi je na prvním místě okolní tlak v hPa, níže je pak možné nalézt relativní vlhkost v procentech, dále teplotu okolí ve stupních Celsia a zakončující údaj je intenzita světla. Vedle je předposlední blok prostřední části front panelu, který zobrazuje data z magnetometru a v posledním jsou data ze snímače hluku a zobrazování dat přijímaných z SD karty v zařízení.

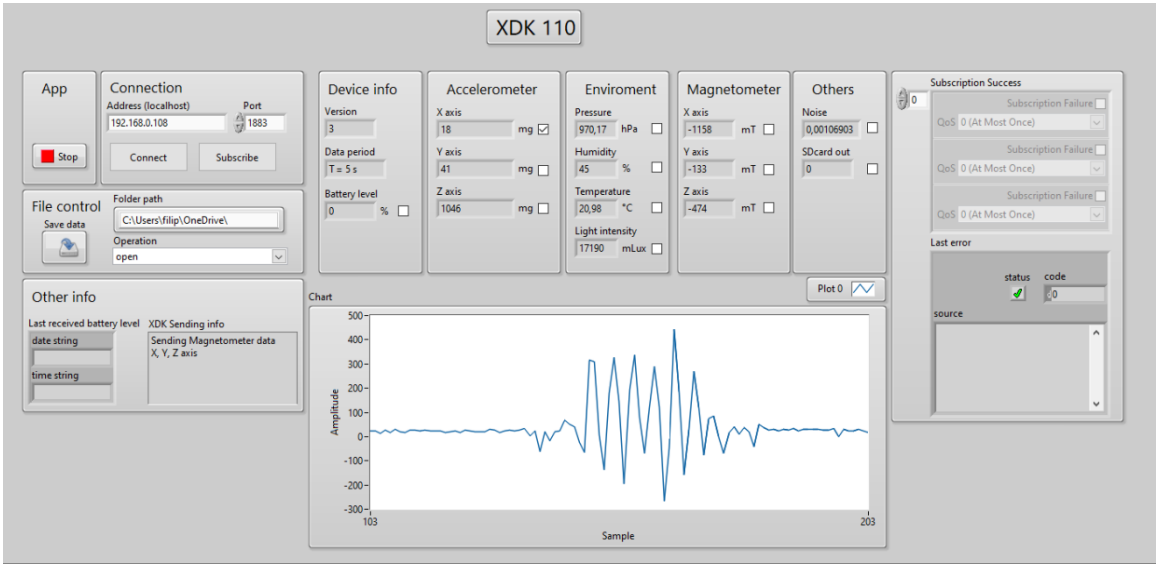



Obrázek 4.16 Front panel (třetí část)

V předposlední, pravé části front panelu (Obrázek 4.16) je možné vidět dva indikační prvky. Prvním je ukazatel programově nastaveného QoS (Quality of Service), kde je zobrazována i související informace bylo-li přijetí zprávy neúspěšné. Níže je pak zobrazován error, pokud k nějakému došlo, konkrétně kód chyby a případný textový popis.

Jako poslední prvek front panelu je graf, na kterém je možné vykreslovat přijímaná data. Zobrazovat je možné vždy jen jeden set dat najednou, jinak jsou průběhy pomíchány. Výběr, která data zobrazovat do grafu je zajištěn pomocí checkboxů, které jsou vidět na Obrázek 4.15 a vždy patří prvku, u kterého jsou uvedeny.

Front panel jako celek je ukázán na Obrázek 4.17.




Obrázek 4.17 Celý front panel

Po otevření LabVIEW projektu, se výše zmíněná aplikace nachází pod názvem „MQTT_Subscriber.vi“ a aby byla možná komunikace přes MQTT, je taktéž nutné využít aplikaci „MQTT_broker.vi“, která plní funkci brokeru. Do ní se pro správnou funkci zadává pouze adresa a port.

# 5. TESTOVÁNÍ A MĚŘENÍ

V této kapitole bude otestována aplikace s firmwarem v zařízení XDK 110. Ve vývojovém hardwaru jsou dostupné snímače, pomocí kterých bude provedeno několik praktických měření a bude tak vyzkoušena funkčnost celé implementace a zároveň to bude ukázkou možností použití.

Měřena bude výdrž baterie pro více konfigurací, respektive módů, a více teplot okolí, aby bylo možné určit, na jak dlouho lze zařízení použít bez připojení k napájení. Jako praktické ukázky použití bude provedeno měření v lednici pro enviromentální data, tedy teplota, tlak a vlhkost nebo měření magnetické indukce a hluku poblíž elektromotoru.

## 5.1 Testování výdrže akumulátoru

Testování výdrže akumulátoru zařízení bylo proměřeno v několika různých módech. Nejdříve při rychlém odesílání dat z akcelerometru a následně při pomalejším odesílání dat ze snímačů okolního prostředí. XDK 110 bylo nejprve připojeno k napájení a plně nabito, poté byl spuštěn program a po následném nastavení parametrů bylo zařízení odpojeno a ponecháno do úplného vybití.

Dříve bylo implementováno ukládání dat v aplikaci do souboru a mohlo tak být využito této funkcionality pro zaznamenání dat. Stejně jako v předchozím případě, data byla ze zařízení odesílána s větší periodou 5 minut pro menší zátěž akumulátoru a výkonu zařízení. Tato perioda také bohatě stačí pro získání dostatečně podrobného souboru dat pro následné vykreslení grafu a případné analýze.

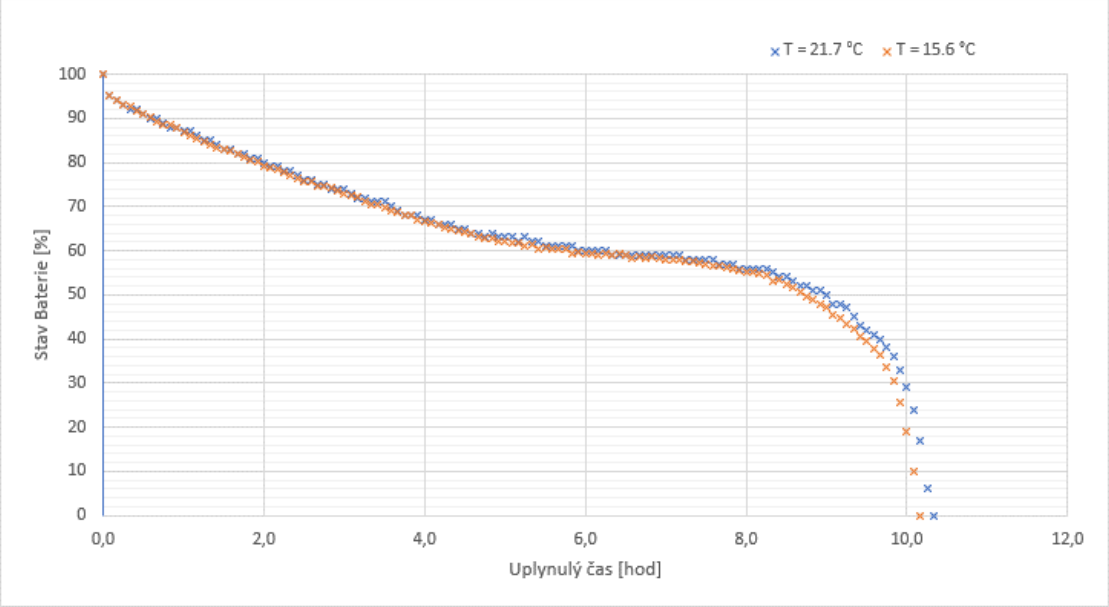
Všechna měření výdrže akumulátoru probíhala buď v běžné vytápěné místnosti pro měření při vyšší teplotě kolem 22 °C a nevytápěné místnosti pro měření při nižší teplotě kolem 16 °C. Po zapnutí a nastavení zařízení a jeho následnému odpojení od napájení, bylo po celou dobu měření ponecháno v dané místnosti bez další manipulace.

### 5.1.1 Testování při frekvenci odesílání dat 20 Hz

Data z akcelerometru jsou odesílána na frekvenci 20 Hz pro všechny tři osy. V tomto módu nastavení bylo zařízení ponecháno při měření. XDK 110 bylo nejprve spuštěno společně s LabVIEW aplikací a následně bylo odpojeno od napájení. Přijatá data z akcelerometru byla pouze zobrazována ale nikoli ukládána, na rozdíl od přijatých dat o stavu akumulátoru. Pomocí dříve popsaných funkcí byla tato data zapisována do souboru ihned po jejich přijetí.

Komunikace probíhala bezdrátově přes WLAN pomocí MQTT protokolu na topiku s názvem „power“. Vyčítání a odesílání stavu akumulátoru bylo vykonáváno každých 5 minut a ze získaných hodnot byl vytvořen graf (Obrázek 5.1). Byly zaznamenány také podmínky na počátku měření, jako teplota 21.7 °C, tlak 962,5 hPa a relativní vlhkost 48 %. Na konci měření byly podmínky naměřeny jako 22.4 °C, 967 hPa a 46 % vlhkosti. Naměřené hodnoty byly orientačně získány pomocí XDK 110.

Z grafu je patrné že baterie v daném módu vydržela napájet zařízení po dobu více než 10 hodin. Za povšimnutí také stojí, že hodinu před vybitím ukazovalo zařízení úroveň baterie 48 % a o další přibližně hodinu později byl již téměř vybitý. Průběh vybíjení je pak ve zmíněném grafu dobře viditelný.




Obrázek 5.1 Graf výdrže akumulátoru (20 Hz)

Později bylo provedeno další měření při stejné konfiguraci zařízení, jen s rozdílnými okolními podmínkami, konkrétně při nižší teplotě prostředí. Testování bylo prováděno při 15.6, 974.2 hPa a 44% relativní vlhkosti. Na konci měření byly okolní podmínky naměřeny jako 16.2 °C, 969.1 hPa a 45 % relativní vlhkosti. Ze získaných dat lze vidět že výdrž zařízení se při nižší teplotě zkrátila přibližně o 10 minut.

Na výše uvedeném grafu je možné vidět zjištěnou charakteristiku akumulátoru při teplotě okolí přibližně 21.7 °C. Lze pozorovat, že na začátku je mírný skok ve stavu akumulátoru, tedy ze 100 % na přibližně 95 %. Průběh vybíjení je poté pozvolný s téměř lineárním charakterem až do hodnoty kolem 64 %, kdy se rychlost vybíjení sníží a u hodnoty 55% strmost začne prudce narůstat. Na 59 % akumulátoru si je možno také povšimnout téměř hodinového úseku, kdy je úroveň nabití konstantní.

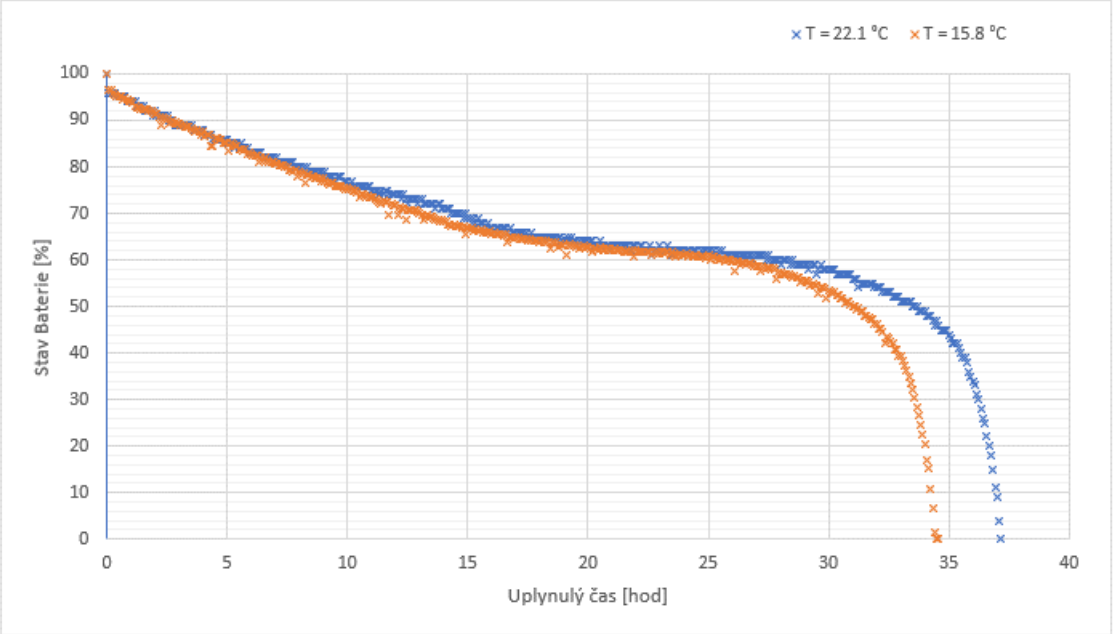
Při teplotě okolí přibližně 15.6 °C se průběh vybíjení akumulátoru po většině délky téměř shoduje s předchozím průběhem, ovšem ke konci se hodnoty začínají lišit výrazněji a lze tak vidět, že výdrž zařízení je při nižší teplotě okolí kratší.

### 5.1.2 Testování při frekvenci odesílání dat 0.2 Hz

Pro testování výdrže akumulátoru při frekvenci odesílání dat 0.2 Hz bylo použito totožné metodiky jako u předchozího měření v podkapitole 5.1.1. Zařízení bylo nejprve plně nabito a následně po jeho spuštění zároveň s aplikací LabVIEW bylo odpojeno od napájení.

Zařízení odesílalo každých 5 sekund hodnoty ze snímačů okolního prostředí, ze kterých byly zjištěny podmínky měření. Zaznamenáváno bylo i přijímání hodnot o úrovni nabití akumulátoru. Tyto informace byly opět přijímány s periodou 5 minut a ukládány do souboru. Ze získaných dat byl vytvořen graf (Obrázek 5.2). Podmínky měření byly na začátku určeny jako 22.1 °C, 968.6 hPa a 49 % relativní vlhkosti. Na konci měření byly podmínky zjištěny pomocí XDK 110 jako 21.4 °C, 964.7 hPa a 48 % relativní vlhkosti. Celková výdrž v tomto případě činila 37,2 hodin.

Hodnoty z druhého měření výdrže zařízení byly získány při nižší teplotě 15.8 °C, tlaku 977.4 hPa a 46 % relativní vlhkosti, které byly zaznamenány na začátku měření. Na konci měření byly hodnoty okolních podmínek 15.3 °C, 971.5 hPa a 49 % relativní vlhkosti. Z dat je patrné že výdrž je tentokrát 34,6 hodiny, takže opět došlo ke snížení výdrže při nižší teplotě okolí a ta se zkrátila o 2,6 hodiny.




Obrázek 5.2 Graf výdrže akumulátoru (0.2 Hz)

V grafu výše je pak z celého souboru dat vykreslen průběh vybíjení při teplotě okolí přibližně 22.1 °C. Tvar je téměř totožný jako v předchozím případě, jen se jedná o delší časový úsek. Lze pozorovat podobný skok ze 100 % na 95 % na začátku průběhu následované pozvolným klesáním. Opět se na hodnotě lehce přes 60 % snižuje strmost klesání a vybíjení se zpomaluje a o několik procent níže se prudce zvyšuje. V některých bodech je také možné vidět určitý rozptyl, zvláště kolem méně strmější části. Hodnota nabytí je v určitém bodě nižší než v několika následujících bodech.

Průběh při teplotě kolem 15.8 °C se s předchozím průběhem shoduje pouze na začátku, přibližně v prvních pěti hodinách měření. Dále se průběhy začnou od sebe odchylovat, kde lze mezi 10 a 15 hodinou měření pozorovat zvětšené odchýlení, které se následně opět zmenší až do přibližně 25 hodiny měření. V tomto bodě se průběhy postupně začnou odchylovat nejvíce až do úplného vybití.

### 5.1.3 Zhodnocení měření výdrže akumulátoru

Měření výdrže akumulátoru bylo provedeno ve dvou konfiguracích zařízení a při různých okolních podmínkách. Nejdříve bylo zařízení proměřeno při odesílání dat z akcelerometru na frekvenci 20 Hz, a to při teplotě okolí přibližně 21.7 ± 2 °C a 15.6 ± 2 °C. V prvním případě výdrž dosáhla 10.3 hodin a při nižší teplotě se výdrž snížila na 10.2 hodin. Ze získaných dat byl vytvořen graf, který je možno vidět na Obrázek 5.1.

Dále bylo zařízení proměřeno při odesílání enviromentálních dat (teplota, tlak a vlhkost) při frekvenci 0.2 Hz. První měření bylo při teplotě 22.1 ± 2 °C, kde bylo dosaženo výdrže 37.2 hodiny a následně při teplotě okolí 15.8 ± 2 °C, kde byla výdrž 34.6 hodin. Všechny hodnoty jsou pak znázorněny v grafu na Obrázek 5.2. Teplota v místnostech, ve kterých probíhalo měření se pohybuje přibližně v rozmezí ± 2 °C, jak bylo zjištěno pomocí zařízení XDK 110.

Snížení výdrže akumulátoru může být způsobeno zvýšením jeho odporu při snížení teploty a tím dochází k vyšší spotřebě, což se pak více projeví v delším časovém intervalu, jak může být patrné z výsledků měření.

## 5.2 Měření korekčního koeficientu

Vzhledem k tomu, že snímač teploty je z hlediska konstrukce umístěn uvnitř pouzdra chránící zařízení (Obrázek 1.1), senzor tak měří spíše teplotu uvnitř tohoto obalu než teplotu vnější, která je vlivem vlastního zahřívání zařízení rozdílná. Také z tohoto důvodu má výsledná teplota určitý offset, který je pro správné určení hodnoty teploty nutné kompenzovat. Na webu výrobce [26] se lze v rubrice Mita/Sensors/Enviromental v ukázkovém příkladu dozvědět, že zde používají jako hodnotu korekčního koeficientu pro kompenzaci offsetu 6.5975 °C. Při použití zmíněného koeficientu, ale při vyčítání hodnot teplota neodpovídala skutečnosti a bylo tak pro další měření nutné najít správný

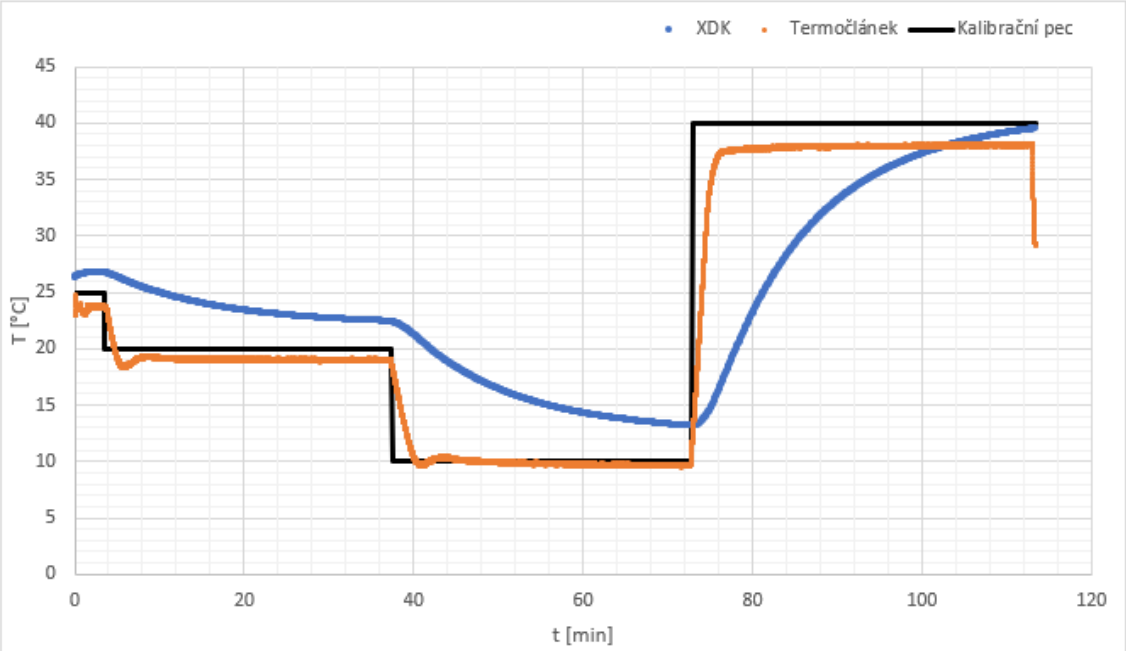
korekční koeficient snímače teploty pro toto konkrétní zařízení. Na straně 16 v manuálu [1] se lze dočíst, že odhad koeficientu při používání senzorů a BLE je mezi 3 a 4 K.

Pro měření bylo vhodné mít prostředí s definovanou teplotou a k tomuto účelu byla použita kalibrační pec AOIP Hyperion s číslem 000000317417-0000. Kromě integrovaného snímače teploty v peci byl ještě použit termočlánek typu J pro měření teploty uvnitř komory pece. Pro sběr dat z termočlánku byla použita platforma Compactdaq s evidenčním číslem 15S64/003 a sériovým číslem 168C56F zároveň s měřící kartou NI 9219 198848A-01L se sériovým číslem 1700119.


Obrázek 5.3 Měření korekčního koeficientu

Nejprve bylo zapnuto XDK 110 v konfiguraci, při které byla data o teplotě ukládána s periodou 5 sekund na SD kartu a následně bylo zařízení vloženo do pece, jak je možné vidět na Obrázek 5.3 vlevo. Na stejném obrázku vpravo je pak už zařízení uvnitř přiklopeno a dovnitř vede vodič termočlánku. Jak je zmíněno v předchozím odstavci, termočlánek byl připojen k měřící kartě. Z karty byly data sbírány pomocí LabVIEW aplikace do PC a zde ukládány. Při vkládání zařízení do komory bylo na kalibrační peci nastaveno 25 °C, po vložení a přiklopení pak teplota byla změněna na 20 °C. Kvůli vyšší

časové konstantě zařízení snímače teploty v zařízení XDK 110 byla nastavená teplota ponechána v délce asi 30 minut a následně změněna na 10 °C a po stejném časovém úseku byla teplota opět na peci změněna na 40 °C. Okolní podmínky v laboratoři byly zaznamenány na začátku měření jako: teplota 23.8 °C, vlhkost 36 % a tlak 991.1 hPa. Měření probíhalo 10. 5. 2022 v čase 11:31 až 13:28.




Obrázek 5.4 Průběhy naměřených teplot termočlánku a XDK 110

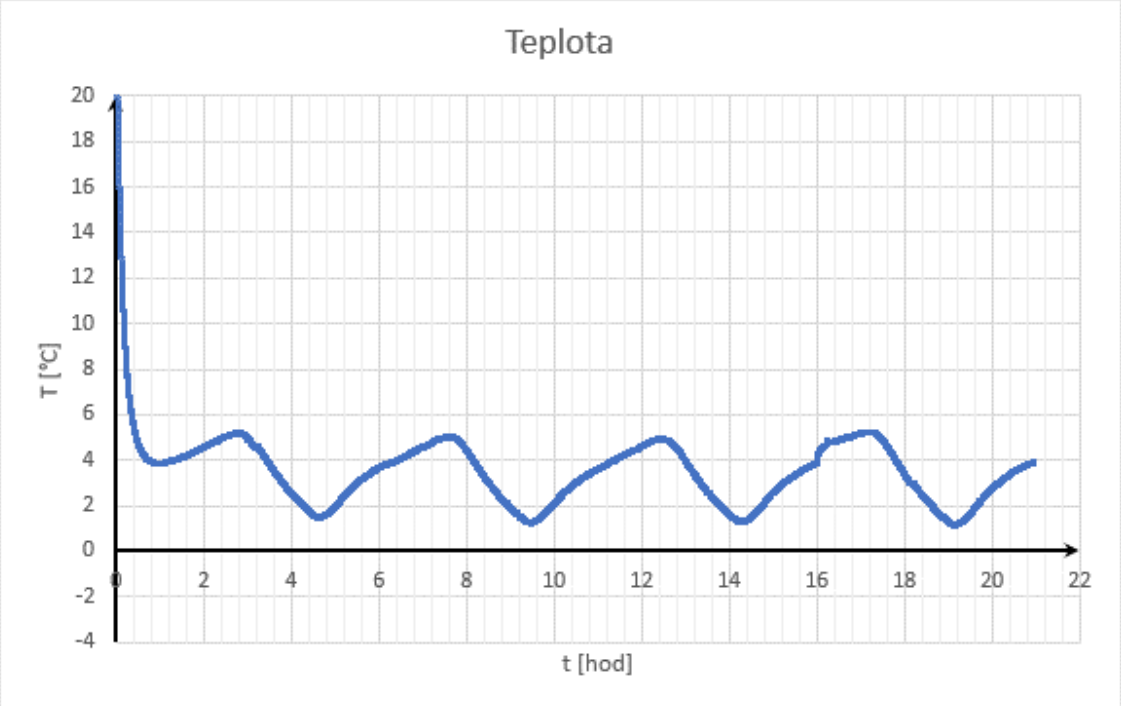
Výsledná data z měření jsou vykreslena na Obrázek 5.4. Černý průběh zobrazuje nastavenou teplotu v kalibrační komoře. Vývoj skutečné teploty je zachycen oranžovým průběhm, který patří termočlánku. Na modrém průběhu jsou pak vidět hodnoty měření snímačem teploty z XDK 110. Je zde patrná, dříve zmiňovaná, vyšší časová konstanta a také patrný rozdíl v teplotě. Rozdíl mezi nastavenou hodnotou v kalibrační peci a hodnotou naměřenou z termočlánku může být způsobena nedokonalým utěsněním komory pece. Rozdíly mezi hodnotou termočlánku a hodnotou z XDK 110 činily v přibližně ustáleném stavu 3.42 °C u nastavené teploty komory 20 °C, 3.49 °C u nastavené teploty komory 10 °C a 1.49 °C při nastavené teplotě komory 40 °C. Je tedy patrné že se vzrůstající teplotou se offset zkoumaného snímače snižuje, což by mohlo odpovídalo realitě, kdy vliv vlastního ohřevu zařízení se snižuje při vzrůstající teplotě okolí. Při běžném měření v přibližném rozsahu teplot kolem 10 °C a pokojové teplotě kolem 20 °C lze odhadnout výsledný koeficient mezi 3.5 °C a 3.4 °C a při vyšších teplotách se tento koeficient snižuje a kolem 40 °C ho lze odhadnou na 1.5 °C.

## 5.3 Měření teploty v lednici

Jako testování fungování zařízení a zároveň jako demonstrace možného použití výsledku této práce bylo zařízení použito pro měření teploty v lednici. XDK 110 bylo v konfiguraci kdy odesílalo každých 5 vteřin aktuální informaci o teplotě tlaku a vlhkosti, které byly přijímány a následně i zaznamenávány v LabVIEW aplikaci. Každých 5 minut byla také ze zařízení odeslána hodnota o stavu baterie, která byla zobrazována na front panelu aplikace, aby byla snadno dostupná informace o stavu zařízení.

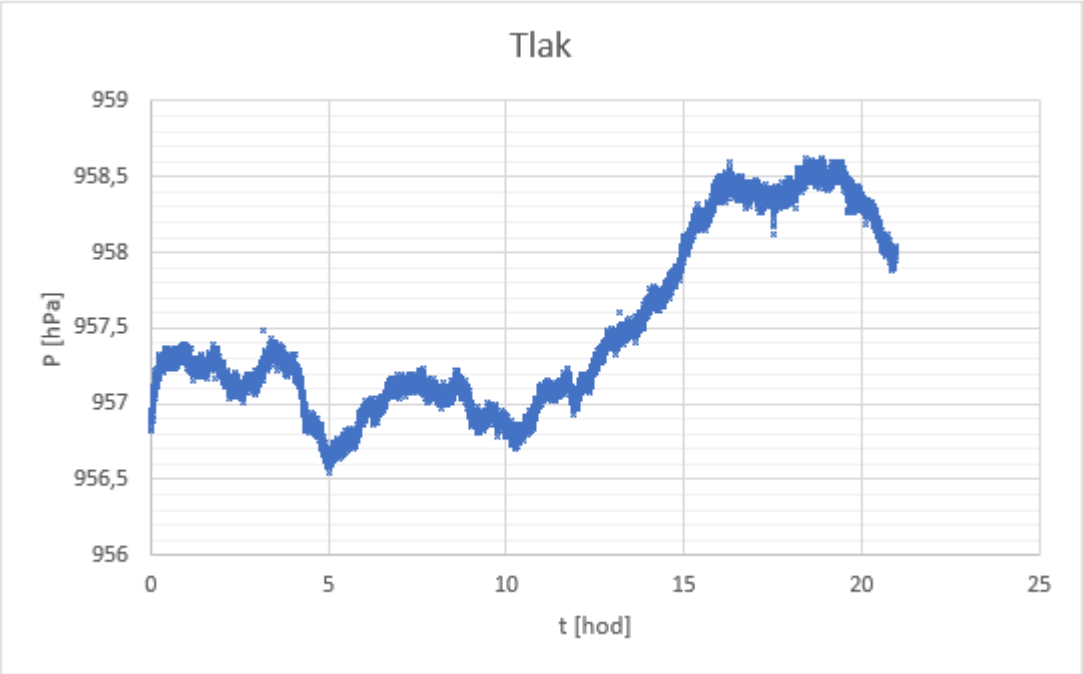
Lednice, ve které bylo prováděno měření je běžná domácí vestavná lednice Electrolux KNT2LF18S. Chlazení bylo nastaveno pomocí otočného kolečka uvnitř lednice na číslo 4. Zařízení XDK 110 bylo umístěno na kraj blízko dvířek do prostřední police. Měření probíhalo za běžného provozu lednice, a tedy jedním z předmětů zjišťování je, jak budou snímače, respektive měřené veličiny reagovat na otevření lednice. Ta byla otevřena během měření celkem třikrát, a to přibližně v časech 3 hodiny od začátku měření, 18 hodin od začátku měření (u obou se jednalo jen o krátké otevření s délkou asi 5 sekund) a 16 hodin od začátku mření (v tomto posledním případě otevření trvalo přibližně dvojnásob co předchozí).

Zařízení bylo nejprve zapnuto zároveň s LabVIEW aplikací a následně umístěno do lednice kde bylo ponecháno po celou dobu měření. Po umístění zařízení bylo zapnuto zaznamenávání hodnot a před vyjmutím zařízení z lednice bylo ukládání v aplikaci zastaveno. Pro měření při nízké teplotě byla jako korekční koeficient použita hodnota 3.5 °C.



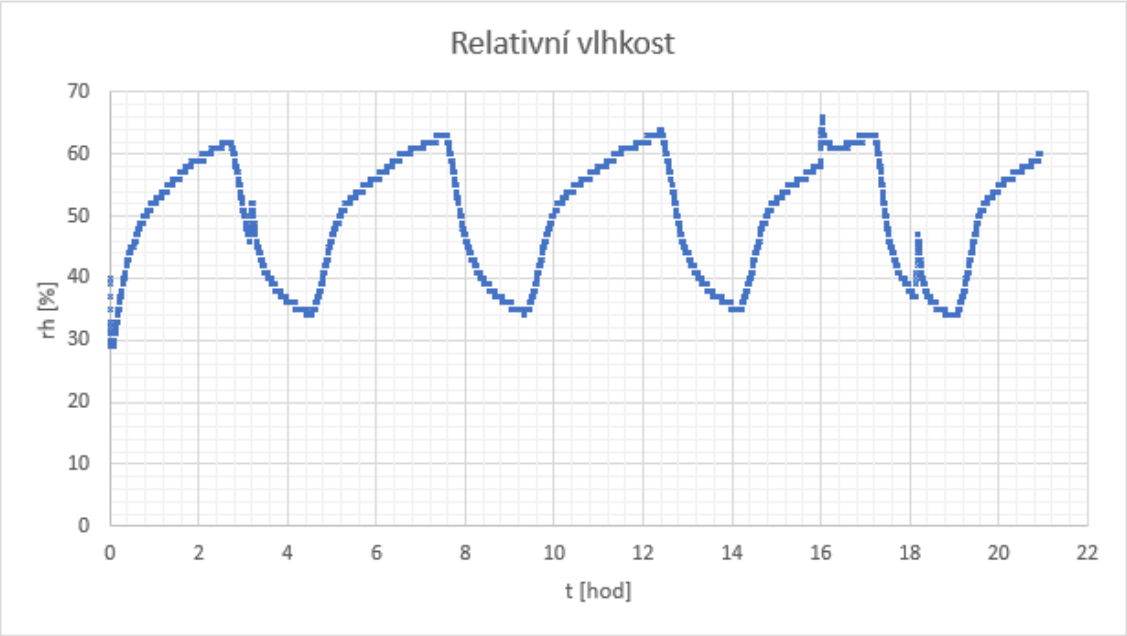

Obrázek 5.5 Graf měření teploty v lednici

Na Obrázek 5.5 je již z naměřených dat vytvořen graf získaných hodnot teploty. Zde je patrné, že snímači, po vložení z teplého (21.7 °C) do chladného prostředí s teplotou kolem 3 °C, trvá několik desítek minut, než se dostane na okolní teplotu. Na dalším průběhu je jasně vidět, jak teplota osciluje kolem přibližně 3.5 °C a taktéž je patrná šířka hystereze asi 4 °C. Výkyvy způsobené krátkodobým otevřením lednice v přibližných časech 3 hodiny a 18 hodin od začátku měření jsou jen velmi obtížně zpozorovatelné. Lépe viditelnější je zde delší otevření v čase 16 hodin od začátku měření, kde dochází sice k malému, ale přesto viditelnému prudšímu nárůstu teploty.




Obrázek 5.6 Graf měření okolního tlaku v lednici

Další zaznamenávanou veličinou, jak bylo předešle zmíněno, je tlak. Získaný soubor dat byl opět zpracován do grafu, který je možné vidět na Obrázek 5.6. Hodnoty se pohybují přibližně mezi 956 hPa a 959 hPa, takže tato veličina není příliš proměnná, což bylo ovšem očekáváno. Lze také vidět že hodnoty ze snímače mají určitý rozptyl v řádu několika desetin hPa.



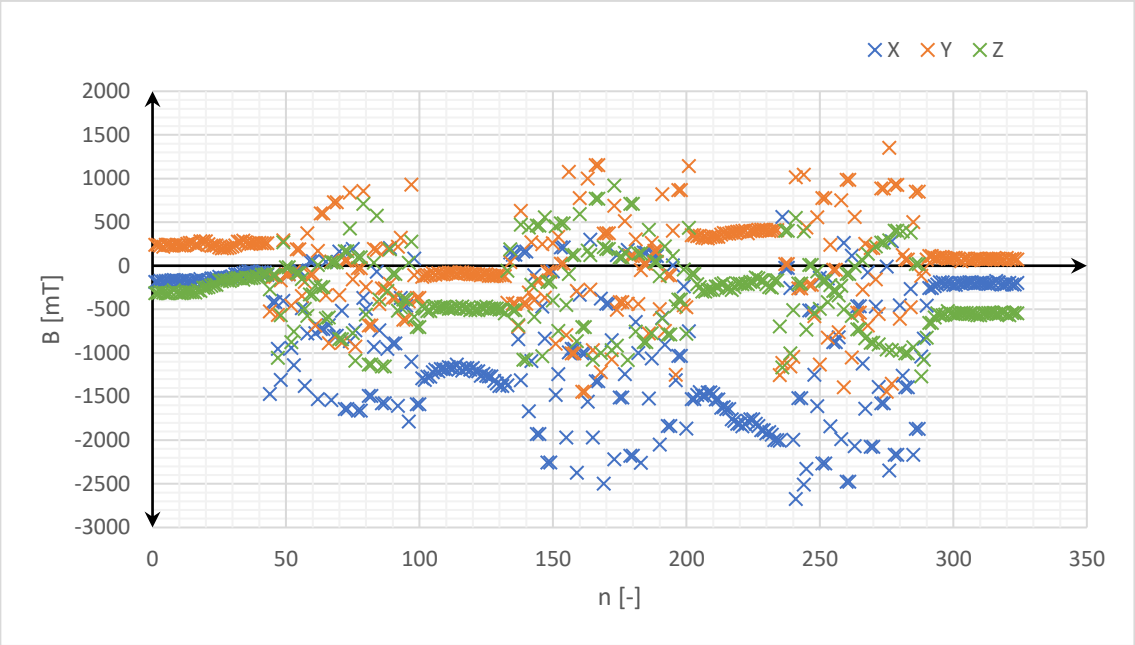

Obrázek 5.7 Graf měření relativní vlhkosti v lednici

Poslední měřenou veličinou byla relativní vlhkost, jak je vyobrazeno v grafu na Obrázek 5.7. Při porovnání tohoto grafu s grafem průběhu teploty na Obrázek 5.5, lze vidět že data shodně oscilují, tedy při stoupající teplotě stoupá i relativní vlhkost a naopak. Na této veličině je lépe vidět, kdy byla lednice otevřena. Krátká otevření jsou zde mnohem lépe patrná než u grafu snímače teploty, kde je lze jen velmi těžce identifikovat.

Na závěr testu lze zhodnotit že zařízení pracovalo dle předpokladů, tedy sběr dat ze snímačů a bezdrátový přenos do LabVIEW aplikace, kde byla data přijata a uložena, fungoval bez problémů. Byly proměřeny teplota, tlak a vlhkost a získaná data byla zpracována do grafů na obrázcích Obrázek 5.5 ažObrázek 5.7. Z výsledného vyplynulo, s jakým rozptylem je regulována teplota v konkrétní lednici. Dále by bylo možné na základě enviromentálního snímače například detekovat otevření lednice, a to nejlépe pomocí snímače relativní vlhkosti.

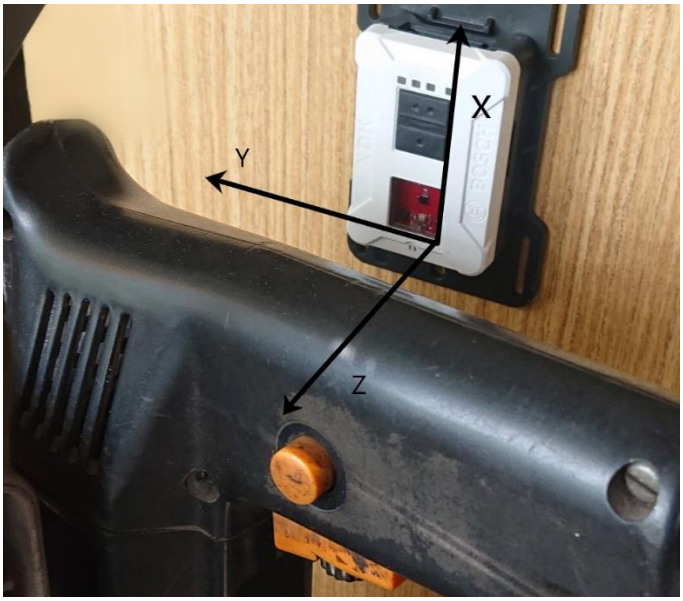
## 5.4 Měření magnetické indukce a hluku

Pro otestování magnetometru byl jako zdroj magnetické indukce použita vrtačka s elektromotorem a zároveň s tím mohl být otestován snímač hluku. Zařízení XDK 110 bylo umístěno do blízkosti vrtačky, která je nehybně upevněna ve stojanu a byly prozkoušeny tři stupně rychlosti otáček motoru. Tímto měřením bylo ověřeno, že pomocí výsledné aplikace je možné měřit magnetické pole či úroveň hluku, v místě, kde bude zařízení umístěno. Pomocí XDK 110 byly zaznamenány okolní podmínky při měření: teplota 18.6 °C, tlak 969.78 hPa a relativní vlhkost 49 %.



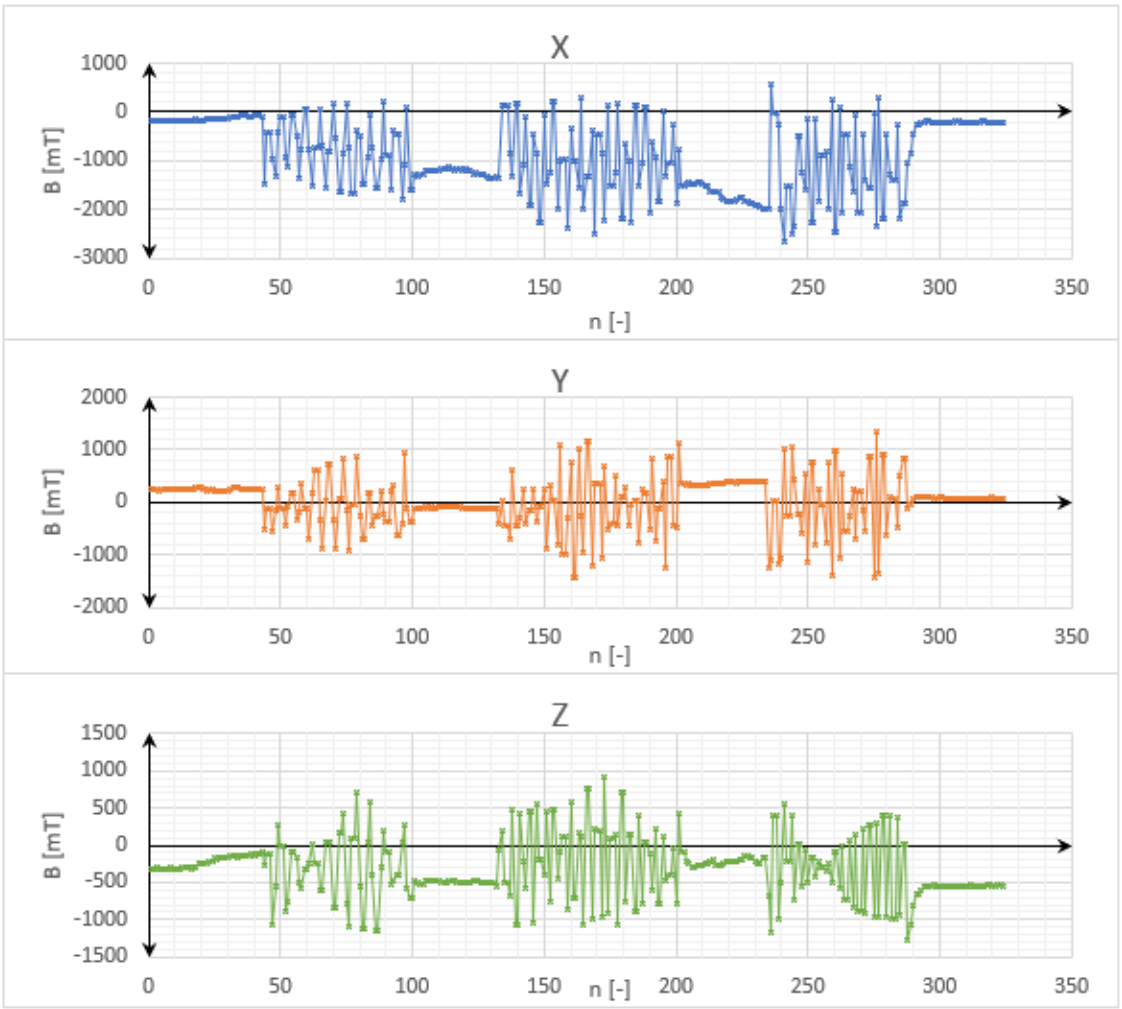

Obrázek 5.8 Graf měření magnetické indukce (společný graf pro všechny osy)

Zařízení XDK 110 bylo zavěšeno na dřevěnou stěnu skříně ve vzdálenosti (zařízení) asi 8 cm od vrtačky, která je upevněná ve stojanu (Obrázek 5.9). Zařízení směřovalo osou Z směrem k vrtačce, osou X směrem vzhůru a osa Y směřovala doleva při pohledu od vrtačky směrem k XDK 110.



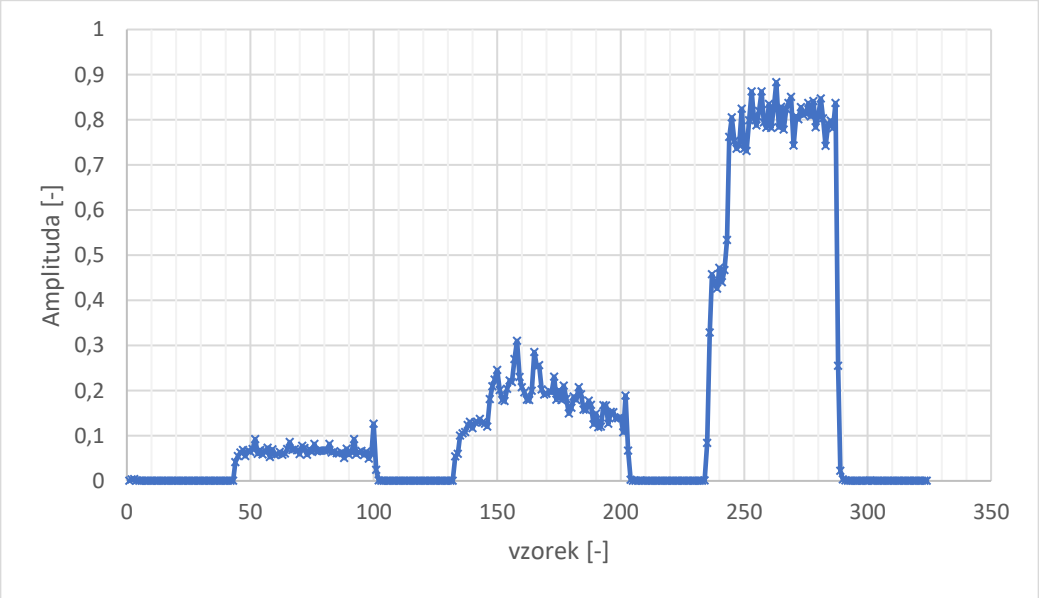

Obrázek 5.9 Měření magnetické indukce a hluku

Pro měření byla nejprve provedena úprava firmwaru, aby bylo možné měřit jak magnetickou indukci, tak také úroveň hluk najednou. Oba použité snímače byly vyčítány se stejnou frekvencí, stačilo tedy sloučit módy tři a čtyři přepsáním jedné podmínky. Poté bylo zařízení umístěno do blízkosti elektromotoru, který byl během jednoho měření postupně spuštěn ve třech různých rychlostech vzestupně. Všechny hodnoty z měření, tedy tři osy magnetometru a úroveň hluku byly zaznamenávány v LabVIEW aplikaci. Data byla vyčítána a odesílána ze zařízení s periodou 50 ms.




Obrázek 5.10 Grafy měření magnetické indukce (osy odděleně)

Výsledná data z měření byla zpracována do grafů. První graf je vykreslen na Obrázek 5.8, kde jsou uvedeny všechny tři osy magnetometru pro jejich vzájemné porovnání a vidíme zde závislost magnetické indukce, kde na ose x je vždy uvedeno číslo vzorku. Stejnou závislost lze vidět na Obrázek 5.10, kde jsou vyobrazeny tři grafy, pro každou osu zvlášť, aby byla zvýšena přehlednost jednotlivých průběhů. Na grafech je jasně vidět, kdy byl motor spuštěn (magnetická indukce v té chvíli začne oscilovat), a kdy motor nebyl v pohybu (průběh v klidové poloze). Lze také pozorovat jev, kdy mezi spuštěními motoru hodnoty sice neoscilují, ovšem jejich poloha je posunuta na rozdíl od klidových hodnot na začátku měření. Obzvláště je to patrné u osy X, kde jsou i naměřené hodnoty při spuštěném motoru výrazně zápornější než u ostatních os, což může být způsobeno tím, že magnetické pole převážně působilo pod zařízením, tedy v záporném směru osy X. U os X a Z je možné si také všimnout mírného nárůstu před spuštěním motoru, což mohlo být způsobeno při nastavování zařízení pomocí tlačítek, kdy se ruka obsluhy nacházela mezi zařízením a vrtačkou a mohla tak fungovat jako stínění před kovovými částmi krytu vrtačky. Vzhledem k pomalé rychlosti vzorkování nelze z naměřených dat získat přesný průběh magnetické indukce, která vzniká při chodu elektromotoru, a vykreslené grafy jsou tak spíše pro demonstraci funkčnosti snímače.



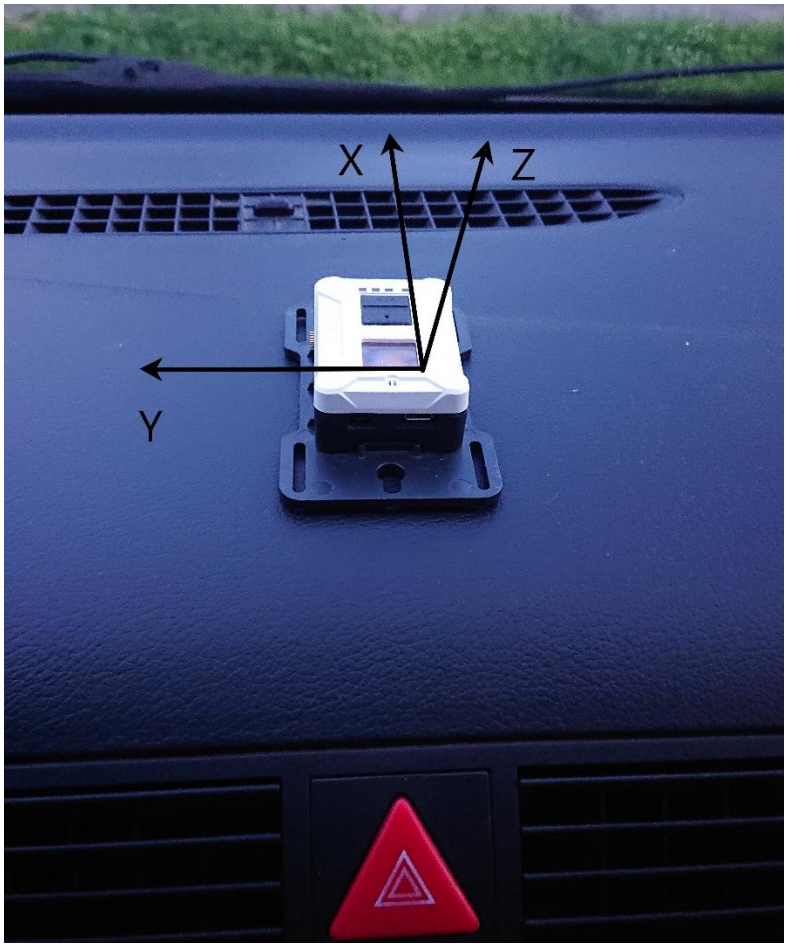

Obrázek 5.11 Graf výstupu snímače hluku

Jak bylo uvedeno dříve, s magnetickou indukcí byla měřena i úroveň hluku a výsledky byly zaznamenány do grafu na Obrázek 5.11, kde hodnoty sbírané ze snímače hluku jsou bezrozměrné.

Při porovnání výsledných grafů z obrázků Obrázek 5.8 aObrázek 5.11 je vidět, že grafy korespondují, tedy při zaznamenání hluku bylo zaznamenáno i magnetické pole. Jak bylo zmíněno dříve, testování probíhalo pro tři různé rychlosti otáčení. U grafů magnetické indukce nelze příliš rozpoznat rychlost otáčení, ovšem u grafu s výstupem snímače hluku je jasně rozlišitelná úroveň hodnot, kdy při nejpomalejší rychlosti je amplituda nejmenší, a naopak při nejvyšší rychlosti je amplituda nejvyšší, což je očekávané.

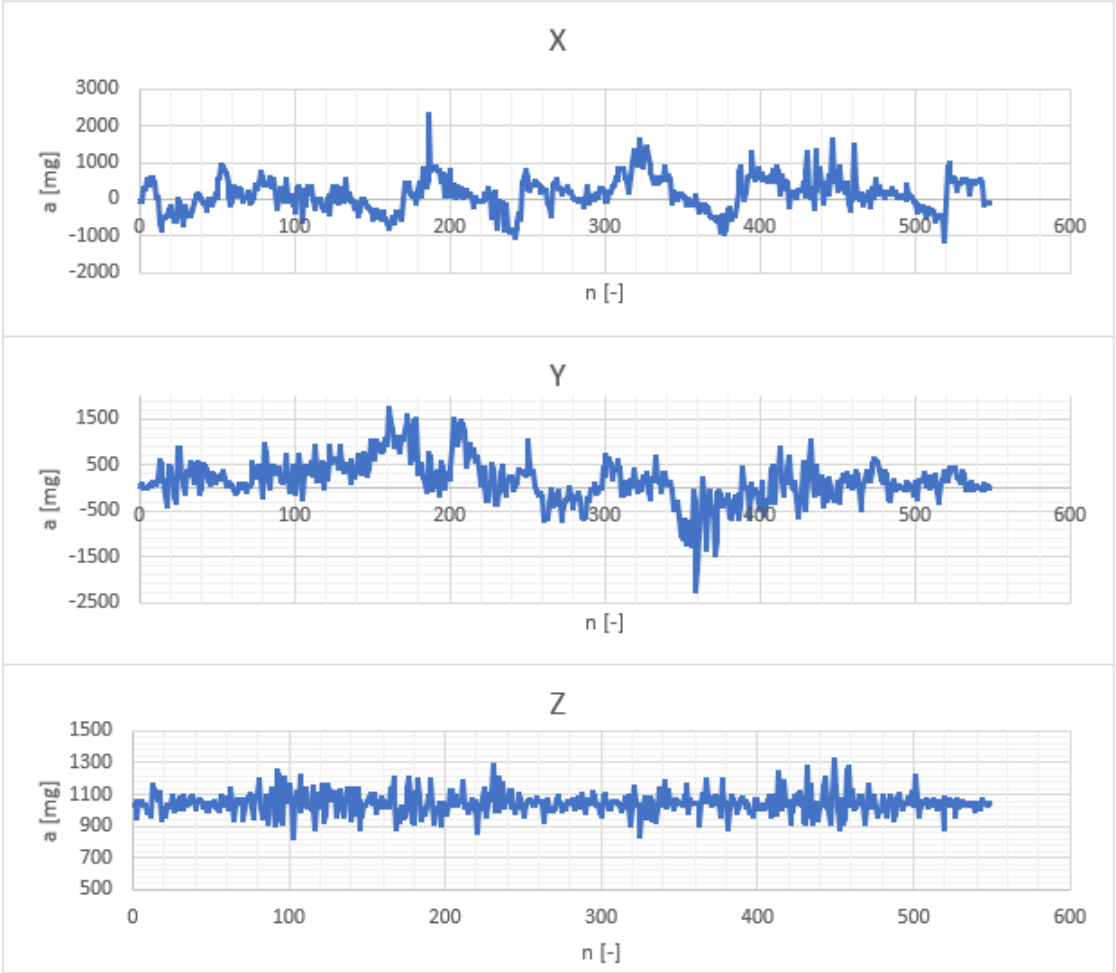
## 5.5 Měření zrychlení

Jako poslední byl testován akcelerometr, opět v praktickém měření. Zařízení Bosch XDK 110 bylo připevněno na palubní desku auta (Obrázek 5.12) oboustrannou lepící páskou přibližně v rovině, kdy osa X směřovala po směru jízdy vpřed, osa Y směřovala vlevo při pohledu po směru jízdy vpřed a osa Z směřovala vzhůru. Pomocí XDK byly také zaznamenány okolní podmínky měření jako teplota 17.8 °C, tlak 969.93 hPa a 47 % relativní vlhkosti.




Obrázek 5.12 Zařízení připevněné na palubní desce

Testován zde byl také chod zařízení, kdy není dostupná bezdrátová síť a hodnoty jsou ukládány na SD kartu. Zde se ovšem vyskytl problém. Pokud je volána funkce pro odesílání dat a zároveň není dostupná žádná síť, zařízení je zacykleno ve funkci pro znovu připojení k síti a není možné jej ovládat, tedy přepínat módy pomocí tlačítek. Lze to vyřešit tak, že měření se zahájí v místě, kde je síť dostupná. Pomocí tlačítek je navolen mód 5 kde jsou data ukládána na SD kartu a zároveň není používána WLAN a následně je možné zařízení použít i mimo dosah sítě. Další možností by bylo přidání módu, který by nic nevykonával a ze kterého by bylo možné přepnout přímo ukládání dat na SD kartu. Takto by bylo možné zahájit měření i mimo signál WLAN. V tomto měření mohlo být měření spuštěno v dosahu, a tak nemusel být firmware dále upravován.




Obrázek 5.13 Zrychlení v osách X, Y a Z

Jak bylo zmíněno výše, zařízení bylo připevněno na palubní desku a po jeho zapnutí a nastavení byla zahájena jízda. Po projetí dráhy bylo měření ukončeno a zařízení vypnuto a data byla později vyčtena z SD karty. Event pro ukládání dat na SD kartu byl volán periodicky každé 2 ms, tudíž frekvence vzorkování se dá přibližně určit jako 500 Hz.

Hodnoty byly vykresleny do grafů, ale byl zde značný šum, a proto byl použit jednoduchý filtr pro zvýšení přehlednosti průběhu. Tato data již byla vykreslena do grafů, které lze vidět na Obrázek 5.13.

V tomto obrázku je možné vidět průběhy pro všechny tři měřené osy. Je vidět že zrychlení v osách X a Y se přibližně pohybují kolem nuly, kdežto zrychlení v ose Z se pohybuje kolem hodnoty 1000 mg. Toto je způsobeno polohou snímače, jak bylo uvedeno dříve, že osa Z směřuje vzhůru, a tudíž je neustále ovlivňována vlastní hmotností.

# ZÁVĚR

V kapitole číslo 1 bylo popsáno zařízení Bosch XDK 110, včetnš jeho senzorů a možností komunikace. Na základě porovnání vlastností jednotlivých možností bezdrátové komunikace, které zařízení XDK 110 nabízí a požadavků, které by měli splňovat, bylo vybráno rozhraní WLAN. Bylo také porovnáno několik protokolů, ze kterých byl zvolen MQTT. Byl také zvolen programovací jazyk pro firmware jako Eclipse Mita. Vše je blíže popsáno v kapitole 2. V následující kapitole číslo 3 je blíže popsán návrh celkového softwaru, tedy firmwaru pro zařízení a LabVIEW aplikace. Návrh firmwaru je vyobrazen na Obrázek 3.2 a Obrázek 3.3 a aplikace na Obrázek 3.4 a zároveň je celkový návrh na Obrázek 3.1.

Zmíněný návrh, byl implementován, jak je podrobně popsáno v kapitole 4. Výsledný software obsahuje požadované funkce, jako vyčítání dat ze snímačů a jejich následné bezdrátové odesílání do LabVIEW aplikace na PC nebo ukládání na SD kartu a možnost následného bezdrátového odeslání těchto uložených dat. Snímače, které byly použity jsou akcelerometr, snímač teploty, tlaku a vlhkosti, dále magnetometr, senzor intenzity osvětlení a jako poslední senzor hluku. Software postrádá možnost bezdrátové konfigurace XDK 110 z LabVIEW aplikace, což bylo nahrazeno možností ovládání pomocí tlačítek dostupných na zařízení. Diagram výsledného firmwaru je možné vidět na Obrázek 4.13 a kód je uveden v příloze B.

Během implementace bylo zjištěno, že nedokumentovanou vlastností použitého protokolu MQTT je zde rychlostní limit odesílání dat ze zařízení, takže aby byl zajištěn jejich plynulý tok, je maximální používaná frekvence odesílání 20 Hz. Aby bylo možné získávat data i s vyšší frekvencí, je využita možnost ukládat data na SD kartu, kde je při záznamu možné dosáhnout rychlosti stovek Hz.

V LabVIEW aplikaci jsou přijatá data zobrazována na front panelu, kde je možné i jednotlivé průběhy vykreslovat do grafu a případně je ukládat zároveň s časovou značkou přiřazenou v aplikaci.

V poslední kapitole číslo 5 byla otestována výsledná implementace. Nejdříve byly zjištěny některé vlastnosti zařízení, jako výdrž baterie při více konfiguracích (rychlostech čtení a odesílání dat) a okolních teplotách nebo korekční koeficient pro snímač teploty, aby bylo možné měřit správnou hodnotu teploty. Výdrž baterie dosahovala při nastavení pomalejší periody kolem 36 hodin v závislosti na teplotě okolí a při rychlejší periodě cca 10 hodin. Po zjištění zmíněných vlastností byla testována funkčnost v různých praktických měřeních. Mezi ně patří měření zrychlení v autě, měření teploty, tlaku a vlhkosti při běžném provozu lednice nebo měření magnetické indukce a hluku v blízkosti vrtačky. Během testování bylo zjištěno, že aktuální verze firmware vyžaduje před přepnutím do módu s ukládáním na SD kartu dostupnou síť. Tento nedostatek je možné obejít přepnutím zařízení do daného módu v dosahu sítě.

Z měření vyplynulo že výsledná implementace je funkční a lze ji použít pro měření na dostupných snímačích ať už při bezdrátovém odesílání dat do LabVIEW aplikace, nebo při ukládání dat na SD kartu.

# LITERATURA


[1] *User Guide XDK110. Bosch [online*]. 61 [cit. 2021-12-21]. BCDS – XDK110-UG. Dostupné z: https://manualzz.com/doc/30098284/user-guide---bosch-xdk---bosch-connected-devices-and-solu...

[2] *BMA280: Digital, triaxial acceleration sensor* [online]. 119 [cit. 2021-11-17]. Dostupné z: https://www.bosch-sensortec.com/products/motion-sensors/accelerometers/bma280/#technical

[3] *BMG160: Digital,* triaxial gyroscope sensor [online]. 92 [cit. 2021-11-17]. Dostupné z: https://www.mouser.com/datasheet/2/783/BST-BMG160-DS000-09-1221199.pdf

[4] *BMM150: Geomagnetic sensor [online]*. 56 [cit. 2021-11-17]. Dostupné z: https://www.bosch-sensortec.com/products/motion-sensors/magnetometers-bmm150/

[5] *BME280: Combined humidity and pressure sensor [online*]. 56 [cit. 2021-11-17]. Dostupné z: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/

[6] *AKU340: Bottom port, analog silicon MEMS microphone* [online]. 16 [cit. 2021-11-17]. Dostupné z: https://cz.mouser.com/datasheet/2/720/DS26-1.04%20AKU340%20Datasheet-770074.pdf

[7] *MAX44009: Industry's Lowest-Power Ambient Light Sensor with* ADC [online]. 20 [cit. 2021-11-17]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/MAX44009.pdf

[8] BMI160: Small, low power inertial measurement unit [online]. *114 [cit. 2021-11-17]*. Dostupné z: https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi160/

[9] USB 2.0 Specification. *In: Usb.org [online]. [cit. 2021-12-21]. Dostupné z: https://www.usb.org*/document-library/usb-20-specification

[10] *Introduction to Bluetooth Device Testing: From Theory* To Transmitter and Receiver Measurements [online]. 2016, 119 [cit. 2021-12-21]. Dostupné z: http://download.ni.com/evaluation/rf/intro_to_bluetooth_test.pdf

[11] M.GROVER, , *S*.K.PARDESHI, N. SINGH a S. KUMAR. Bluetooth low energy for industrial automation. 2nd International Conference on Electronics and Communication Systems (ICECS) [online]. 4 [cit. 2021-12-21]. Dostupné z: doi:10.1109/ECS.2015.7124960

[12] The Bluetooth Range Estimator. In: Bluetooth.com [online]. [cit. 2021-12-21]. Dostupné z: https://www.bluetooth.com/learn-about-bluetooth/key-attributes/range/#estimator

[13] Introduction to Wireless LAN Measurements: From 802.11a to 802.11ac [online]. 42 [cit. 2021-12-21]. Dostupné z*:* http://download.ni.com/evaluation/rf/Introduction_to_WLAN_Testing.pdf

[14] TJENSVOLD, Jan Magne. Comparison of the IEEE 802.11, 802.15.1, 802.15.4 and 802.15.6 wireless *standards* [online]. 7 [**cit**. 2021-11-17]. Dostupné z: http://lars.mec.ua.pt/public/LAR%20Projects/SystemDevelopment/2016_RaquelAlves/wireless/comparison-ieee-802-standards.pdf

[15] Internet of Things (*IoT*) using LoRa technology. IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS) [online]. Selangor, Malaysia, 2019, 7 [cit. 2021-11-10]. Dostupné z: doi:10.1109/I2CACIS.2019.8825008

[16] XDK Lora extension. *In*: Conrad [online]. [cit. 2021-11-10]. Dostupné z: https://www.conrad.com/p/xdk-lora-extension-bosch-connected-devices-and-solutions0-273-600-072-001rf-transmitter-2148498

[17] Enable all Sensors in one. In: Bosch [online]. [cit. 2021-12-23]. Dostupné z: https://developer.bosch.*com/*products-and-services/sdks/xdk/develop/**c**/sensors/enable-all-sensors-in-one

[18] Modeling the Energy Performance of LoRaWAN [online]. 30 [cit. 2021-11-17]. Dostupné z: https://www.researchgate.net/publication/320435869_Modeling_the_energy_performance_of_LoRaWAN

[19] Comparative Study of Communication Interfaces for Sensors and Actuators in the Cloud of Internet of Things. International Journal of Internet *of Things* [online]. 2017, **6**(1), 9-13 [cit. 2021-11-17]. Dostupné z: doi:10.5923/j.ijit.20170601.02

[20] Eclipse Mita [online]. In: . [cit. 2022-05-01]. Dostupné z: https://www.eclipse.org/mita/

[21] RFC 2616: HTTP/*1*.1 [online]. 60 [cit. 2021-12-23]. Dostupné z: https://www.hjp.at/*doc/rfc*/rfc2616.html

[22] An improved UDP protocol for video transmission over Internet-to-wireless networks. IEEE Transactions on Multimedia [online]. 2001, 3(3), *10* [cit. 2021-12-23]. Dostupné z: doi:10.1109/6046.944478

[23] User Datagram Protocol [*online*]. 3 [cit. 2021-12-23]. Dostupné z: https://www.hjp.at/doc/rfc/rfc768.*html*

[24] Internet of Things: *Survey* and open issues of MQTT protocol. International Conference on Engineering & MIS (*ICEMIS)* [online]. 6 [cit. 2021-12-23]. Dostupné z: doi:10.1109/ICEMIS.2017.8273112

[25] MQTT [online]. *2022* [cit. 2022-05-12]. Dostupné z: https://mqtt.org/

[26] Eclipse Mita [online]. [cit. 2022-02-01]. Dostupné z: https://www.eclipse.org/mita/

[27] Eclipse Mosquitto: An *open* source MQTT broker [online]. [cit. 2022-02-27]. Dostupné z: https://mosquitto.org/

[28] MQTT. In: Bosch*:* Developer portal [online]. [cit. 2022-03-04]. Dostupné z: https://developer.bosch.com/*products-and-services*/sdks/xdk/develop/mita/protocols/mqtt

# SEZNAM PŘÍLOH

# Příloha A - Ukázky softwaru

## A.1 SendDataOverUdp (AppController.h)

```
/**
 * WLAN_CONNECT_WPA_SSID is the SSID of the WIFI network you want to
connect to.
 */
#define WLAN_SSID                       "YourWifiNetwork"
/**
 * WLAN_CONNECT_WPA_PASS is the WPA/WPA2 passphrase (pre-shared key) of
your WIFI network.
 */
#define WLAN_PSK                        "YourWifiPassword"

/**
 *  WLAN_USERNAME is the Enterprise WIFI network username(unused if
IS_ENTERPRISE_WIFI_SELECTED is false)
 */
#define WLAN_USERNAME                   "YourWifiUserName"

/**
 * IS_ENTERPRISE_WIFI_SELECTED is a boolean. If "true" then XDK device
will connect to Enterprise WIFI Network and If "false" then XDK device
will connect to Personal WIFI Network.
 */
#define IS_ENTERPRISE_WIFI_SELECTED     false

/**
 * DEST_SERVER_IP is the destination server IP address of the web server
we will send UDP payloads.
 * If you want to test this example without setting up your own server,
you can use publicly available services.
 */
#define DEST_SERVER_IP                  XDK_NETWORK_IPV4(0, 0, 0, 0)

/**
 * DEST_SERVER_PORT is the UDP port to which we will send UDP payloads.
 */
#define DEST_SERVER_PORT                UINT16_C(0)

/**
 * APP_UDP_TX_DELAY is the UDP packet transmit frequency in milli second.
 */
#define APP_UDP_TX_DELAY                UINT32_C(1000)
```

# Příloha B - Firmware

```
/**
* XDK Firmware
 * Author: Bc. Filip Vítek
 */

package main;
import platforms.xdk110;

/**********************************************************************
***********/
//          Global      Variables
/**********************************************************************
***********/

let tempCorrection = 3500; // correction factor for the XDKs heatoutput
var firmwareMode = 1;
var HelpVar = 0;
var dataOut : int32;
var fileBuffer : string = "";
var bufferFillSize = 0;

/**********************************************************************
***********/
//                Functions
/**********************************************************************
***********/

/* Checks lenght of number and adds zeroes before when writing to file
*/
/*All numbers have to be same lenght because of way of reading from file
*/
fn addToBuffer(Value: uint32){
      if (Value >= 1000){
            fileBuffer += (`${Value} \n`);
      }
      else if(Value >= 100){
            fileBuffer += (`0${Value} \n`);
      }
      else if(Value >= 10){
            fileBuffer += (`00${Value} \n`);
      }
      else if(Value >= 0){
            fileBuffer += (`000${Value} \n`);
      }
}

/* Sets & resets red and orange LEDs when using SD card */
fn SetLED(){
      if(firmwareMode == 5){
            if(!led.red.read()){
                  led.red.write(true);
            }
      }
      else{
            if(led.red.read()){
                  led.red.write(false);
            }
```

```
	}
	if(firmwareMode == 6){
		if(!led.orange.read()){
			led.orange.write(true);
		}
	}
	else{
		if(led.orange.read()){
			led.orange.write(false);
		}
	}
}

/*Converts string to number*/
/*Function from "string.h" library, converted to Mita using foreign
function inerface */
native unchecked fn atoi(str: string) : int32 header "string.h";

/**********************************************************************
***********/
//		Device & Sensors Setup
/**********************************************************************
***********/

setup led : LED {
	var red = light_up(Red);
	var orange = light_up(Orange);
}

setup accelerometer {
	bandwidth = BW_1000Hz;
	range = Range_8G;
}

setup SDc : SDCard{
	var WriteToFile : string = appendingTextWrite("Data.txt");
	var ReadFromFile : string = resumingTextRead("Data.txt", 6);
}

setup wifi : WLAN{
	ssid = 'MySsid';
	authentication = Personal(psk='MyPsk');
	ipConfiguration = Dhcp();
}

setup mqttXDK: MQTT {
  transport = wifi;
  url = "mqtt://192.168.0.108:1883";
  clientId = "XDK42";

  var accelerometer = topic("accelerometer");
  var accelerometer_x = topic("accelerometer_x");
  var accelerometer_y = topic("accelerometer_y");
  var accelerometer_z = topic("accelerometer_z");
  var HelpVarerature = topic("HelpVarerature");
  var pressure = topic("pressure");
  var humidity = topic("humidity");
  var power = topic("power");
  var version = topic("version");
```

```
  var fromSD = topic("fromSD");
  var magnetometer_x = topic("magnetometer_x");
  var magnetometer_y = topic("magnetometer_y");
  var magnetometer_z = topic("magnetometer_z");
  var noise = topic("noise");
  var light = topic("light");

}


/**********************************************************************
***********/
//                Events
/**********************************************************************
***********/

 every button_two.pressed {
      firmwareMode++;
      if (firmwareMode > 6){
            firmwareMode = 1;
      }
}

 every button_one.pressed {
      firmwareMode--;
      if (firmwareMode < 1){
            firmwareMode = 6;
      }
}

every 50 millisecond {
      if (firmwareMode == 1){
             let Accelerometer_X = accelerometer.x_axis.read();
             let Accelerometer_Y = accelerometer.y_axis.read();
             let Accelerometer_Z = accelerometer.z_axis.read();
             mqttXDK.accelerometer_x.write(`${Accelerometer_X}`);
             mqttXDK.accelerometer_y.write(`${Accelerometer_Y}`);
             mqttXDK.accelerometer_z.write(`${Accelerometer_Z}`);
             SetLED();
      }
}

every 5 seconds {
      if (firmwareMode == 2){
            mqttXDK.version.write(`${firmwareMode}`);
            let    HelpVar    =    (environment.temperature.read()    -
tempCorrection)  / 1000.0;
            mqttXDK.HelpVarerature.write(`${HelpVar}`);
            mqttXDK.pressure.write(`${environment.pressure.read()}`);
            mqttXDK.humidity.write(`${environment.humidity.read()}`);
            mqttXDK.light.write(`${light.intensity.read()}`);
      }
}



every 100 millisecond{
      if(firmwareMode == 1 || firmwareMode == 5 || firmwareMode == 6){
```

```
            mqttXDK.version.write(`${firmwareMode}`);
      }
      if(firmwareMode == 4){
            mqttXDK.version.write(`${firmwareMode}`);
            mqttXDK.noise.write(`${noise_sensor.noise.read()}`);
      }
      if(firmwareMode == 3){
            mqttXDK.version.write(`${firmwareMode}`);

      mqttXDK.magnetometer_x.write(`${magnetometer.x_axis.read()}`);

      mqttXDK.magnetometer_y.write(`${magnetometer.y_axis.read()}`);

      mqttXDK.magnetometer_z.write(`${magnetometer.z_axis.read()}`);

      }
}

every 2 millisecond{
      if (firmwareMode == 5){
            bufferFillSize++;
            HelpVar++;
            let Accelerometer_Mag = accelerometer.magnitude.read();
            addToBuffer(Accelerometer_Mag);

            if(bufferFillSize >= 100){
                  SDc.WriteToFile.write(`${fileBuffer}\n`);
                  fileBuffer = "";
                  bufferFillSize = 0;
            }
            if(HelpVar >=50){  // Delay to check if button is pressed
                  if(button_one.is_pressed.read()){
                        firmwareMode = 4;
                  }
                  else if(button_two.is_pressed.read()){
                        firmwareMode = 6;
                  }
                  HelpVar = 0;
            }
            SetLED();
      }
      if(firmwareMode == 6){
            HelpVar++;

      mqttXDK.fromSD.write(`${atoi(SDc.ReadFromFile.read())}`);
            if(HelpVar >=30){ // Delay to check if button is pressed
                  if(button_one.is_pressed.read()){
                        firmwareMode = 5;
                  }
                  else if(button_two.is_pressed.read()){
                        firmwareMode = 1;
                  }
                  HelpVar = 0;
            }
            SetLED();
      }
}
every 5 minutes {
      let ps = XDK110.powerStatus.read();
```

```
      where(ps) {
            is(PowerStatus.Battery -> level) {
            mqttXDK.power.write(`${level}`);
          }
          is(PowerStatus.Corded) {
                  mqttXDK.power.write(`100`);
          }
      }
}
```