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

UNIVERZÁLNÍ PLATFORMA PRO MĚŘENÍ INERCIÁLNÍCH A TLAKOVÝCH SENZORŮ

UNIVERSAL PLATFORM FOR MEASUREMENT OF INERTIAL AND PRESSURE SENSORS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Jan Usnul

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Petr Fiedler, Ph.D.

SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření** Ústav automatizace a měřicí techniky

Student: Bc. Jan Usnul ***ID:*** 155255
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Univerzální platforma pro měření inerciálních a tlakových senzorů

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte software pro univerzální měřicí platformu, který umožní měření a testování inerciálních a tlakových sensorů.

1. Formulujte podrobně požadavky na realizovaný SW - SW pro PC a SW pro embedded platformu
2. Navrhnětě strukturu SW pro PC
3. Navrhnětě strukturu SW pro embedded platformu
4. Navrhněte GUI pro SW pro PC
5. Navrhněte robustní komunikační protokol pro komunikaci mezi PC a embedded platformou
6. Seznamte se s embedded platformou, jejím vývojovým prostředím.
7. Popište skriptovací jazyk a způsob, jakým bude implementována realizace vykonávání skriptů.

V rámci navazující DP pak realizujte navržený SW a ověřte jeho funkčnost.

DOPORUČENÁ LITERATURA:

Katalogové listy sensorů.

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 15.5.2017

Vedoucí práce: doc. Ing. Petr Fiedler, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

Klíčová slova

embedded C, C#, csharp, univerzální měřicí jednotka, testovací platforma pro inerciální sensory, ISTP

Keywords

embedded C, C#, csharp, inertial sensor test platform, ISTP

Abstrakt

Práce se zabývá vývojem softwaru, pro embedded zařízení a řídicí počítač, který je schopen provádět automatické měřicí testy na inerciálních a tlakových sensorech. Software je vytvářen jak pro řídicí část (platforma Windows), tak pro samotnou platformu (embedded C). Práce popisuje vytvoření a aplikaci komunikačního protokolu, jednotlivé fáze programu, princip a podobu testovacích skriptů.

Abstract

Author describes the creation of software for the initial sensor test platform and controll computer. The platform can measure inertial and pressure sensors regardless their specific type. Measurement is provided by automatic test which is defined by the user. Also, the author describes the creation and implementation of communication protocols between control software (on Windows platform) and test platform (embedded C).

Bibliografická citace:

USNUL, J. *Univerzální platforma pro měření inerciálních a tlakových senzorů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 62s, 4s příloh. Vedoucí práce: doc. Ing. Petr Fiedler, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma univerzální platforma pro měření inerciálních a tlakových senzorů jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové 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: ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce doc. Ing. Petru Fiedlerovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

Děkuji odbornému konzultantovi Ing. Marku Fojtáchovi za možnost podílet se na interním projektu firmy Honeywell s.r.o., možnost vytvořit komplexní řešení pro testování inerciálních a tlakových sensorů s praktickým využitím napříč celou firmou a za jeho odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne:

……
podpis autora

Obsah

Seznam obrázků

Seznam Tabulek

Seznam Rovnic

1. ÚVOD

Důvodem vzniku požadavku na vytvoření univerzální platformy pro měření inerciálních a tlakových sensorů (ISTP) je neustálé testování nových sensorů v letectví. Požadavky na kvalitu sensoru v letadle se zvyšují, ale zároveň se čím dál častěji vyskytuje požadavek na snížení výrobních nákladů celého systémum, aby celý systém měl nižší náklady a výsledná cena mohla být také nižší je potřeba snížit náklady i na samotných sensorech.

Bezpečnost v letectví tkví v jeho velmi přísných požadavcích a požadavky na sensory nejsou výjimkou. Ovšem abychom mohli vyměnit drahé a přesné sensory za jejich levnější, a ne tak přesné varianty musíme mít definované testy, jejichž výsledky nám prozradí, zdali je levnější varianta sensoru dostatečně přesná pro použití v letectví. Náklady na testování jednoho sensoru zahrnují, kromě použitých periferií jako např. klimatické komory, náklady na výrobu speciální senzorové desky, plat inženýra, který takovou desku navrhne včetně vytvoření driverů pro sensor, a navíc vytvoření jednoduchého testovacího programu tak, aby měření bylo proveditelné a hlavně opakovatelné. Univerzální platforma pro měření inerciálních a tlakových sensorů nabízí komplexní hardwarové a softwarové řešení, takže při testování nového sensoru je zapotřebí jen vytvoření driverů a jeho usazení na senzorovou desku, vše ostatní zajišťuje platforma.

Cílem mojí práce je navržení a vytvoření softwaru, který umožní provádět automatické testy bez ohledu na testované sensory. Software (SW) je rozdělen do dvou částí, z nichž první bude vytvoření jádra programu v jazyce C pro embedded platformu. Druhou částí je kompletní řídicí software pro platformu Windows psaný v jazyce C#, který bude nejen řídit embedded platformu, ale navíc ještě veškeré ostatní potřebné periferie: klimatickou komoru, rotační stůl a tlakový generátor.

Hardware (HW) potřebný k této práci vychází z diplomové práce Ing. Maximiliána Tydora [1], který provedl jeho kompletní návrh a realizaci. Od jeho vzniku prošel HW drobnou úpravou, ale princip a myšlenka popsaná v jeho diplomové práci (DP) zůstala.

Od této práce očekávám dvě výzvy. První bude rozpoznávání připojených sensorů. Softwarově se ke každému sensoru musí přistupovat jinak. Každý sensor má vlastní vnitřní registry, jejichž obsah je seřazen dle výrobce. Rozmístění dat v registru je stejné jen v rámci konkrétního typu sensoru od konkrétního výrobce. Druhou výzvou bude samotný počítačový (PC) software, protože bude muset za definovaný čas obsloužit veškeré periferie, a navíc ještě zvládat zpracování dat v reálném čase a jejich ukládání na disk.

Tato práce je vytvořena na základě požadavků od firmy Honeywell s.r.o. v rámci mé stáže v této firmě, proto jsou sensory u výsledků měření označeny obecně a nejsou uvedeny jejich konkrétní názvy.

2. ROZBOR ZADÁNÍ

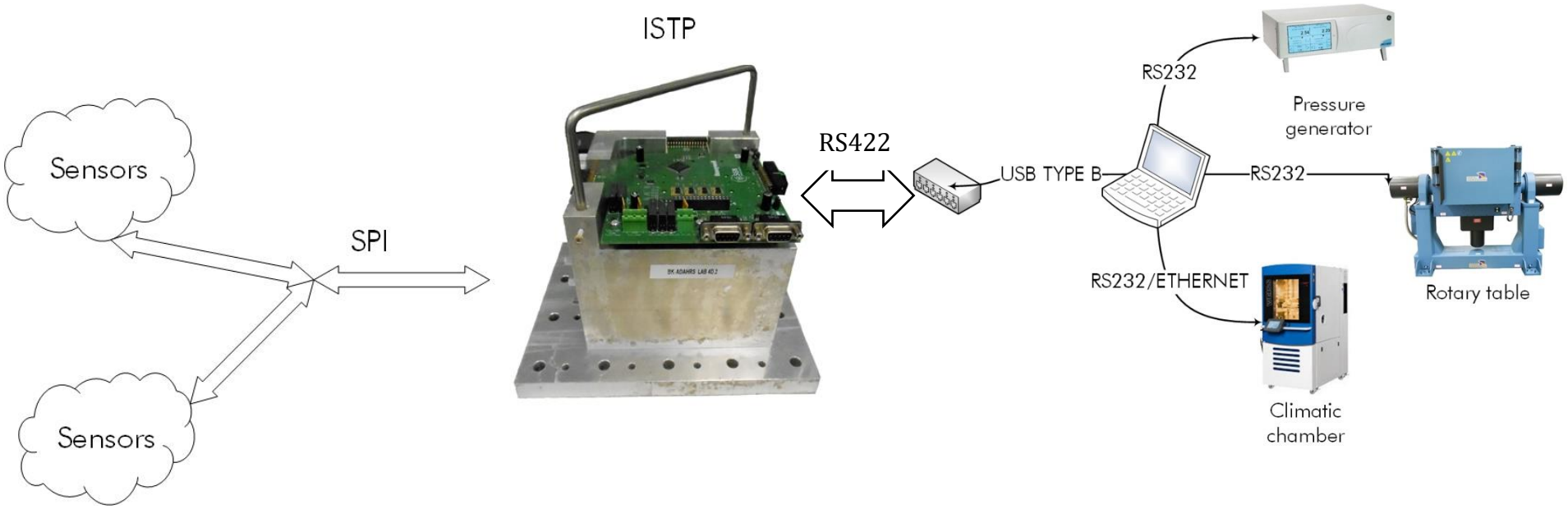
V rámci této práce je navržen a vytvořen software pro testování inerciálních sensorů. Požadavky na SW jsou rozděleny na dvě části – pro PC a pro ISTP. PC část bude mít veškeré rozhodovací funkce, aby bylo možné ISTP ovládat bez fyzického zásahu obsluhy. Oproti tomu embedded část bude muset sama rozhodnout jaké sensory jsou k ní připojené, a tedy jaký driver má použít pro komunikaci s nimi.

2.1. POŽADAVKY NA SW PRO PLATFORMU WINDOWS

I. PC SW musí být nezávislý na sensorech a datech z nich. SW bude jen zprostředkovávat názvy sensorů pro obsluhu, protože každý sensor má různé režimy nastavení, SW musí umět tyto režim zobrazit uživateli, dát mu možnost jejich změny a poté tuto změnu poslat do embedded části, která nastavení sensoru upraví. Tato část bude psána v jazyce C#.
II. Dalším požadavkem je vykonávání automatických testů dle požadavků uživatele. Testy budou definované ve formátu *.csv. Musí mít jednoduchou a přehlednou formu.
III. Přicházející data budou přijímána ve formátu single-precision z ISTP. ISTP musí provést přepočet ‚raw' dat ze sensorů na formát single-precision a až poté je odeslat do PC.
IV. PC bude schopno provádět zpracování dat v reálném čase na přijatých datech.
V. SW bude schopen umět spustit externí skripty a předat jim vstupní parametry potřebné k jejich správnému fungování.
VI. Uživatelské prostředí (GUI) bude účelového typu a zároveň obsluha bude vědět o stavu právě probíhajícího testu.
VII. PC aplikace umožní přidávání driverů pro periferie s minimálním nebo bez zásahu do jádra programu.

2.2. POŽADAVKY NA SW PRO EMBEDDED PLATFORMU

I. Software pro embedded platformu by měl být realizován jako samostatná jednotka, která pouze čeká na příkaz z PC.
II. ISTP bude schopno rozpoznat sensor, nakonfigurovat jej a číst z něj data.
III. Čtecí frekvence ze sensoru, stejně jako frekvence posílání dat do PC bude zcela na volbě uživatele.
IV. Při změně konfigurace sensoru se tato změna zaznamená a bude použita i při znovu připojení stejného sensoru.
V. ISTP bude označovat svůj aktuální stav a příslušnou chybu nahlásit nadřazené PC aplikaci.
VI. SW by měl umožnit přidávání driverů pro nové senzory bez většího nebo žádného zásahu do jádra programu.



Obrázek 2.1 Zobrazení zadání DP

3. POPIS PERIFERIÍ

3.1. EMBEDDED PLATFORMA

Software pro embedded platformu je psán v jazyce C, realizován na mikrokontroleru ATSAM4SD32C, který je založen na architektuře ARM® Cortex-M4 s frekvencí 120 MHz, která je pro požadovanou aplikaci dostačující. [2], [1]

3.1.1. HARDWARE

Použitý mikrokontroler ATSAM4SD32C obsahuje hardwarovou jednotku Cyclic Redundancy Check Calculation Unit (CRCCU) pro počítání Cyclic Redundancy Check (CRC), které je využito jako verifikace zpráv mezi ISTP a PC.

S PC aplikací se komunikuje přes RS422 s převodem na USB, ke kterému je použit externí převodník. Velikost datového pole je omezena na 512 hodnot datového typu single-precision dle IEEE 754-2008 [5]. Pokud provedeme převod na byty dostaneme 2048 B. Velikost vyrovnávací paměti konvertoru má přímý vliv maximální možnou odesílací frekvenci datových paketů stejně jako počet připojených sensorů.

Protože pro validaci inerciálních senzorů je ideální číst data na co největší frekvenci, využije se tzv. FIFO režimu sensorů. Většina inerciálních senzorů má interní paměť typu FIFO, do které je senzor schopen ukládat data na vysoké frekvenci. Mikrokontroler si vyčte všechny hodnoty z této paměti a odešle je jako speciální datový paket, to umožňuje, aby senzor aktualizoval své data na vysoké frekvenci, ale mikrokontroler je četl na frekvenci mnohem menší. Nastavení tohoto režimu je globální pro celou platformu a je volbou uživatele. [2],[3]

3.1.1.1. SPI

Serial Peripheral Interface (SPI) je sériová sběrnice pro komunikaci mezi dvěma zařízeními a je založena na principu Master-Slave. Na ISTP je MCU použito jako Master a jednotlivé sensory jako Slave [1].

Mikrokontrolér nejdříve vybere určitý sensor prostřednictvím Slave Select (NSS), poté začne poskytovat hodinový signál na lince Serial Clock (SPCK). Domluva v rámci této DP je taková, že s jednotlivými sensory se bude vždy komunikovat na jejich maximální podporované frekvenci, aby byl embedded SW schopen přečíst data s co nejmenším zpožděním. K přenosu samotných dat jsou použity linky Master In Slave Out (MISO) a Master Out Slave In (MOSI). [2]

3.2. KLIMATICKÁ KOMORA

Klimatická komora je po platformě nejdůležitější periferií. V rámci testování a kalibrace je zapotřebí mít neustálí přehled o aktuální teplotě okolí a pro určité testy i dodržení stále změny teploty okolí. O řízení teploty a vlhkosti okolí se stará klimatická komora, ve které je výrobek neprodyšně uzavřen. Takových komor je na trhu celá řada

při čemž platí, že čím rychlejší a stabilnější regulace, tím je komora dražší. Na druhou stranu v takových komorách se dokáže simulovat prostředí z celého světa, a proto je jejich využití, při testování sensorů pro letectví, klíčové.

3.2.1. KLIMTICKÉ KOMORY OD FIRMY WEISS

Klimatické komory od výrobce WEISS se vyznačují velmi kvalitní regulací teploty a vlhkosti. Ke komunikaci se používá jazyk ASCII 2, který se snadno implenetuje a je více popsán v [27].

Obrázek 3.1 Ukázka klimatické komory od firmy WEISS [20]

3.2.2. KLIMATICKÉ KOMORY KOMUNIKUJÍCÍ SKRZE PROTOKOL MODBUS

MODBUS je protokol vytvořený firmou Modicon® pro sériovou komunikaci mezi zařízeními. Publikovaný byl v roce 1979 pro použití s programovatelnými logickými kontroléry (PLC). Samotný protokol definuje řadu funkcí spolu s jejich vlastnostmi a použitím stejně jako dobu čekání mezi odesláním požadavku a přijetím odpovědi, časové mezery mezi zprávami a další vlastnosti komunikace. [9]

3.3. TLAKOVÝ GENERÁTOR

V rámci testování tlakových sensorů je nutné mít k dispozici tlakový generátor, který je možné vzdáleně ovládat a který současně umí regulovat tlak na požadovanou hodnotu. Takové požadavky splňuje tlakový generátor PACE6000 od firmy General Electric. V rámci mé práce mi je poskytnuta dvou kanálová verze, která umí regulaci tlaku na každém kanálu zvlášť. [24]

Tlakové generátory PACE6000 jsou plně modulární a jdou skládat na sebe, aby bylo možné jejich zasazení do měřicí stanice. Takové požadavky nemáme, protože při testování tlakových sensorů nám stačí jen jedna jednotka PACE6000. PC komunikuje s PACE6000 skrze RS232 za pomocí SCPI protokolu (pro více informací o SCPI protokolu

[23]) [24]. V rámci vzdáleného nastavování požadovaného tlaku není zapotřebí hlubšího studování protokolu SCPI. Přístroj nám byl dodán již s provedenou kalibrací.

Obrázek 3.2 Tlakový generátor PACE6000 [24]

3.4. ROTAČNÍ STŮL

Rotační stůl je od firmy Ideal Aerosmith, Inc. a je zobrazen na obrázku 3.3. Tato firma dodala firmě Honeywell s.r.o. kompletní systém na testování a kalibraci inerciálních sensorů, která obsahuje nejenom dvouosý rotační stůl, ale také kontrolér AERO 4000 a navíc ještě klimatickou komoru. PC SW, tak skrze kontrolér ovládá rotační stůl a klimatickou komoru zároveň. Po hardwarové stránce má kontrolér dva výstupy RS232, kde jeden slouží pro komunikaci se samotným kontrolérem a druhý pro komunikaci s klimatickou komorou. Z pohledu PC aplikace se tak jedná o dva různé komunikační porty a chová se k nim jako k samostatným periferiím. Vnitřně je v programu udělaná výjimka pro zapnutí/vypnutí komory, protože tato operace se musí provést skrze kontrolér. Kontrolér slouží jen jako aktivní propojovač externího operátora s klimatickou komorou a z tohoto důvodu tato linka je nefunkční, pokud je komora ve vypnutém stavu.

Obrázek 3.3 Ukázka rotačního stolu od firmy Ideal Aerosmith [8]

Samotný kontrolér má vlastní grafické rozhraní s vlastní externí klávesnicí a myší. Obsluha může nastavovat veškeré potřebné parametry pro vnější, případně vnitřní, osu. Parametry, které jsou zajímavé pro testování a kalibraci inerciálních sensorů jsou: Poloha os ve stupních od domovské pozice s definovanou rychlostí změny a rotace vnitřní osy ve stupních za minutu. Jak již bylo zmíněno, tak komunikace s kontrolérem probíhá přes

RS232 s přenosovou rychlostí 9,600kbit/s [8]. Před započetím řízení, přes PC aplikaci, je zapotřebí uvést kontrolér do provozu dle provozního manuálu.

Obrázek 3.4 Řídicí jednotka AERO4000 [8]

Samotná komunikace s AERO 4000 je skrze jazyk Aerosmith Table Language (ATL). Jedná se o sérii jednoduchých textových příkazů, které jsou posílány směrem do kontroléru, ten je plní v pořadí, v jakém mu příkazy přijdou. [7]

4. POPIS INERCIÁLNÍCH SENSORŮ

V této kapitole je popsán princip inerciálních sensorů typu akcelerometr a gyroskop, pro jejichž měření tento měřicí systém vzniká. Společnost Honeywell se soustředí na sensory založené na technologii MEMS, a proto zde popisuji jen sensory s touto technologií.

4.1. KAPACITNÍ SENSORY

Akcelerometry a gyroskopy založené na technologii MEMS měří pohyb závaží kapacitně, jedná se tak o deskový kapacitní snímač s proměnou plochou překrytí. Mezi plochami kondenzátoru je elektroda, která se pohybuje v jedné ose a tím dochází ke změně kapacity, která je přibližně lineární [18].

$$\frac{\Delta C}{\Delta l} \doteq - \frac{C_{max}}{l_{max}} [1 + \left(\frac{\Delta d}{d}\right)^2] \tag{4.1}$$

Rovnice 4.1 Funkční vztah pro deskový kapacitní snímač s proměnnou plochou překrytí [16]

Další možností je výpočet kapacity sensoru za předpokladu, že mezi elektrodami je homogenní pole a skutečnosti, že kapacita mezi dvěma objekty je určena rozdílem jejich potenciálů a mají vzájemně vázaný náboj. V takovém případě nezáleží na přítomnosti ostatních vodivých objektů v sousedství [16].

$$C_{ij} = \frac{Q_{ij}}{U_i - U_j} \tag{4.2}$$

Rovnice 4.2 Výpočet kapacity mezi dvěma objekty mezi nimiž je homogenní pole [16]

4.2. AKCELEROMETRY

Akcelerometry založené na MEMS technologii využívají kapacitního principu, díky kterému se každý nepatrný pohyb elektrod projeví změnou kapacity. Další výhodou jsou jeho malé rozměry. Nevýhodou je závislost na okolní teplotě.

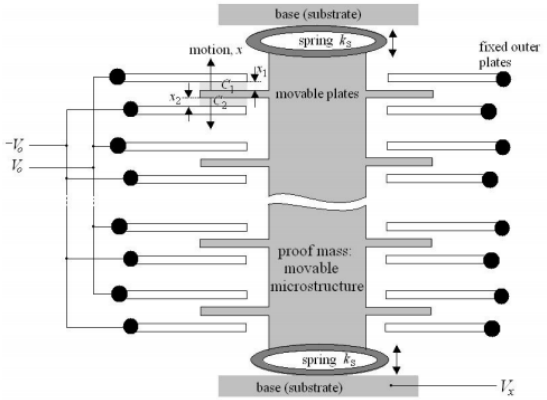
$$C = \varepsilon_0 . \varepsilon_r \frac{A}{d} [F] \tag{4.3}$$

Rovnice 4.3 Výpočet kapacity pro kapacitní akcelerometr [25]

Kapacitní akcelerometr je založený na změně vzdálenosti mezi elektrodami. Jako dielektrikum se používá vzduch s kombinací s plynem, protože použití vzduchu samotného by sice znamenalo navýšení odolnosti vůči okolní teplotě, ale i navýšení piezoresistivity. MEMS akcelerometry jsou složeny z nepohyblivé a pohyblivé části, která je upevněna na rám za pomocí pružin. Obě části reprezentují kapacitory, jejichž vzájemná výchylka je měřena za pomocí diference jejich hodnot. [25]

$$C_1 = \varepsilon_0 . \varepsilon_r \frac{A}{d+x} = C_0 - \Delta C, \qquad C_2 = \varepsilon_0 . \varepsilon_r \frac{A}{d-x} = C_0 + \Delta C$$
$$C_2 - C_1 = 2\Delta C = 2\varepsilon_0 \varepsilon_r A \frac{x}{d^2 - x^2} \tag{4.4}$$
$$\Delta C x^2 + \varepsilon_0 \varepsilon_r A x - \Delta C d^2$$

Rovnice 4.4 Výpočet odchylky v MEMS akcelerometru [25]



Obrázek 4.1 Vnitřní struktura MEMS akcelerometru [25]

4.3. GYROSKOPY

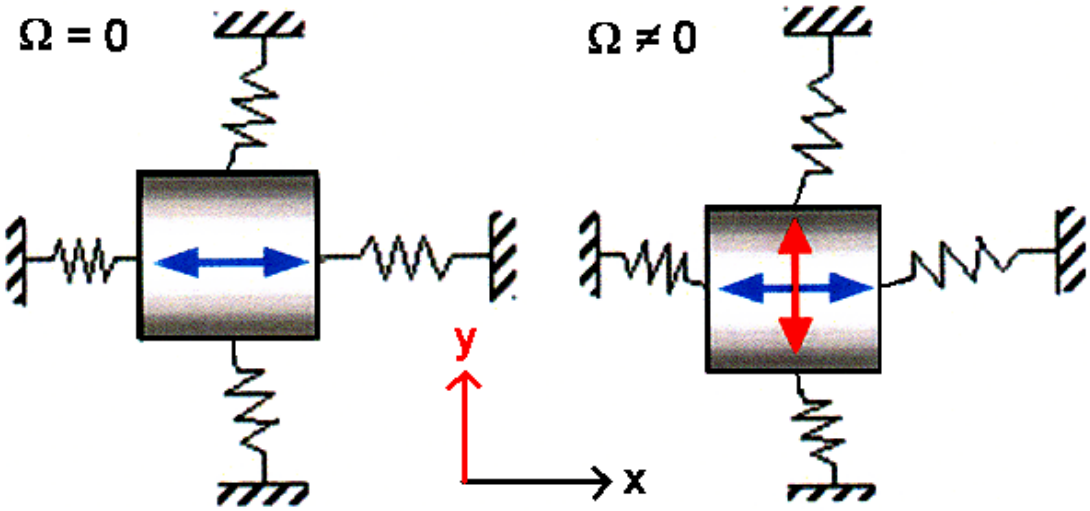
Gyroskopem lze nazývat zařízení, které má tendenci zachovat svou osu rotace za působení momentu setrvačnosti [12]. Takové zařízení se používá na měření rychlosti otáčení a díky tomu nachází využití všude, kde je zapotřebí mít pod kontrolou otáčení jakéhokoli tělesa. V letectví našli uplatnění především kapacitní gyroskopy založené na působení Coriolisovi síly a vyrobené MEMS technologií, která umožnila zmenšit jejich velikost na 1 až 100 μm [13].

Coriolisova síla je přítomná u všech rotačních systémů, a protože velikost této síly je závislá na úhlovém zrychlení, můžeme s její pomocí změřit rychlost natočení daného tělesa. Obrázek 4.2 znázorňuje účinky působení úhlové rychlosti (Ω) na těleso v ose y, které osciluje v ose x.

$$F_{Cor} = 2 . m . \left(\underline{v} \; x \; \underline{\Omega}\right) [N] \tag{4.5}$$

$m \ldots váha\ závaží$
$v \ldots rychlost\ závaží$
$\Omega \ldots úhlová\ rychlost$

Rovnice 4.5 Výpočet Coriolisovi síly dle [14]



Obrázek 4.2 Zobrazení účinků vychýlení tělesa při působení úhlového zrychlení [14]

$$\frac{dx}{dt} = u = V.\sin(2\Omega + \Phi)$$
$$\frac{dy}{dt} = v = V.\cos(2\Omega + \Phi) \quad (4.6)$$

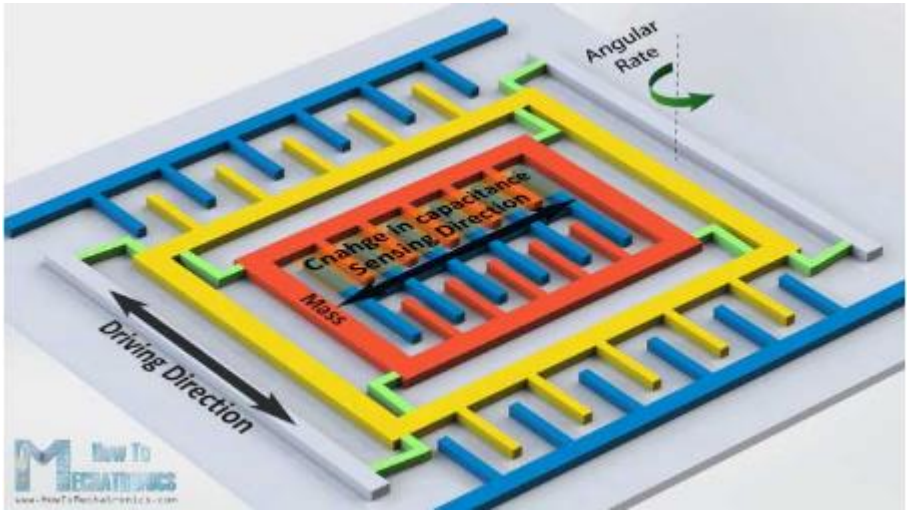
Rovnice 4.6 Výpočet rychlosti pohybu v jednotlivých osách oscilujícího tělesa při působení úhlové rychlosti [15]

V, Φ … *integrační konstanty*

Ω … *úhlová rychlost*

Jak bylo již zmíněno gyroskopy vyráběné technologií MEMS jsou založeny na účincích Coriolisovi síly a jejich vychýlení je měřeno kapacitně. Výhody MEMS gyroskopu dle [26]:

- Malá velikost
- Nízká váha
- Pevná konstrukce
- Nízká spotřeba
- Krátký náběhový čas sensoru
- Nízké náklady na výrobu (ve velkém množství)
- Nízké náklady na údržbu
- Vysoká spolehlivost



Obrázek 4.3 Princip MEMS gyroskopu [13]

Tabulka 4.1 Porovnání gyroskopů GG1320AN a GG5300 od firmy Honeywell [26]

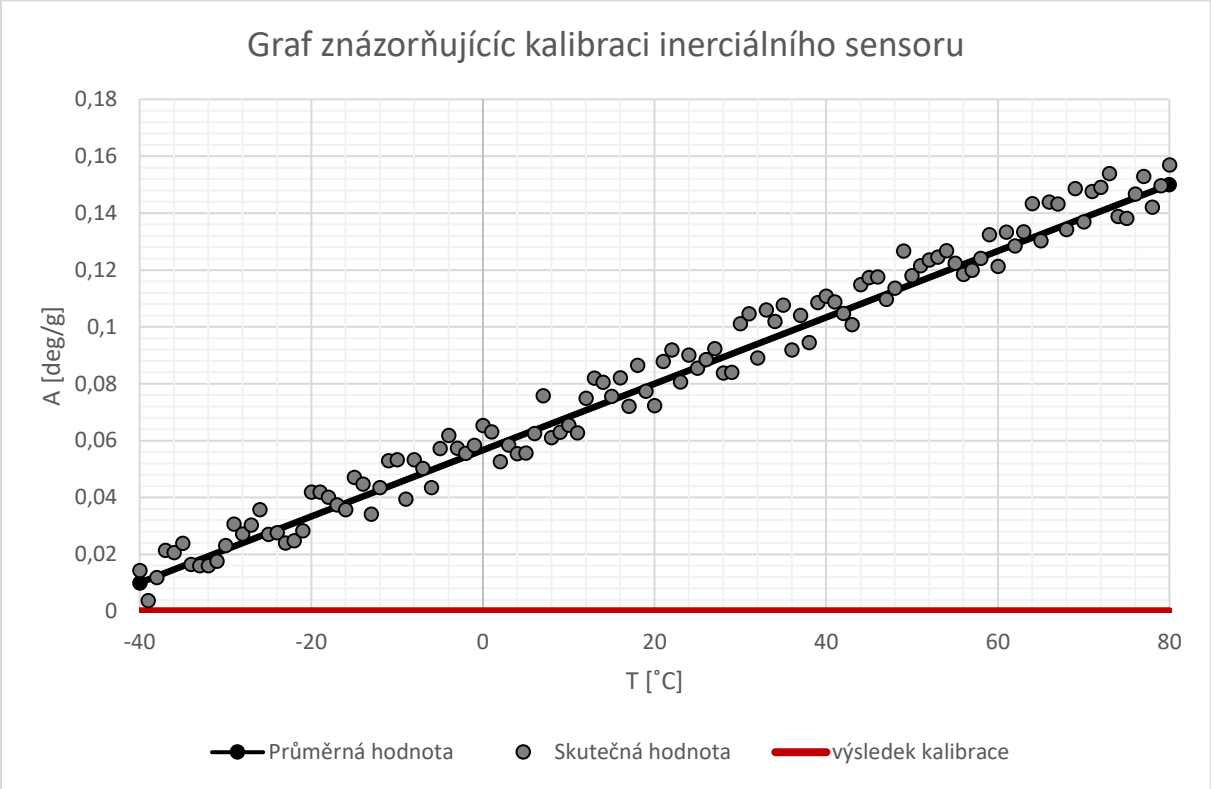
	GG1320AN (Laserový gyroskop)	**GG5300 (MEMS tříosý gyroskop)**
Velikost [mm]	88 x 88 x 45	50 x 50 x 30
Váha [g]	454	136
Čas náběhu [s]	< 4	< 1
Spotřeba	15 Vdc; 1,6 W nominální 5 Vdc; 0,375 W nominální	5 Vdc; < 800mA
Rozsah provozní teploty [℃]	-54 do 85	-45 do 85
Úhlová náhodná chyba [$°/\sqrt{h}$]	0,0035	0,2
Odchylka [$°/h$]	0,0035	< 70

4.4. KALIBRACE

Po zajištění stejného chování sensoru v různých teplotních a tlakových podmínkách je zapotřebí jej zkalibrovat a dostat co nejpřesnější měření z daného sensoru v závislosti na okolních podmínkách. V letectví panují přísné limity na jednotlivé sensory, aby mohli být použity v tomto odvětví. Tento systém se zaměřuje především na testování levných sensorů, které začínají být pro letectví výhodné díky své ceně, ovšem nalezení sensoru, který zároveň splní přísné podmínky může být časově i finančně náročné. ISTP umí provádět definované testy, a to i opakovaně, tím se snižuje cena na kalibraci jednoho sensoru.

Každý sensor měří s jistou chybou, kterou je potřeba zjistit a poté číselně opravit kalibrací. Chyba levných inerciálních sensorů se projevuje nejčastěji šumem v klidovém stavu při stejných teplotních a tlakových podmínkách. Při změně teplotních podmínek dochází k aditivní chybě v sensoru, a právě tuto chybu je třeba změřit a opravit kalibrací.

Graf níže ukazuje příklad kalibrace (nezobrazuje reálná data ze sensoru, slouží jen jako příklad pro doplnění představy o kalibraci inerciálních sensorů)



Tradiční metodou, která se ke kalibraci používá, je definované natáčení sensoru za stabilních teplotních podmínek. Ideálně by měřená gravitační síla a úhel z akcelerometrů a gyroskopů na ISTP odpovídala gravitační síle a úhlu, který je na ISTP aplikován, v průběhu měření v daném časovém okamžiku [11]. Reálně je měření zatíženo chybou, kterou lze vyjádřit:

$$L(\theta) = \sum_{k=0}^{K-1} \{\|u_k\|^2 - \|h(y_k, \theta)\|^2\}^2 \tag{4.7}$$

Rovnice 4.7 Výpočet hledané kalibrační hodnoty pro IMU [11]

θ … *vektor parametrů z akceleromatru a gyroskopu*

u_k … *odhad hledných výstupů založený na sensorovém výstupu*

y_k … *sensorový výstup*

$L(\theta)$ … *hledaná kompenzace pro daný vektor parametrů*

4.5. INERCIÁLNÍ NAVIGACE

Inerciální navigace je technika lokalizování tělesa v prostoru relativně k jeho známým počátečním podmínkám jako je startovací bod, rychlost a natočení. Inerciální navigační systém (INS) je tvořen zpravidla tříosým gyroskopem a tříosým akcelerometrem, které měří úhlovou rychlost a lineární zrychlení tělesa. Gyroskopy a akcelerometry založené na technologii MEMS zajišťují vytvoření levné a malé inerciální

měřicí jednotky (IMU), která obsahuje oba zmíněné sensory a zpravidla ještě teploměr, protože jak gyroskop, tak akcelerometr jsou závislé na okolní teplotě. INS nachází využití v letectví, taktických raketách, kosmonautice, ponorkách a lodích. [26]

5. POPIS TLAKOVÝCH SENSORŮ

Společně s teplotou je tlak jeden z nejdůležitějších fyzikálních parametrů. Konkrétní hodnota tlaku je důležitá v aplikacích napříč obory, např. v oborech termodynamiky, letectví, akustiky, mechaniky kapalin, mechaniky pevných látek a biofyziky. Tato kapitola popisuje fyzikální princip tlaku a fungování tlakových sensorů. [4]

5.1. STATICKÝ TLAK

V této aplikaci je důležitý statický tlak vyjadřovaný v jednotkách psi (Pound per Square Inch), který je definován jako element síly $d\vec{F}$ působící na element plochy $d\vec{A}$.

$$1\ psi = 6{,}890.10^3\ [Pa] = 0{,}06895\ [Bar] \tag{5.1}$$

Rovnice 5.1 Číselné vyjádření jednoho psi v jednotce SI

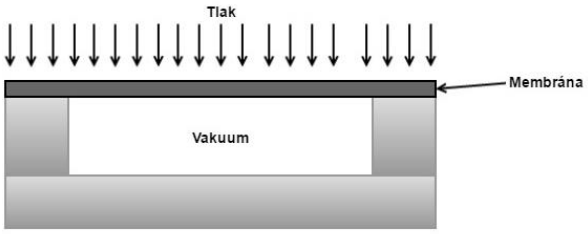
$$p = \frac{d\vec{F}}{d\vec{A}} \left[\frac{kg}{m\,.\,s^2} = Pa\right] \tag{5.2}$$

Rovnice 5.2 tlaku p v závislosti na velikosti díly F působící na plochu A

5.2. TYPY TLAKOVÝCH SENSORŮ

5.2.1. ABSOLUTNÍ TLAKOVÝ SENSOR

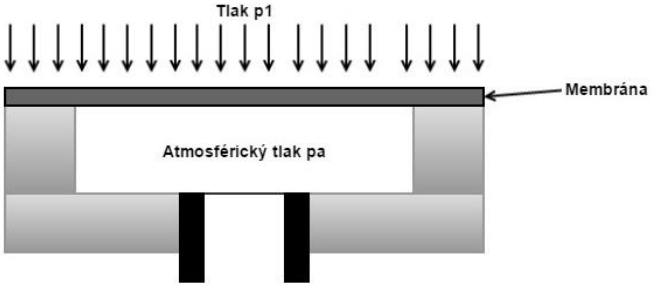
Absolutní tlakový sensor měří statický, dynamický nebo úplný tlak vzhledem k vakuu. [4]



Obrázek 5.1 Princip absolutního snímače tlaku [4]

5.2.2. RELATIVNÍ TLAKOVÝ SENSOR

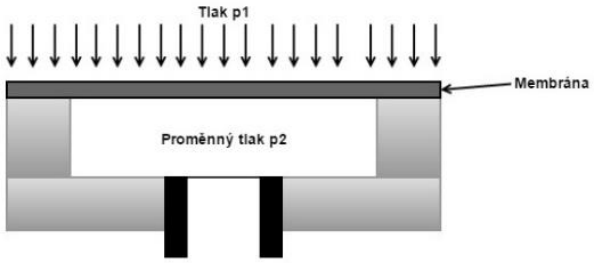
Relativní tlakový sensor měří statický, dynamický nebo úplný tlak vzhledem k okolnímu atmosférickému tlaku. [4]



Obrázek 5.2 Princip relativního snímače tlaku [4]

5.2.3. DIFERENCIÁLNÍ TLAKOVÝ SENSOR

Diferenciální tlakový sensor měří statický, dynamický nebo úplný tlak vzhledem k nespecifikovanému referenčnímu tlaku. [4]



Obrázek 5.3 Princip diferenčního snímače tlaku [4]

5.3. KALIBRACE TLAKOVÝCH SENSORŮ

Kalibrace tlakových sensorů spočívá ve vystavení měřeného sensoru definovanému tlaku při definované teplotě. Definovanou teplotu poskytuje teplotní komora, ve které je sensor umístěn, zatímco o definovaný tlak se stará externí tlakový generátor. Zdroje tlaku jsou přivedeny do tlakového generátoru, který reguluje výstupní tlak dle nastavené hodnoty. Tento generátor je ovládán externě PC aplikací a jeho konkrétní hodnoty tlaku v daném čase jsou určeny uživatelem definovaným profilem.

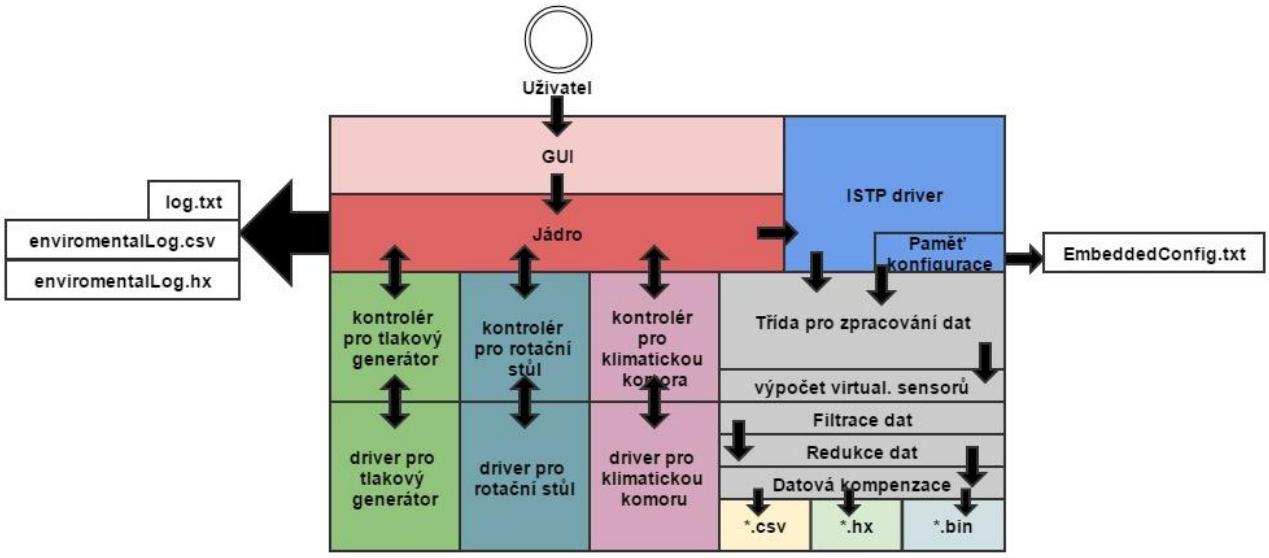
Tlakový generátor je spojen se sensorem pomocí hadiček. Přesnost sensoru je tak určena vzhledem k referenčnímu tlaku kalibrovaného tlakového generátoru.

6. STRUKTURA SW PRO PC

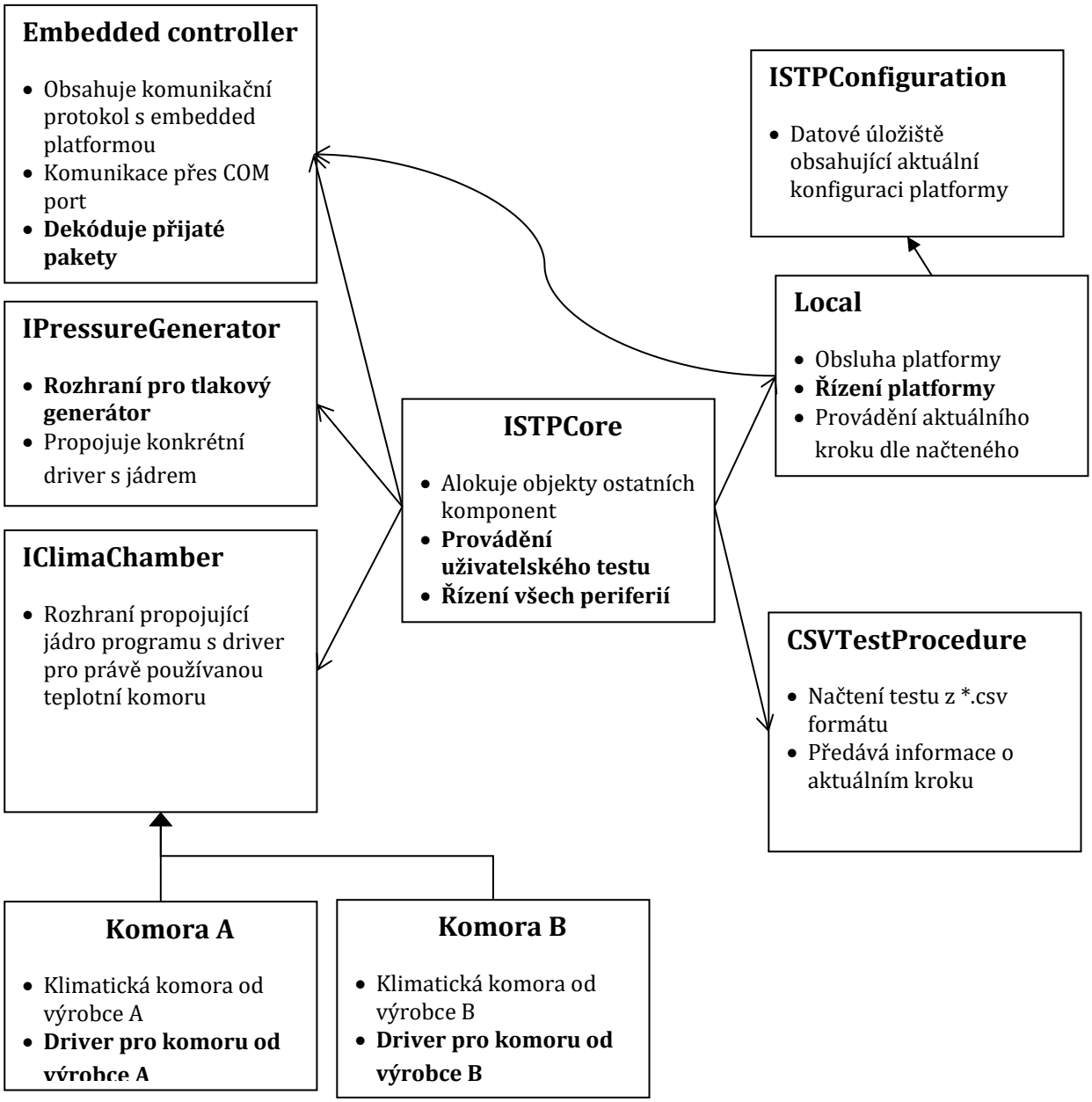
SW pro PC se skládá z několika částí. První částí je samostatné uživatelské rozhraní (GUI), které odděluje obslužnou část od jádra programu. Druhou částí je obsluha ISTP prostřednictvím definovaného protokolu a obsluha ostatních periferií. Třetí částí je ukládání a zpracování dat v reálném čase včetně spouštění externích skriptů.

Jádro programu dostává informace od uživatele prostřednictvím GUI a dle nich řídí veškeré připojené periferie. Důležitou součástí je oddělení uživatelského GUI od jádra programu, tím se předejde kolizi dat, případně kolizi dvou oddělených vláken programu, která by znamenala vyvolání výjimky a možný pád programu. Obsluha periferií probíhá nepřímo přes tzv. rozhraní. Z pohledu jádra se každý typ periferie chová stejně bez ohledu na jejího výrobce a tím i bez ohledu na použitý protokol potřebný pro komunikaci s periferií. Jedinou periferií k jejichž obsluze dochází přímo je ISTP. Další výhodou je možnost snadného přidání nové periferie, kdy stačí nově vytvořený driver zapsat na jediné místo v kódu. Každý typ periferie má vlastní řídicí třídu, která právě skrze rozhraní spojuje konkrétní driver s jádrem programu. Rozhraní tak definuje funkce a jejich návratové proměnné, které daný driver musí obsahovat.

Zpracování a ukládání dat je řízeno skrze jedinou třídu, která volá ostatní třídy nutné pro zpracování a ukládání dat. Tato třída obsahuje hlavní vykonávací funkci, která má jako vstupní parametr komunikační paket. Při startu programu je odkaz na tuto funkci předám čtecímu vláknu, a to onu funkci volá pokaždé, kdy přečte datový paket. Zpracování a ukládání dat probíhá tímto způsobem po jednotlivých paketech.



Obrázek 6.1 Diagram rozložení PC aplikace



Obrázek 6.2 Ukázka struktury PC aplikace

6.1. JÁDRO PROGRAMU

Jádro programu obsahuje dvě vlákna, které jsou aktivní během testu. Jak bylo již zmíněno, tak hlavní vlákno se stará o dekódování a zpracování příchozích paketů z periferií a o jejich obsluhu. Na obsluhu paketů z ISTP stačí jediné vlákno. Paket se dekóduje a předá příslušné funkci, která jej zpracuje dle požadavků uživatele. Funkce pro zpracování dat provede příslušné operace (dle volby uživatele) i zapsání dat do příslušného souboru, protože taková přímá cesta je nejjednodušší na implementaci, ovšem její nevýhodou je, že vyžaduje větší výkon počítače, na kterém je aplikace spuštěna.

Vzhledem k požadavkům a k praktickému nasazení ve firmě Honeywell lze tuto nevýhodu minimalizovat zakoupením PC, které se nachází ve výkonové střední třídě. Pokud se rovněž omezí počet spuštěných jiných aplikací, minimalizuje se zmíněný problém.

Druhé vlákno provádí řízení všech periferií dle uživatelem definovaného profilu. Tohle vlákno je spuštěno jednou za sekundu hlavním časovačem, přičemž si spuštěné vlákno ověří, zdali má provést další krok dle načteného profilu. Z toho vyplývá, že uživatel si definuje testovací profil dle času v sekundách, kdy daný čas určuje, co se má na připojených periferiích nastavit a do jakého stavu se mají uvést, ovšem je nutné zajistit přesné časování dle komunikační rychlosti jednotlivých periferií. Pokud periferie nekomunikuje tak rychle jak je potřeba, nastane opoždění dalšího kroku, protože před samotným vykonáváním profilového kroku se ověří, zdali vykonávací vlákno již běží, pokud ano a zároveň se v aktuálním kroku má provést nová série příkazu, je tento krok o jednu sekundu posunut. Z pohledu jádra programu se každý typ periferie chová stejně, protože jádro nerozlišuje, s jakou konkrétní periferií komunikuje, pouze ví, o jaký typ periferie se jedná, tedy o typ: Klimatická komora, rotační stůl, tlakový generátor nebo ISTP.

Každý typ periferie má vlastní řídící třídu, tzv. kontrolér, která spojuje konkrétní driver pro konkrétní periferii s jádrem programu. Na to, aby mohl kontrolér spojit libovolný driver je zapotřebí definice funkcí, která daný driver musí obsahovat, a právě tuto úlohu zajišťuje rozhraní.

Kromě důležitého řízení, jádro také zaznamenává průběh testu a upozorňuje uživatele na nestandardní stavy. Výstupem řízení jsou dva soubory, první obsahuje průběh testu (*log.txt*), včetně případných chybových hlášení, přičemž druhý soubor obsahuje stavy periferií (pro ISTP volitelné) od spuštění testu (např. u teplotní komory je jejím stavem nastavená a aktuální teplota jakou udává její interní snímač).

6.2. ROZHRANÍ

Bylo již zmíněno, že rozhraní definuje funkce, které musí driver obsahovat, proto jeho hlavní výhodou je zjednodušení celého programu. Jádro volá příslušnou funkci kontroléru pro daný typ periferie, ten si zavolá konkrétní funkci z driveru pro periferii, který si uložil při inicializaci spojení. Důležitá je informace, že při použití rozhraní lze s danou třídou obecně komunikovat jen přes funkce definované právě tímto rozhraním, a proto tyto funkce musí pokrývat celé spektrum možných řídicích operací.

Dané funkce pokrývají komunikaci s periferií a jejím ovládáním. V případě změny periferie programátor napíše driver pro konkrétní zařízení a aplikuje do něj definované funkce, a nakonec přidá nový driver na jediné místo v kontroléru, tím je jeho práce hotova. Při výběru nové periferie se zavolá nový driver a zahájí se komunikace.

6.3. PERIFERIE

Požadované periferie můžeme rozdělit do čtyř kategorií:

1. ISTP (Embedded platforma)
2. Klimatická komora
3. Rotační stůl
4. Tlakový generátor

Každá periferie má vlastní driver, tedy z obecného hlediska se s každou z nich komunikuje jiným způsobem, protože vybraný komunikační protokol záleží pouze na jejím výrobci. Pokud výrobce nabízí možnost použití více protokolů je důležité, aby programátor, který driver píše, vybral ten nejvhodnější v rámci míněného využití zařízení. Vzhledem k nutnosti přesného časování je nejhodnějším protokolem ten, který nabízí nejrychlejší komunikaci. Programátor při tvorbě driveru si musí být vědom, že obsluha dané periferie nesmí trvat více jak 300ms v případě, že jádro musí řídit všechny typy periferií zároveň. V takovém hrubém odhadu je platformě věnováno pouze 100ms, ale vzhledem k použité komunikaci a protokolu je tento čas zcela dostatečný.

6.4. ZPRACOVÁNÍ DAT V REÁLNÉM ČASE

Zpracování dat v reálném čase je volitelnou složkou programu a provádí se ihned po přijetí datového paketu a před tím, než jsou data uložena na disk. Pořadí vykonávání je určeno z praktického hlediska, kdy první se provede výpočet všech virtuálních sensorů, poté se na celý datový objem paketu aplikuje filtr z externího dll souboru, pak je na řadě redukce dat na 80 Hz, a nakonec jejich kompenzace. Veškeré zmíněné typy zpracování jsou volitelné a mohou být použity samostatně.

6.4.1. VIRTUÁLNÍ SENSORY

Jedná se o zpracování v reálném čase, kdy se dle uživatelem definovaného souboru provede průměr z hodnot, které takový soubor definuje. Program vytvoří virtuální sensor (IMU), který má až šest hodnot (tři osy z akcelerometru a tři osy z gyroskopu) a jejichž počet je závislý na počtu daných souborů. V takovém souboru si uživatel volí, které konkrétní hodnoty z konkrétního sensoru budou započteny do jedné z šesti hodnot, přičemž si volí, zdali hodnota bude přičtena nebo odečtena v závislosti na její orientaci. Takto vytvořený sensor se zanese do konfigurační paměti, ve které jsou uloženy informace o všech aktuálně připojených sensorech a tím je zajištěno, že další třídy na zpracování a ukládání dat s ním zacházejí jako s reálně připojeným sensorem.

6.4.2. FILTRACE

Filtrace dat se aplikuje na každý přijatý datový paket. Filtr je realizován externím souborem ve formátu *.dll a tím je zajištěna jeho údržba bez nutnosti zasahovat do kódu PC aplikace. Uživatel si může vybrat frekvenci propustnosti filtru v jednotkách Hertz.

Vstupem do filtru je datové pole právě dekódovaného paketu, které se předá externí funkci ze souboru dll. Celková velikost datového pole, která do této funkce vstupuje obsahuje 2000 prvků ve formátu double precision dle IEEE 754-2008 [5], jestliže datové pole přijatého paketu je menší, doplní se nulami na požadovanou délku. Výsledkem je pole ve stejném formátu a velikosti jako pole vstupní.

Konkrétní filtrace záleží na použitém dll souboru. Pokud uživatel dodrží velikost vstupního i výstupního pole a nezmění název souboru, není od něj potřeba zasahovat do kódu PC aplikace a může tímto způsobem aplikovat libovolný typ filtrace.

6.4.3. REDUKCE FREKVENCE NA 80 HERTZ

Uživatel může definovat až dvě frekvence, které přímo ovlivňují čtení a posílání dat. První je frekvence čtení dat (sample frequency), pokud není aktivován režim first-in-first-out (FIFO) je tato frekvence i frekvencí posílání dat do PC, jestliže uživatel aktivuje režim FIFO představuje tato frekvence jen frekvenci čtení. Při aktivním režimu FIFO se frekvence posílání dat do PC nastavuje zvlášť.

Redukce frekvence probíhá přes dva uživatelsky nastavitelné parametry, tzn. Up scale a down scale. První parametr, up scale, určuje kolikrát se nakopíruje do paměti každý datový záznam. Maximální délka paměti je určena čtecí frekvencí vynásobenou tímto parametrem. Druhý parametr určuje z kolika datových záznamů se bude dělat průměr, jakmile je paměť plná. Níže je uvedený příklad:

Up scale = 8

Down scale = 10

$paket_{1,1}$	$paket_{1,2}$	...	$paket_{1,8}$	$paket_{2,1}$	$paket_{2,2}$	...	$paket_{2,8}$

$$paket_1(x) = \frac{paket_{1,1}(x) + \cdots + paket_{2,2}(x)}{10}$$

$$x \ldots . x - t\acute{y}\ prvek\ daného\ paketu$$

Frekvence 80 Hz je zvolena kvůli speciálnímu režimu (SPS) v jehož průběhu se spouští externí sada skriptů, které požadují data zaznamenána na frekvenci 80 Hz.

6.4.4. DATOVÁ KOMPENZACE

Datová kompenaze se aplikuje stejným způsobem jako filtrace. V uživatelem vytvořeném DLL souboru jsou zakódovány kompenzační polynomy pro jednotlivé sensory. Vstupem do souboru jsou data z daného sensoru a jeho identifikační číslo určené pořadím v DLL souboru. Výstupem jsou kompenzovaná data.

6.5. UKLÁDÁNÍ DAT

Data se ukládají do formátu *.csv, *.hx nebo *.bin. Poslední dva formáty jsou určené pro interní použití ve společnosti Honeywell.

Formát *.hx (Honeyx) je zvolen kvůli hotovému toolboxu pro program Matlab, který s daným formátem umí pracovat. Díky tomuto toolboxu se data jednoduše importují do Matlabu. Toolbox je vlastnictvím společnosti Honeywell s.r.o. a není možné jej přidat k této práci.

Formát *.bin slouží ke speciálnímu módu SPS. Mód SPS má definovanou sktrukturu i podobu datových souborů, jejichž výhodou je spuštění externích skriptů ihned po dokončení měřicího cyklu, a tím se redukuje čas čekání na výsledky testu, protože dané skripty jsou spouštěny ve chvíli, kdy se z platformy nelogují žádná data. Jedná se tedy o úsporu času za maximálního využití daných prostředků.

6.5.1. FORMÁTY CSV A HX

Formát CSV je klasickým formátem, který lze otevřít na libovolném počítači, to je jeho velikou výhodou. Nevýhodou je jeho příliš vysoká velikost při dlouhém kontinuálním ukládání dat, navíc je zapotřebí mnohem větší velikost paměti RAM při otevírání takového souboru.

Formát HX je oproti formátu CSV snadněji otevíratelný, ovšem jeho nevýhodou je nutnost použití programu Matlab s toolboxem od společnosti Honeywell, navíc je třeba tomuto souboru vytvořit korektní hlavičku jinak jej nepůjde otevřít a data budou ztracena. Tento formát je v rámci společnosti Honeywell hojně využíván právě díky výhodám, které nabízí a hlavně proto, že data z testů se dále zpracovávají v programu Matlab.

6.5.2. SPECIÁLNÍ REŽIM UKLÁDÁNÍ DAT – SPS (FORMÁT BIN)

Tento režim byl již krátce představen. Nejedná se jen o způsob ukládání dat, ale i o vytvoření struktury složek a spouštění externích skriptů na již uložená data. Samotný formát je velice jednoduché ukládání dat v následujícím pořadí pro jeden vzorek:

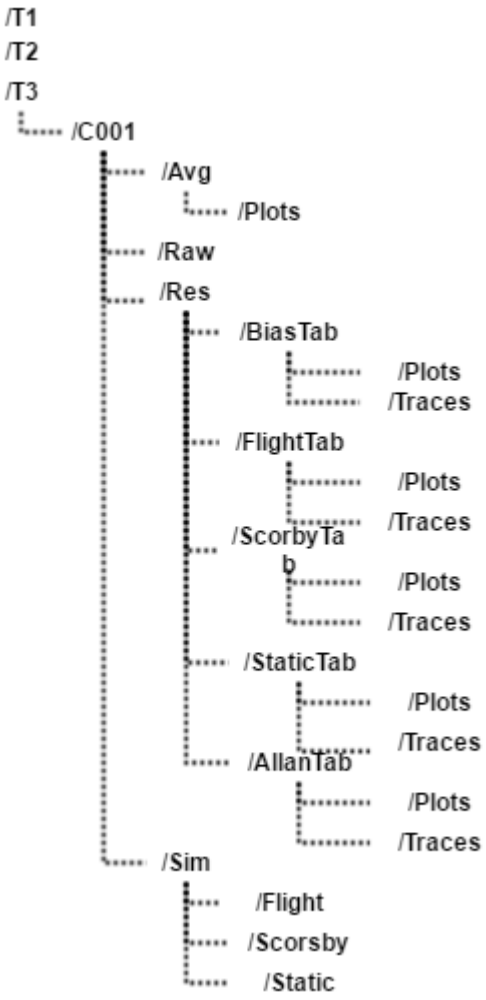
Tabulka 6.1 Zobrazení ukládání do formátu BIN

Čas [s]	Teplota	Acc-X	Acc-Y	Acc-Z	Gyro-X	Gyro-Y	Gyro-Z

Čas je v sekundách a je nutná jeho kontrola, aby začínal časem prvního přijatého vzorku, stejně jako kontrola jeho lineárního narůstání.

Teplota je v jednotkách Celsia a jedná se o teplotu sensoru nebo senzorové desky. Výběr teploty závisí na tom, zdali pochází ostatní hodnoty z jediného sensoru, pokud ne tak se jedná o aritmetický průměr dvou teplot. V případě, že uživatel chce nahradit teplotu daného sensoru teplotou z jiného sensoru – zpravidla se jedná o náhradu interního teploměru s nevyhovujícím rozlišením teploměrem, který je vedle sensoru s vyhovující rozlišení – stačí zadat jejich specifické ID do souboru *SPStempList.csv*.

Další hodnoty pocházejí z akcelerometru od jeho osy X po Z následující třemi hodnotami z gyroskopu od jeho osy X po Z. V případě, že jeden z nich chybí jsou hodnoty nahrazeny nulami.



Obrázek 6.3 Struktura složek v režimu SPS

6.5.2.1. PÁROVÁNÍ SENSORŮ

Z výše uvedeného popisu formátu BIN je zřejmé, že je nutné mít v jediném souboru data jak z akcelerometru, tak z gyroskopu. V případě, že měříme IMU máme všech sedm hodnot, včetně teploty, z jediného sensoru, ovšem pokud měříme zvlášť akcelerometr a gyroskop, kde každý z nich má své vlastní pouzdro a může mít i vlastní senzorovou desku, je nutné hodnoty z takových sensorů vzájemně spárovat.

Párování probíhá v inicializační fázi módu SPS, která začíná po zahájení testu. V takovém případě je již ukončena konfigurace platformy a PC má celou aktuální konfiguraci uloženou ve své interní paměti. Program prvně rozdělí všechny nakonfigurované sensory do kategorie akcelerometrů – ty sensory, které mají svou hodnotu výstupu pojmenovanou jako *accX* (X je číslo osy) – a kategorie gyroskopů – ty sensory, které mají svou hodnotu výstupu pojmenovanou jako *gyroX* (X je číslo osy) –. V rámci tohoto třídění se nekontroluje, zdali se jedná o jeden sensor. Po hrubém třídění do dvou kategorií se vytvoří výsledné páry, a to tím způsobem, že se vezme první uložený akcelerometr a zkontroluje se, zdali je jeho ID obsaženo v kategorii gyroskopů a tím se zjistí, zda se jedná o IMU. V případě že tomu tak není, je spárován s prvním gyroskopem

v dané kategorii. Vyslednému páru se přířadí název Sxx, kde ‘x‘ je číslo páru začínající na 01. Zmíněné informace jsou uloženy do souboru ‘*SensorList.csv*‘.

Z vyššího pohledu senzorové páry závisejí na jejich uložení na osách embedded platformy, protože konfigurace platformy probíhá vždy od osy A od senzorové desky číslo jedna, jsou takto nalezené senzory uloženy do paměti dříve než sensory na jiných osách.

6.5.2.2. STRUKTURA EXTERNÍCH SKRIPTŮ

Jak bylo zmíněno výše, tak aplikace umí spouštět externí skripty a předávat jim potřebné parametry tak, aby nebyla porušena filozofie celé aplikace – potřeba minimálních úprav při změně periferie/skriptů – má uživatel možnost výběru jaké skripty a v jakém pořadí se spustí. Nejjednodušší je aplikace textového souboru, ve kterém jsou uvedeny názvy všech skriptů, které se mají spouštět.

Definujme dva typy skriptů. První typ (I) se spouští ihned po ukončení měřicího cyklu, ovšem druhý typ (II) se spouští až po dokončení určitého množství cyklů. Všechny skripty jsou ve složce „SPS“, která je ve stejné složce jako PC aplikace. Způsob definování skriptu je následující:

I. X.Name.exe.copy,file
   - X – pořadí v jakém se skript spustí. Pokud má více skriptů stejné číslo X, spustí se paralelně.
   - Name.exe – Jméno skriptu.
   - .copy – Tento údaj je nepovinný. Pokud je uveden, znamená překopírování souboru „*file*“ do složky cyklu, při kterém se daný skript spustí.

II. END.Y.X.Name.exe.copy,file
   - END – povinný údaj, který značí druhý typ skriptu.
   - Y – celé číslo udávající za kolik cyklů se daný skript spustí.
   - X – pořadí v jakém se skript bude spouštět. Pokud má více skriptů stejné číslo X, spustí se paralelně.
   - Name.exe – Jméno skriptu.
   - .copy – Tento údaj je nepovinný. Pokud je uveden znamená překopírování souboru „file“ do složky cyklu, při kterém se daný skript spustí.

Danému skriptu se vždy při spouštění předávají parametry. Uživatel má možnost takové parametry definovat, a protože SPS režim má definovanou strukturu je možné tyto parametry zapsat obecně. Parametry se udávají pod konkrétním skriptem, přičemž další skript je oddělen prázdným řádkem. Definujme tři typy parametrů:

1. „\Raw\“ – „Raw“ udává název složky, která se vždy vytvoří v daném měřicím cyklu. Složek, které se vždy vytvoří je více a mají definovaný název.

Lomítko „\“ na začátku udává, že se tato složka nalézá ve složce konkrétního cyklu. Lomítko na konci určuje, že jde o složku.

2. „\“ – Jestliže je pod definicí skriptu uvedeno samotné lomítko je jako parametr předána cesta ke složce, ve které se nalézá poslední měřicí cyklus.
3. „“ – Pokud pod definicí skriptu není uveden žádný znak, je jako parametr předána cesta ke kořenovému adresáři.

6.5.2.3. URČENÍ KONCE PŘECHODNÉHO DĚJE

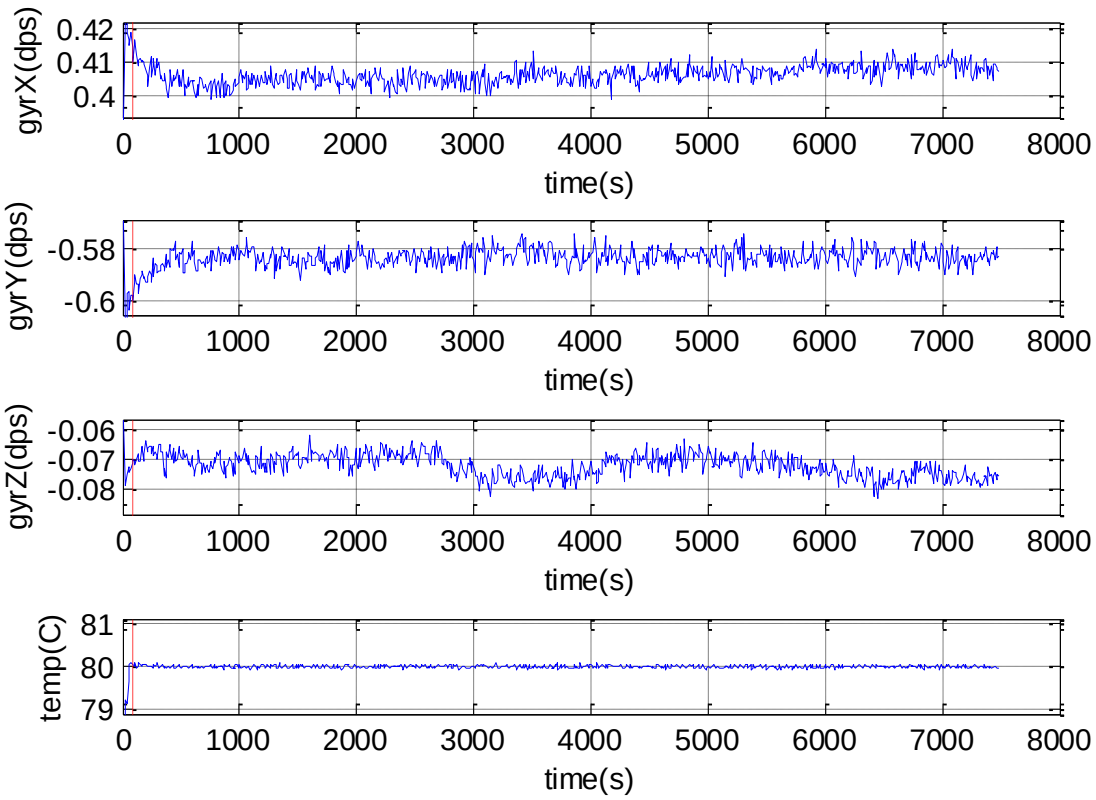
Výše je popsán způsob ukládání dat do formátu BIN i způsob definování a spouštění externích skriptů, ovšem při reálném testování mají senzory přechodné děje, které je potřeba vyloučit, a proto je uživateli umožněno nastavení způsobu, kterým se určí okamžik, od kterého data již nejsou ovlivněna přechodovým dějem.

SPS režim umí pracovat se dvěma způsoby kontroly přechodového děje. První je časový a druhý teplotní.

Časová kontrola přechodného děje probíhá ověřování času příchozích paketů s časem, který nastavil uživatel. Uživatel definuje čas v sekundách a první paket, jehož embedded čas bude větší, než ten definovaný je zapsán do souboru *‚tempStabTimes.bin‘*.

Teplotní kontrola je složitější než časová. Nejedná se jen o kontrolu výstupní teploty ze sensoru, ale také o její stabilizaci. Stabilita teploty se neřídí, jen kontroluje. Uživatel si definuje ‚offset‘ čas, tedy čas, od něhož se začne kontrolovat teplota sensoru. Po uplynutí ‚offset‘ času se musí teplota sensoru nacházet v rozmezí, které definuje uživatel, jakmile tomu tak je, jsou spuštěny interní stopky a teprve po uplynutí definované doby je embedded čas daného vzorku zapsán do souboru *‚tempStabTimes.bin‘*. Pokud je embedded čas větší než maximální možný čas, která taktéž definuje uživatel, je zapsán onen maximální čas a uživatel ví, že daný sensor se nestihl teplotně stabilizovat v časovém rozmezí.

Na obrázku 6.4 je ukázka reálných dat z gyroskopu, který byl vyhříván na konstantních 80 °C. Tento graf byl vygenerován prostřednictvím externího skriptu, který zpracovává naměřená data ihned po skončení jejich záznamu v rámci daného cyklu. Patrná červená přerušovaná čára značí čas, od kterého byl sensor stabilizován na 80 °C. Teplotní stabilizaci má na starosti patřičný driver pro konkrétní senzorovou desku, který není součástí mé DP.



Obrázek 6.4 Záznam dat z gyroskopu vyhřívaného na 80 °C

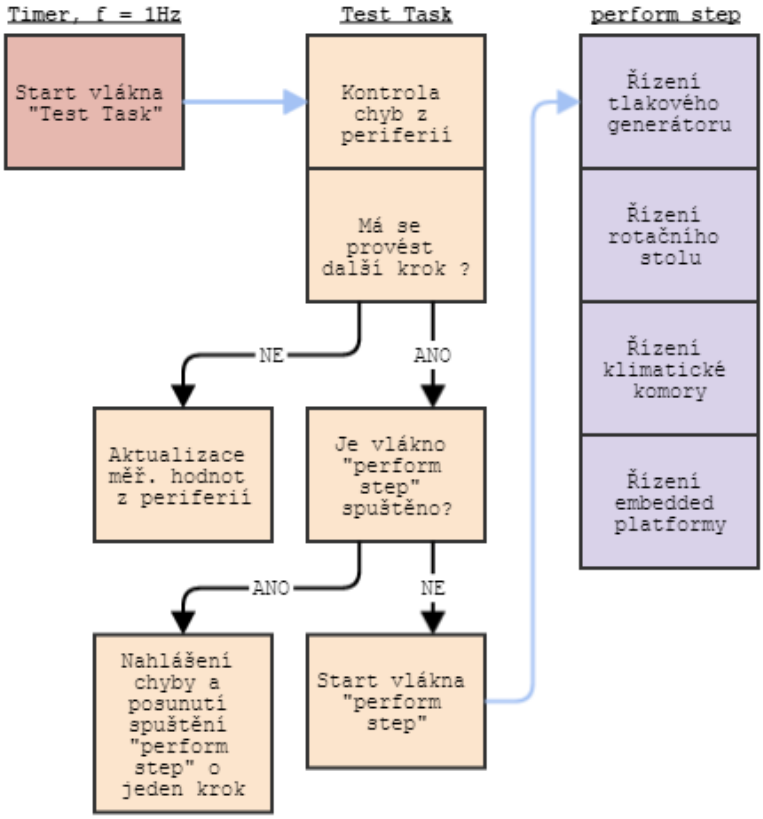
6.6. SPUŠTĚNÍ TESTU

Spustit test lze až ve chvíli, kdy je načten testovací profil, zvolen finální typ datového souboru, případně vybrán způsob zpracování dat, a nakonfigurovány všechny potřebné periferie. Ihned po spuštění testu se do testovacího logu zaznamená veškeré nastavení, které uživatel provedl a spustí se interní časovač s periodou 1 s. Tento časovač každou sekundu spustí vlákno ‚*test task*' a to bez ohledu na to je-li toto vlákno již spuštěno, aby nedocházelo k vzájemnému konfliktu takto spuštěných vláken, je při spuštění *test task* uzamknut statický objekt. To znamená, že každé další vlákno, které chce stejný objekt uzamknout musí počkat na jeho odemknutí. Objekt může být zamknutý vždy pouze jedním vláknem.

Test task se spouští každou sekundu z důvodu minimálního časového kroku, který se může vyskytnout v testovacím profilu. Toto vlákno prvně zkontroluje přítomnost chyb z periferií, případně tyto chyby vyčte a předá uživateli, poté zkontroluje, zdali se stále čeká na potvrzovací zprávu (ACK) z ISTP a pokud ano, oznámí tento stav uživateli a znovu pošle poslední kontrolní zprávu do platformy. Po těchto kontrolách porovná své číslo s krokem z testovacího profilu, jestliže se nemá provést testovací krok vyčtou se hodnoty z periferií a zaznamenají do *enviromental logu,* ovšem pokud se má provést, provede se kontrola, zdali se může spustit vlákno *perform step,* které se nemůže spustit pouze v případě je-li již

spuštěno, v takovém případě je jeho další spuštění posunuto o jednu sekundu a celá situace je zaznamenána do logu.

Perform step je vlákno, které se spouští pouze v časech, které jsou definované testovacím profilem. V rámci tohoto vlákna se provádí řízení všech periferií dle příslušného profilu. Lze vyčíst z předchozího textu, že na řízení všech periferií je pouze jedna sekunda. Časování je v tomto okamžiku kritické, protože jakékoli zdržení znamená potencionální posunutí dalšího kroku a tím může dojít i k znehodnocení celého testu. Veškeré časování je ovlivněno příslušným driver, který musí zajistit, že vlákno *perform step* v něm nestráví více než 300ms, při uvažování maximální obsluhy ISTP na 100ms.



Obrázek 6.5 Diagram provádění jednotlivých kroků testovacího profilu

6.7. UŽIVATELSKÉ PROSTŘEDÍ

Uživatelské prostředí (UI) poskytuje možnosti nastavení připojených sensorů na platformě, nastavení frekvence posílání dat spolu s frekvencí čtení, výběr režimu ukládání dat a typu výsledných datových souborů, načtení uživatelem definovaného testu, výběr reálného zpracování dat a kompletní přehled informací o právě probíhajícím testu.

Vytvoření UI můžeme rozdělit do čtyř fází [19].

6.7.1. FÁZE 1. SBĚR A ANALÝZA UŽIVATELSKÝCH INFORMACÍ

V této fázi se sbírají data o typu uživatelů, kteří výsledný SW budou používat. Hlavními informacemi o nich je jejich znalost dané problematiky, tj. inerciálních sensorů a jejich testování. Dle zadání z firmy Honeywell je nutné, aby daný test mohl spustit

kdokoli, přičemž hlavním kritériem je co nejjednodušší nastavování všech potřebných parametrů. V této fázi se určilo, co bude uživatel nastavovat sám a co nastaví samotný SW automaticky.

6.7.2. FÁZE 2. DESIGN UŽIVATELSKÉHO PROSTŘEDÍ

Design uživatelského prostředí je určen použitím SW. Hlavním úkolem PC SW je řízení a zpracování senzorových dat z embedded platformy, a jelikož výsledný SW bude jen pro interní použití ve firmě Honeywell, tak jeho design je jednoduchý se zaměřením na účelnost.

6.7.3. FÁZE 3. VYTVOŘENÍ UŽIVATELSKÉHO PROSTŘEDÍ

Design se vytvářel průběžně s tím, jak se měnily informace z 1. a 2. vývojové fáze. Vždy při změně či doplnění informací od uživatele se provedla 1. a 2. fáze, které se výsledně aplikovali ve 3. fázi. Tím vznikl prototyp nového designu, který se vyzkoušel na několika testech, dokud nebyl zcela otestován, čímž vývoj designu přešel do 4. fáze.

6.7.4. FÁZE 4. VALIDACE UŽIVATELSKÉHO PROSTŘEDÍ

Při vytvoření nebo změně designu uživatelského prostředí se vždy provedlo jeho otestování, při kterém se současně testovali nové sensory. Tento proces se opakoval, dokud test neproběhl dle požadavků uživatele a zároveň se tímto procesem našla a opravila většina softwarových chyb.

7. STRUKTURA SW PRO EMBEDDED PLATFORMU

7.1. ZPRACOVÁNÍ KONTROLNÍCH PŘÍKAZŮ

Zpracování příchozích kontrolních příkazů probíhá ve stejné smyčce programu jako čtení ze sensorů. Dekódování příchozích paketů má přednost před čtením dat ze sensorů a jakmile se paket správně dekóduje, ihned se provede činnost určená PC aplikací.

Příjem paketů probíhá dvou fázově. V první fázi jsou data zkopírována DMA kanálem, ve druhé fázi mikrokontroler čte příslušné místo v paměti, kde se má nacházet nakopírovaný paket. Jakmile se paket s kontrolním příkazem přijme, je verifikován pomocí CRC a poté dle typu přidělen do patřičné částí programu, která se stará o konkrétní vyřízení kontrolních požadavků. Po hrubém rozdělení pomocí typu je do kontrolního PC odeslána potvrzovací zpráva o úspěšném přijetí paketu a dle jeho konkrétního ID je vykonám patřičný příkaz.

Výpočet kontrolního součtu (CRC) zajišťuje CRCCU (Cyclic Redundancy Check Calculation Unit), dle zvoleného polynomu, kterým je zvolen, na základě interních požadavků, CCITT802.3. Jednotka CRCCU má vlastní DMA kanál a je tvořena HW prvky, čímž je zajištěno rychlejší počítání CRC, než by to zvládla vlastní funkce. V takovém případě je zrychlení několikanásobné. Protože je důležité obsloužit příchozí požadavek, stihnout přečíst data ze všech sensorů a poslat je během jedné periody, je nutné minimalizovat čas potřebný na výpočet kontrolního součtu.

7.2. IDENTIFIKACE PŘIPOJENÝCH SENSORŮ

K identifikaci připojených sensorů je využita EEPROM paměť, která musí být zabudovaná na každou senzorovou desku. MCU se snaží přes SPI číst všechny možné pozice a jakmile dostane správnou odpověď je patrné, že na dané pozici se nachází senzorová deska. Po identifikaci pozice se daná EEPROM ihned vyčte a větší část jejího obsahu je odeslána do řidicího PC. Níže v tabulce 7.1 je ukázka rozdělení paměti EEPROM s příkladem pro senzorovou desku, která obsahuje dva sensory: Akcelerometr a gyroskop. Oranžová barva znázorňuje povinnou část, kterou musí obsahovat každá EEPROM, zatímco žlutá uvádí seznam možných nastavení pro veškeré sensory daného typu obsažené na desce. Třetí částí je komentář, který také není povinný a může být uveden jen pro některé sensory. Poslední částí je vynulování zbytku paměti EEPROM, které se doporučuje z důvodu předcházení potenciálních problémů.

Tabulka 7.1 Příklad obsahu paměti EEPROM pro sensorovou desku s jedním akcelerometrem a jedním gyroskopem

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Obsah	0x23	0x00	0x00	0x00	0x01	0x25	0x31	0x2C
Index	**8**	**9**	**10**	**11**	**12**	**13…23**		**24**
Obsah	0x32	0x25	0x03	0x02	0x25	‚Sensor_A_01‘		0x25
Index	**25…28**	**29**	**30…33**	**34**	**35…38**	**39**	**40…43**	**44**
Obsah	‚temp‘	0x25	‚acc1‘	0x25	‚acc2‘	0x25	‚acc3‘	0x25
Index	**45**	**46…56**		**57**	**58…62**	**63**	**64…68**	**69**
Obsah	0x25	‚Sensor_G_01‘		0x25	‚rate1‘	0x25	‚rate2‘	0x25
Index	**70…74**	**75**	**76**	**77**	**78**	**79…98**		
Obsah	‚rate3‘	0x25	0x25	0x24	0x26	‚Par_list_Sensor_A_01‘		
Index	**99…102**	**103**	**104…107**	**108**	**109**	**110…129**		
Obsah	‚par1‘	0x25	‚par2‘	0x25	0x25	‚Par_list_Sensor_G_01‘		
Index	**130…133**	**134**	**135…138**	**139**	**140**	**141**	**142**	**143**
Obsah	‚par1‘	0x25	‚par2‘	0x25	0x25	0x00	0x01	0x24
Index	**144**	**145**	**146…166**				**167**	**168**
Obsah	0x31	0x25	‚Komentar_Sensor_A_01‘				0x24	0x00

7.2.1. POVINNÁ ČÁST V PAMĚTI EEPROM

V tabulce 7.1 je označena oranžově s indexy od 0 do 77. Úvodním znakem je vždy ‚# (0x23)‘, za ním následuje číslo driveru, který se bude volat pro obsluhu dané senzorové desky. Obecným oddělovačem použitým v rámci celé EEPROM je procento ‚% (0x25)‘. Za číslem driveru je binární kódování sensorů, kde číslo určuje pozici jedničky v 16 bitovém slově. Jednička na dané pozici určuje pořadí daného sensoru. V uvedeném příkladu bude jednička na pozici 1 a 2. Na indexu 10 a 11 je obsažena celková délka bytu dat, která se obdrží z konkrétního driveru pro čtení hodnot ze sensorů, přičemž index 10 znázorňuje desítky a index 11 jednotky, v příkladu je celková délka dat ze senzorové desky 32 B. Další část od indexu 12 po 77 obsahuje unikátní jméno sensoru a názvy veličin, které sensor měří. V rámci celé DP se používají stejné názvy pro stejné veličiny, jak ukazuje tabulka 7.2, zároveň se názvy posílají PC aplikaci, a proto jsou kódovány dle ASCII.

Tabulka 7.2 Vysvětlivky jmen veličin

Zápis	Popis veličiny
‚accH‘	Pro hodnotu z akcelerometru, kde ‚H‘ značí číslo osy sensoru, pro které platí: osa X = 1, osa Y = 2, osa Z = 3
‚rateH‘	Pro hodnotu z gyroskopu, kde ‚H‘ značí číslo osy sensoru, pro které platí: osa X = 1, osa Y = 2, osa Z = 3
‚temp‘	Teplota z interního teploměru obsaženého v sensoru
‚psi‘	Tlak ze sensoru

7.2.2. NEPOVINNÉ ČÁSTI V PAMĚTI EEPROM

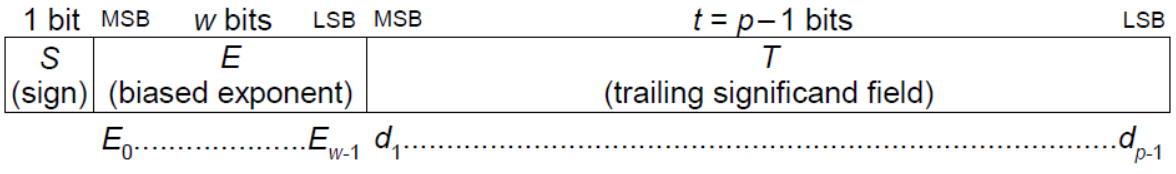
Nepovinné části jsou zvyrazněny žlutou a zelenou barvou v tabulce 7.1. První část (žlutá) obsahuje jednotlivé parametry sensorů, které je možné nastavit. Nastavení je vždy pro celou senzorovou desku pro konkrétní typ sensorů. Detailně se jedná o nastavení, které zajímá obsluhu. Veškerá změna je zapsána na konec této části (indexy 141, 142) a tím je zajištěno stejné nastavení sensorů i po výpadku napájení. Druhá část (zelená) obsahuje komentář, který doplňuje informace určené pro obsluhu. Komentář se vztahuje jen na konkrétní sensor.

7.3. ČTENÍ ZE SENZORŮ

Se sensory se komunikuje přes SPI, ke kterému jsou napsány obecné funkce tak, aby je mohl volat jakýkoli driver pro konkrétní sensor. Čtecí frekvence je nastavena vždy na nejvyšší možnou vzhledem k senzoru se kterým se komunikuje. Tuto frekvenci si udává příslušný driver.

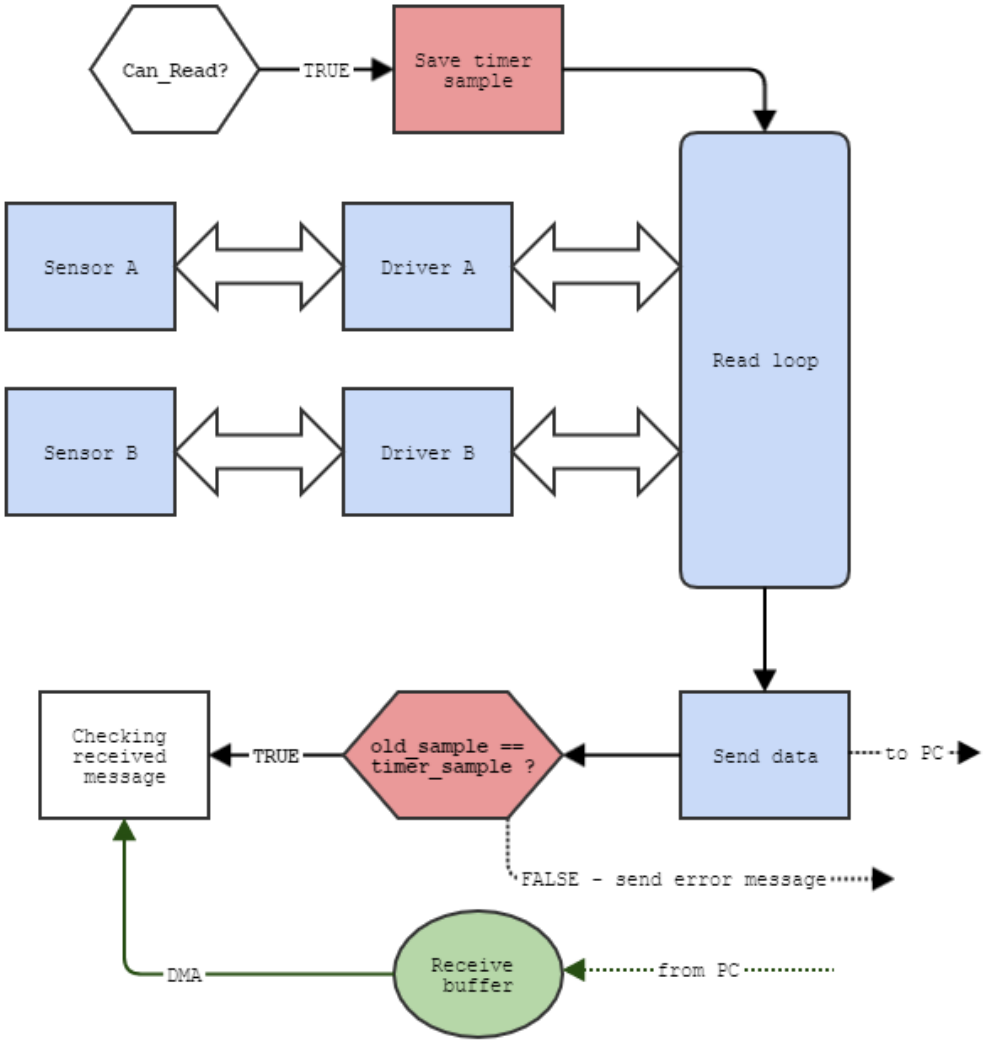
Každý driver má pouze tři funkce na svou obsluhu, přes které se volá. Kolik funkcí celkem a jaké budou implementovány navíc záleží jen na programátorovi, který příslušný driver tvoří. První funkcí je funkce pro inicializaci sensoru, ta provede počáteční nastavení sensoru včetně nastavení parametrů, např. frekvence filtru. Parametr bude reprezentován číslem ve formě jeho pořadí, tedy programátor určí i pořadí možných nastavení. Poté tato funkce pod číslem pořadí najde konkrétní hodnotu parametru, na kterou sensor nastaví. Pořadí odpovídá pořadí uloženého v paměti EEPROM, která se nachází na každé senzorové desce. Z obecného pohledu je konkrétní hodnota poslána do PC aplikace jako pole ASCII znaků. Obsluha, tak ví o konkrétním pořadí a jaký konkrétní parametr se skrývá pod pořadovým číslem. PC aplikace a ISTP spolu komunikují jen prostřednictvím těchto pořadových čísel. Konkrétní hodnotu pořadovému číslu dá až příslušný driver.

Druhou a třetí funkcí jsou funkce na čtení dat ze sensorů. Tyto funkce vrátí pole ve formátu single-precision dle IEEE 754-2008 [5], které je naplněno počtem parametrů ze sensorů. Pole má neměnnou velikost v rámci konkrétní senzorové desky. Pokud by platforma přestala se sensorem komunikovat musí driver doplnit pole nulami, aby se dodržela jeho velikost. Při této fázi je důležité, aby se mikrokontroler zbytečně nezdržoval, kód tak musí být vytvořen efektivně vzhledem k časové obsluze. Ony čtecí funkce se mezi sebou liší jen použitým režimem pro čtení, tedy normální režim a FIFO režim mají každý vlastní čtecí funkci z důvodu, že ne všechny inerciální sensory podporují FIFO režim.



Obrázek 7.1 Bitová reprezentace čísla ve formátu float dle IEEE 754-2008 [5]

Ve čtecí fázi mikrokontroler musí přečíst data ze všech sensorů, sestavit je do jednoho pole, tak jak jdou sensory za sebou, vypočítat CRC a odeslat data. Pořadí sensorů je určeno jejich fyzickým osazením, ale MCU určuje pořadí od své osy A od 1. senzorové desky. Jakmile se nestihne tato operace během jedné periody bude PC aplikaci odeslána chybová zpráva, ale PC automaticky nezastaví probíhající měření.



Obrázek 7.2 Proces čtení dat ze sensorů a jejich následné odeslání do PC

8. KOMUNIKAČNÍ PROTOKOL

Hlavním a nejdůležitějším prvkem celé práce je navržení a realizace komunikačního protokolu mezi ISTP a PC. Zatímco ostatní obsluhované periferie mají implementovaný vlastní protokol od výrobce, používaná embedded platforma je navržena a vyrobena M. Tydorem [1] v rámci jeho diplomové práce pro společnost Honeywell s.r.o.

Jeho použitý protokol nevyhovoval požadavkům na SW, které mi společnost Honeywell zadala, a proto bylo nutné vytvořit protokol vlastní. Nový protokol je dostatečně robustní, ale co nejjednodušší na obsluhu. Požadavek na dobu obsluhy je důležitý z pohledu embedded platformy, která má omezený čas na vyřízení příchozích požadavků a odeslání dat z daného čtecího cyklu.

8.1. STRUKTURA PAKETU

Níže je zobrazena struktura jednoho paketu s detailním popisem jeho elementů.

Tabulka 8.1 Ukázka struktury paketu

Hlavička	Číslo paketu	Časová známka	Typ paketu	Identifikační číslo paketu	Délka datového pole	Datové pole	CRC
4B	4B	8B	4B	4B	4B	(délka datového pole) * 4B	4B

8.1.1. HLAVIČKA

Hlavička reprezentuje začátek paketu, a proto byla zvolena jako 4 znaky typu ‚#' (0x23).

8.1.2. ČÍSLO PAKETU

Číslo paketu je důležité při fázi posílání dat ze sensorů. Při této fázi číslo začíná nulou a inkrementuje se o jedničku každým poslaným paketem. Tím je zajištěna identifikace ztracených paketů.

Číslo paketu je vyjádřeno v neznaménkovém typu integer, jehož maximální hodnota je $2^{32} - 1$. Pro přetečení je nutné, aby platforma byla v logovací fázi, protože číslo paketu je vždy vynulováno na začátku této fáze. V úvaze budeme pokračovat a určíme, že do výsledného souboru se bude ukládat jediné číslo typu float, které má velikost 4B. Při vynásobení velikosti jediného čísla typu float s maximálním možných číslem paketu nám vyjde, že výsledný soubor by měl velikost téměř 16GB. Běžně z inerciálních senzorů čteme minimálně čtyři hodnoty (jednu hodnotu z každé osy a teplotu) a z tlakových hodnoty dvě (tlak a teplotu), z toho nám plyne, že výsledný soubor by byl mnohonásobně větší. Výsledkem je, že není potřebné se zabývat přetečením čísla paketu, protože

zaznamenávání dat při reálných testech netrvá tak dlouho, abychom se k maximálnímu číslu alespoň přiblížili.

8.1.3. ČASOVÁ ZNÁMKA

Časová známka je, stejně jako číslo paketu, nejdůležitější při posílání dat ze sensorů. Časovou známku tvoří embedded platforma v jednotce sekunda. Jakmile přijde do embedded platformy příkaz o startu logovací fáze, časová známka se vynuluje a dle nastaveného interního časovače v embedded platformě se inkrementuje.

Časová známka je typu double a je vyjádřená v sekundách. Výpočet časové známky vždy záleží na pořadované frekvenci čtení dat ze sensorů.

$$t = \frac{timestamp \, . \, \frac{\frac{f_{MCU}}{TC0_{div}}}{f_{sample}} + TC0_{counter}}{\frac{f_{MCU}}{TC0_{div}}} \, [s] \tag{8.1}$$

$$timestamp \, [-] \ldots interní \, proměná \, typu \, 64bit \, integer$$

$$f_{MCU}[Hz] \ldots frekvence \, MCU$$

$$TC0_{div}[-] \ldots dělitel \, interního \, časovače$$

$$f_{sample}[Hz] \ldots požadovaná \, frekvence \, pro \, čtení \, dat$$

$$TC0_{counter}[s] \ldots hodnota \, interního \, časovače$$

Rovnice 8.1 Výpočet časové známky

Interní časovač je inicializován požadovanou frekvencí pro čtení dat ze sensorů a s touto frekvencí volá přerušení, ve kterém se inkrementuje hodnota proměné ‚timestamp' o jedničku. V rámci tohoto přerušení je povoleno čtení ze sensorů a změněn stav indikační LED. Proměná ‚f_{MCU}' je rovna frekvenci MCU, tedy 120MHz. Velikost proměnné ‚$TC0_{div}$' je závisla na hodnotě f_{sample} tím způsobem, že hodnota f_{sample} musí být větší než $TC0_{min}$, jak ukazuje následující tabulka:

Tabulka 8.2 Velikost $TC0_{div}$ v závislosti na f_{sample} [2]

$\boldsymbol{TC0_{div}[-]}$	$\boldsymbol{TC0_{min}[Hz]}$
2	915,5
8	228,9
32	57,2
128	14,3

Výpočet $TC0_{min}$ je dán následujícím vztahem, který je součástí inicializační funkce interního časovače:

$$TC0_{min} = \frac{f_{MCU}}{TC0_{div}} \cdot \frac{1}{65536}\ [Hz] \tag{8.2}$$

Rovnice 8.2 Výpočet $TC0_{min}$ [2]

Jako příklad uvádím výpočet časové známky pro $f_{sample} = 100Hz$.

$$t = \frac{timestamp \cdot \frac{\frac{120 \cdot 10^6}{32}}{100} + TC0_{counter}}{\frac{120 \cdot 10^6}{32}} = timestamp \cdot 0{,}01 + \frac{TC0_{counter}}{3{,}75 \cdot 10^6}\ [s] \tag{8.3}$$

Rovnice 8.3 Příklad výpočtu časové známky pro čtecí frekvenci 100 Hz

8.1.4. TYP PAKETU

Typ paketu určuje jeho význam. Dle toho se pozná, jaké data paket obsahuje. Jedná se o hrubé třídění zpráv.

Tabulka 8.3 Význam číselných hodnot v poli Type

Číselná hodnota	Název	Význam
0x01	Force stop	Závažná chyba vedoucí k ukončení veškeré činnosti
0x02	Error	Chyba, jejíž závažnost posoudí uživatel
0x04	Configure	Konfigurační paket
0x05	Version	Verze SW nahraného v ISTP
0x08	Initialization	Inicializační paket
0x10	Data	Datový paket
0x11	Health data	Status embedded platformy
0x20	Control	Řídicí paket
0x40	Comment	Paket obsahující komentář
0xFF	Acknowledge	Potvrzovací paket

8.1.5. IDENTIFIKAČNÍ ČÍSLO

Identifikační číslo (ID) paketu udává jeho bližší význam v souladu s jeho typem. Jedná se o jemné třídění zpráv. Pro různé typy může paket nabývat různých významů.

A. Force stop

- ID zde postrádá význam. Paket typu Force stop vede vždy k zastavení probíhajícího testu.

B. Error

- 0x01 – číslo paketu obsahuje počet ztracených paketů
- 0x02 – požadovaná čtecí frekvence nemůže být dosažena. Embedded platforma nestihla v jedné periodě provést všechny požadované operace.
- 0x03 – Chyba CRC
- 0x04 – Neznámá zpráva

C. Configure

Konfigurační paket se posílá při započetí komunikace mezi platformou a PC. Jeho význam je různý v závislosti na straně, která jej posílá. Pokud je zpráva z PC jedná se o požadavek k nakonfigurování nebo o změnu konfigurace. Pokud jde zpráva z ISTP obsahuje kompletní informace o nalezené senzorové desce včetně sensorů, které senzorová deska obsahuje.

- ***Z embedded platofrmy do PC***
    - V ID je zakódováno umístění senzorové desky a počet sensorů, které obsahuje.
    - Vrchních 8 bitů specifikuje orientaci na platformě.
    - Dalších 8 bitů reprezentuje číslo senzorové desky. První deska má číslo nula.
    - Zbylých 16 bitů udává počet sensorů na senzorové desce. Počet je zakódován bitovou jedničkou na příslušné pozici. První sensor bude mít bitovou jedničku na nejnižším bitu, druhý sensor na druhé pozici atd.

Tabulka 8.4 Kódování senzorů v ID

Pořadí sensoru	Číslo reprezentující sensor
1	0x0001
2	0x0002
3	0x0004
4	0x0008
5	0x0010
6	0x0020
7	0x0040
8	0x0080
9	0x0100
10	0x0200
11	0x0400
12	0x0800
13	0x1000
14	0x2000
15	0x4000

- ***Z PC do embedded platformy***
    - **0x01** – Požadavek o konfiguraci.
    - **0x02** – Požadavek na konfiguraci pouze jedné senzorové desky
    - **0x04** – Změna konfigurace senzorové desky
      Detail ID:
      0x OB TT PP 04
      O - Orientace (kódovaná binárně)

B - Číslo desky (kódované binárně)
TT - Číslo parametru, který se má změnit
PP - Číslo pořadí konkrétní hodnoty, která se nastaví.

- **0x07** – Zapnutí Health dat
  Detail ID:
  0x RR RR EE 07
  R - Rezervováno
  EE - Zapnutí = 0x01, Vypnutí = 0x00
- **0x08** – Nastavení čtecí frekvence v Hz. Horních 16 bitů reprezentuje požadovanou čtecí frekvenci
- **0x10** – Počet simulačních sensorů (jeden simulacni sensor obsahuje akcelerometr, gyroskop a teploměr)
- **0x20** – De/aktivace FIFO režimu. Horních 16 bitů reprezentuje zapnutí (1) nebo vypnutí FIFO režimu (jiná hodnota než 1, typicky 0).

D. Version

Verze implementovaného SW v embedded platformě. Důležitá informace pro obsluhu, která tak může zkontrolovat správnost verze.

E. Initialization

Požadavek na inicializaci embedded zařízení. Inicializace probíhá zapnutím napájecích linek. Pokud jsou ve stavu Low a mají přejít do stavu High, tak se po zapnutí linek provede konfigurace sensorů.
Detail ID:

- 3V3 = 0x00000001
- 5VD = 0x00000002
- 5VA = 0x00000004
- EXT = 0x00000008

F. Data

Zde ID nemá definovaný význam.

G. Health data

Obsahuje status informace z platformy, které zahrnují hodnoty AD převodníků kontrolující napětí jednotlivých větví, teplotní a vlhkostní sensory umístěné přímo na hlavní desce.

H. Control

Podobně jako konfigurační paket je i význam kontrolního paketu závislý na tom, kdo jej posílá.

- ***Z PC do embedded platformy***
  - 0x00000001 = Start logovací fáze.
  - 0x00000002 = Konec logovací fáze.
- ***Z embedded platformy do PC***
  - 0x00000004 = Konec konfigurace embedded zařízení.

I. **Comment**
ID je stejné jako konfigurační paket z embedded do PC. Data obsahují komentář.

J. **Acknowledge**
ID není pro tento typ paketu definované.

8.1.6. DÉLKA DATOVÉHO POLE

Specifikuje délku datového pole v jednotkách byte ve zprávě.

8.1.7. DATOVÉ POLE

Datové pole paketu obsahující čísla ve formátu float dle IEEE 754-1985. Pokud se jedná o datový paket, tak tohle pole obsahuje data z jednotlivých sensorů. Pokud se jedná o ACK je v tomto poli uchován Typ a ID zprávy, na kterou se ID vztahuje.

- ***Typ – Configure***
  Z ID PC určí pozici senzorové desky a počet sensorů na ní. Datové pole obsahuje veškeré informace z EEPROM, která je na senzorové desce. Informace jsou pro všechny sensory, které deska obsahuje.

Tabulka 8.5 Struktura dat v konfiguračním paketu

Oddělovač ´%´ ASCII	**První aktivní sensor**	**Oddělovač ´%%´ ASCII**	**Druhý aktivní sensor**	**Oddělovač ´%%´ ASCII**
1B	Dle délky dat	2B	Dle délky dat	2B

Oddělovač je reprezentován znakem procento v ASCII formátu. Jeden oddělovací znak je použitý na začátku a dva oddělují příslušné informace pro jednotlivé sensory.

Informace o sensoru jsou kódovány následovně:

Tabulka 8.6 Kódování dat pro jeden senzor

Jméno sensoru	**Oddělovač ´%´ ASCII**	**Název první proměnné**	**Oddělovač ´%´ ASCII**	**Název druhé proměnné**	**Oddělovač ´%´ ASCII**	**…**
Dle délky dat	1B	Dle délky dat	1B	Dle délky dat	1B	…

- ***Typ – Comment***
  V tomto typu paketu se posílají komentáře k senzorové desce nebo seznam nastavitelných parametrů.
  - Spodní dva byty nejsou nulové:

Poté datové pole obsahuje komentář, který je zakódován jako pole ASCII znaků

- Spodní dva byty jsou nulové:

Datové pole obsahuje názvy parametrů pro sensory na desce. Nastavení je stejné pro všechny sensory na celé senzorové desce.

Detail:

Tabulka 8.7 Struktura dat v paketu typu komentář

'&' ASCII	Jméno parametr listu	'%' ASCII	Číslo 1. parametru ASCII	Mezera ' ' ASCII	Název 1. parametru	'%' ASCII	Číslo 2. parametru ASCII	Mezera ' ' ASCII	Název 2. parametru	'%%' ASCII	'$' ASCII	Zvolený parametr	'$' ASCII
1B	Dle délky dat	1B	1B	1B	Dle délky dat	1B	1B	1B	Dle délky dat	2B	1B	Dle délky dat	1B

- ***Typ – Data***
  Datový paket obsahuje zaznamenaná data ze sensorů. Data mezi sebou nemají žádný oddělovací znak. Jejich pořadí je určeno pořadím sensorů, tak jak přišli za sebou v konfigurační fázi. PC nemůže ověřit pořadí (nepozná přehození dat), proto je tato režie prováděna v embedded části a ta musí zařídit nezměněné pořadí po celou dobu testování. Data ze sensorů jsou ve formátu float a maximální délka datového pole je omezena na 2048 B.
- ***Typ – Health data***
  Datové pole obsahuje float hodnoty z AD převodníků a případných snímačů, které jsou součástí základní desky v ISTP (např. snímač vlhkosti). Tato data jsou do PC posílána s frekvencí 1 Hz.
  Detail:

Tabulka 8.8 Struktura dat v pakety typu Health Data

+3V3	+5V	+5V_SW	+3V3_B	+3V3_A	+3V3_C	EXT_PWR	EXT_PWR2
4B	4B	4B	4B	4B	4B	4B	4B

- ***Typ – Version***
  Datové pole obsahuje ASCII znaky reprezentující název (číslo) implementované SW verze v embedded platformě.
  Detail:
  XX.XX.XX.XX

8.1.8. KOTROLNÍ SOUČET

Kontrolní součet je realizován ve 32 bitovém formátu s použitým polynomem CCITT802.3. Tento polynom má HW podporu na straně použitého procesoru Cortex-M4 z rodiny SAM4S. Díky HW podpoře je počítání CRC na straně embedded platformy výrazně rychlejší.

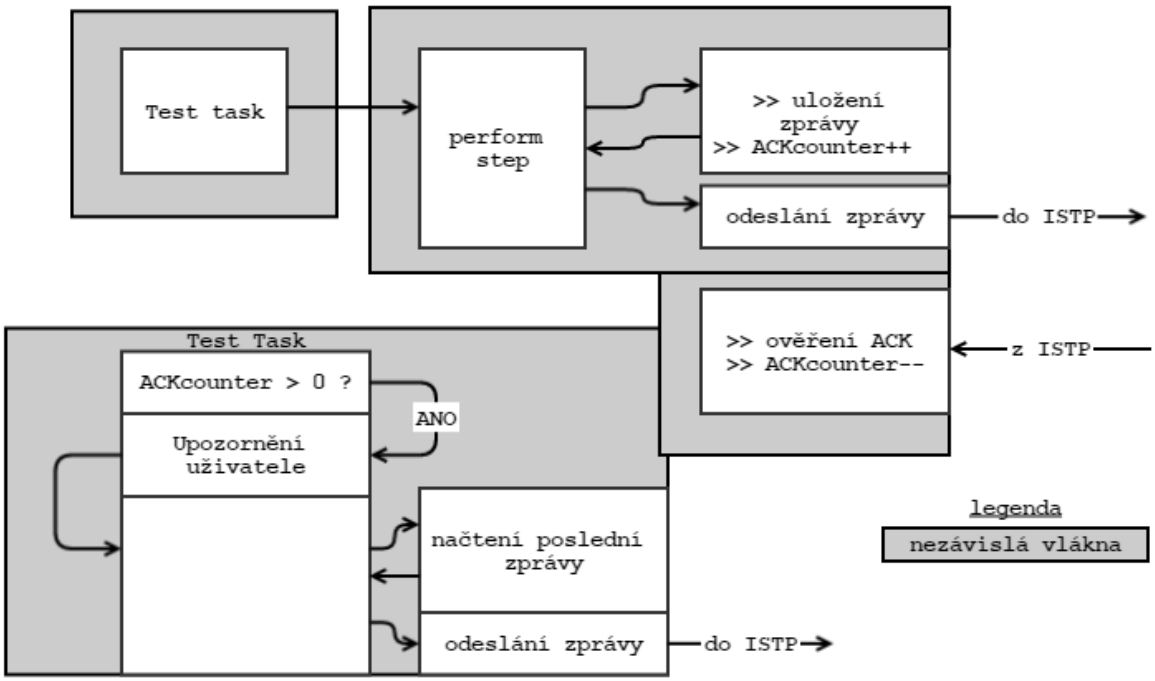
8.2. ACKNOWLEDGE

Potvrzovací zpráva neboli Acknowledge (ACK) je důležitým bezpečnostní prvkem, protože při reálné komunikaci může platforma přestat komunikovat z libovolných důvodů, případně může dojít k poruše komunikačního kabelu. Oba tyto stavy jsou nežádoucí a je potřeba na ně reagovat, a hlavně informovat obsluhu, že takový stav nastal.

V použitém protokolu se ACK vyžaduje od platformy na jakoukoli zprávu, která jde směrem z PC. Při opačných zprávách, z platformy do PC, se ACK nevyžaduje.

Na obrázku níže je zobrazena obsluha ACK zpráv na straně PC. Ve chvíli, kdy se, dle testového předpisu, má provést řídicí krok, se vláknem vytvoří a spustí nové vlákno *perform step.* V rámci onoho vlákna se řídicí zpráva vytvoří, uloží do paměti, zvýší se hodnota interní proměnné *ACKcounter* o jedna a poté se zpráva pošle do platformy. Toto vlákno nekontroluje a nečeká na příchozí ACK, to má na starosti mateřské vlákno *Test task.*

Vlákno *Test task* kontroluje hodnotu interní proměnné *ACKcounter* při každém svém spuštění. Jakmile je hodnota této proměnné větší, něž nula znamená to, že v krátké minulosti se do platformy poslala řídicí zpráva, a ještě nedošlo ACK. *Test task* je spouštěn interním časovačem každou sekundu, a proto mezi odesláním řídící zprávy a kontrolou přijetí ACK je téměř jedna sekunda. *Test task* načte poslední řídicí zprávu a znovu ji odešle do platformy. O tomto stavu informuje uživatele.



Obrázek 8.1 Obsluha ACK v PC SW

8.3. KONTROLA PŘÍCHOZÍCH ZPRÁV

Kontrola příchozích zpráv je velmi důležitou součástí každého protokolu. Vytvořený protokol obsahuje hned tři prvky, které umožňují kontrolu správnosti příchozích dat. Prvním prvkem je unikátní hlavička, následována časovou značkou a posledním prvkem je kontrolní součet přes celý obsah zprávy.

8.3.1. KONTROLA POMOCÍ UNIKÁTNÍ HLAVIČKY

Unikátní hlavička je základem každé zprávy v mém protokolu a zajišťuje správné určení začátku paketu. V rámci přenosu se data mohou poškodit a v přijímači špatně dekódovat, a proto je hledání unikátní bitové posloupnosti správným řešením, jak zvýšit robustnost komunikačního protokolu. Jakmile je v bitové toku nalezena unikátní bitová posloupnost je započat proces dekódování jednotlivých částí paketu.

8.3.2. KONTROLA POMOCÍ CRC32-802.3

Redundantní cyklický kontrolní součet (CRC) je způsob ověření přijaté zprávy, že její obsah je nepoškozen. Tento způsob ověření je hojně používán napříč technickou praxí. Běžně používané CRC standardy jsou CRC-8, CRC-12, CRC-16, CRC-32 a CRC-CCIT. Vytvořený komunikační protokol využívá CRC-32-IEEE802.3. [21]

Použitý kontrolní součet se primárně využívá hlavně v aplikacích komunikující přes Ethernet a ve formátech MPEG2 [22]. Důvodem použití pro nově vzniklý protokol byla nejen jeho hardwarová podpora přímo v MCU Cortex-M4, ale také jeho ověřenost v technické praxi.

$$y = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (8.4)$$

Rovnice 8.4 Polynom CRC-32-IEEE802.3 [22]

Tabulka 8.9 Typ možných chyb v komunikaci a jejich řešení v rámci protokolu pro ISTP

Typ chyby	CRC-32-IEEE802.3	ACK	Časová známka
Náhodný chybný bit	✔		
Chybná série bitů	✔		
Ztráta paketu		✔	✔

8.3.3. KONTROLA POMOCÍ ČASOVÉ ZNÁMKY

Časová známka je velmi důležitou součástí datové zprávy, pro ostatní typy paketů je její význam zanedbatelný, a proto se se u ostatních typů ani nekontroluje. V rámci logovací fáze čte ISTP data z aktuálně připojených senzorů a před jejich odesláním zaznamená časovou známku z interního časovače. Ten se resetuje vždy při začátku logovací fáze. Jakmile je datový paket doručen PC aplikaci a je úspěšně dekódován – úspěšně projde kontrolou CRC – je časová známka zkontrolována a paket je předán k dalšímu zpracování, které pracuje právě s časem uloženým v paketu, protože jedině tento čas sděluje, v jakém okamžiku byla data zaznamenána. Kontrola je pouze porovnání velikosti poslední časové známky se známkou právě doručenou a jakmile má nově příchozí paket větší hodnotu časové známky než jedna perioda (dle čtecí frekvence) je zřejmé, že se ztratilo několik paketů a tento stav je oznámen uživateli. Po případném provedení zpracování dat v reálném čase je daný čas zaznamenán do výsledného souboru.

9. STRUKTURA TESTOVACÍHO SKRIPTU

Testovací skript je ve formátu *.csv. PC aplikace řídí jednotlivé periferie dle tohoto testu. Ten si definuje uživatel a je zodpovědný za časování mezi jednotlivými fázemi testu. Aplikace pozná periferie potřebné k testu díky hlavičce v testu.

První řádek je věnován názvům jednotlivých sloupců. Tyto názvy musí být srovnatelné s definicí. Pořadí sloupců může být přehozené (nemusí být dva sloupce, které spolu souvisí vedle sebe). Definice názvu sloupců a jejich význam:

Tabulka 9.1 Povolené hlavičky testovacího skriptu

Název sloupce v *.csv	Význam	Periferie
Time (s)	Časový sloupec udává, co se v daném čase má stát	Povinné pro všechny periferie
Climatic chamber state (on/off)	Zapnutí (1) nebo vypnutí (0) klimatické komory	Klimatická komora, rotační stůl
Temperature (deg C)	Teplota v komoře	Klimatická komora, rotační stůl
Temperature rate (deg C/min)	Teplotní gradient v komoře	Klimatická komora, rotační stůl
Humidity (%)	Vlhkost v komoře	Klimatická komora, rotační stůl
Humidity rate (%/min)	Vlhkostní gradient v komoře	Klimatická komora, rotační stůl
Pressure state (on/off)	Zapnutí (1) nebo vypnutí (0) tlakového generátoru	Tlakový generátor
Static pressure (mbar)	Statický tlak	Tlakový generátor
Pitot pressure (mbar)	Pitot tlak	Tlakový generátor
AxisInner state (on/off)	Zapnutí (1) nebo vypnutí (0) vnitřní osy rotačního stolu	Rotační stůl
Rotate axisInner (on/off)	Zapnutí (1) nebo vypnutí (0) otáčení vnitřní osy	Rotační stůl
Position axisInner (deg)	Nastavení pozice vnitřní osy	Rotační stůl
Velocity axisInner (deg/s)	Nastavení rychlosti otáčení vnitřní osy	Rotační stůl
AxisOuter state (on/off)	Zapnutí (1) nebo vypnutí (0) vnější osy rotačního stolu	Rotační stůl
Rotate axisOuter (on/off)	Zapnutí (1) nebo vypnutí (0) otáčení vnější osy	Rotační stůl

Position axisOuter (deg)	Nastavení pozice vnější osy	Rotační stůl
Velocity axisOuter (deg/s)	Nastavení rychlosti otáčení vnější osy	Rotační stůl
Sensor power (on/off)	Zapnutí (1) nebo vypnutí (0) napájecích linek na platformě	Embedded platforma
Logging state (on/off)	Zapnutí (1) nebo vypnutí (0) logovací fáze	Embedded platforma

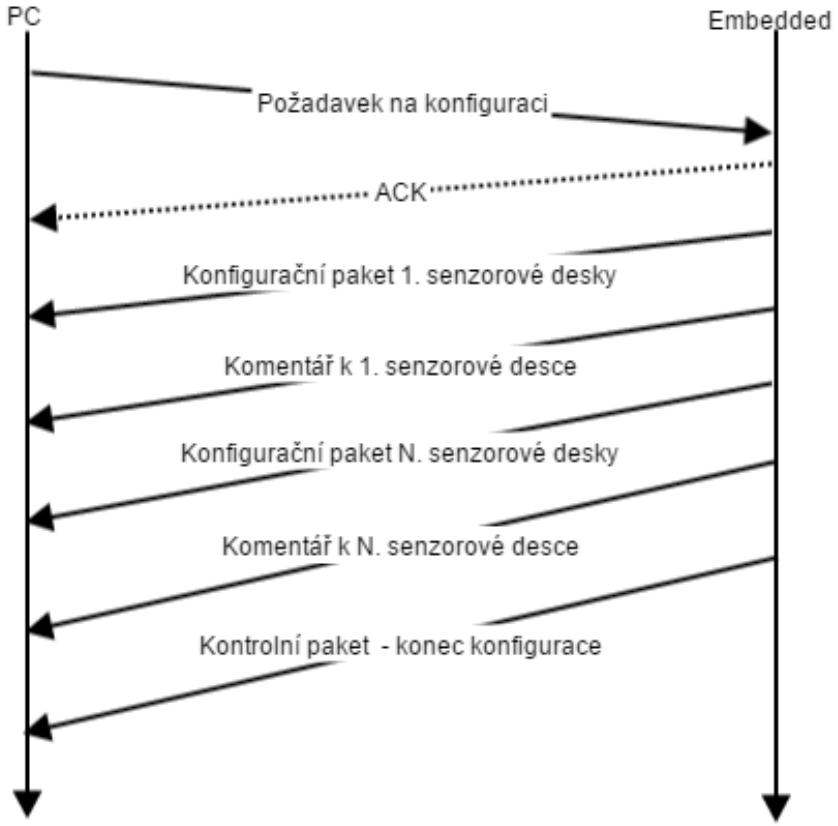
Pro každou periferii musí být celý balík sloupců v daném testovacím skriptu, jinak program nahlásí chybu – neznámý testovací skript – uživateli. Nejdůležitější je časový sloupec '*Time (s)*'. Ten začíná vždy nulou a poté definuje čas v sekundách, ve kterém se mají nastavit proměnné na daném řádku. Celý test tedy probíhá postupně po řádcích, dle zadaného času.

10. FÁZE KOMUNIKACE MEZI PC A ISTP

V této části jsou popsány jednotlivé fáze programu. Tyto fáze reprezentují určitý stav programu a embedded platformy a definují operace přípustné v tomto stavu.

10.1. KONFIGURAČNÍ FÁZE

Tato fáze začíná vždy požadavkem od PC k nakonfigurování embedded platformy. PC platforma pošle kontrolní paket k nakonfigurování, ISTP odešle ACK a poté začne posílat konfigurační pakety a pakety obsahující komentář. Celá fáze je ukončena kontrolní zprávou o konci konfigurace, která je odeslána embedded platformou. Jestliže je mezi pakety zpoždění větší než 2 s je port uzavřen a tento chybový stav je oznámen uživateli.



Obrázek 10.1 Časový diagram konfigurační fáze

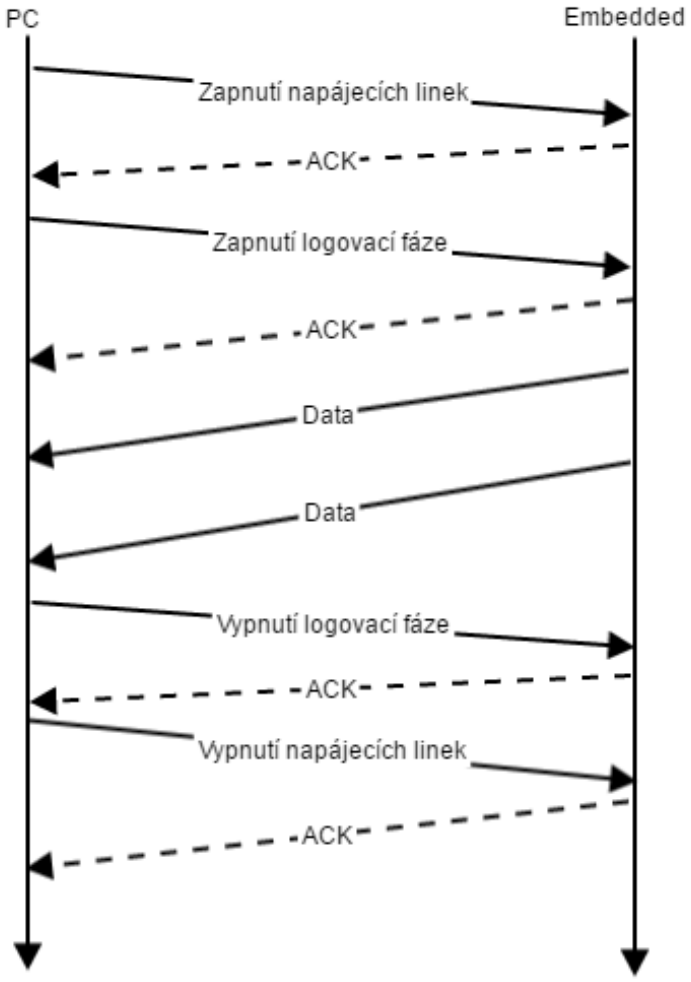
10.2. DATOVÁ (LOGOVACÍ) FÁZE

V této fázi se posílají data ze sensorů. Tato část je hlavní částí a probíhá v ní reálné měření. Tuto část začíná PC aplikace s požadavkem na zapnutí napájecích linek. Embedded část nijak nekontroluje zapnutí těchto linek v případě, že požadavek o zapnutí nedojde. Tohoto omylu se musí vyvarovat PC aplikace. Po odeslání zprávy o zapnutí napájecích linek se ihned pošle zpráva o zapnutí přenosu dat.

Před tím, než je možné číst data ze sensorů, se musí provést jejich inicializace. Ta se provádí po přijetí zprávy o startu posílání dat. Jakmile se provádí inicializace,

předpokládá se, že jsou potřebné napájecí linky pro sensory ve stavu High. Inicializace se provádí pro každou senzorovou desku zvlášť přes její driver. Samotná rychlost inicializace tak závisí na složitosti této operace pro konkrétní sensor a na kvalitě driveru. Pokud je start datové fáze časově kritický je nutné, aby obsluha znala dobu inicializace pro použité senzorové desky.

Jakmile se má, dle testovacího skriptu, ukončit logovací fáze, PC aplikace pošle kontrolní zprávu do platformy. Ta zastaví čtení ze senzorů. Pokud se mají nastavit napájecí linky do stavu Low, tak se tak učiní po odeslání kontrolní zprávy z PC do embedded.



Obrázek 10.2 Časový diagram datové fáze

10.3. DATOVÁ FÍZE – FIFO REŽIM

V rámci tohot režimu se ličší frekvence zaznamenávání dat sensory a frekvence jejich čtení a odesílání do PC. Sensory použitelné pro tento režim mají interní paměť typu FIFO, do které si zaznamenávají data o vysoké frekvenci (zpravidla 1 kHz). Čtecí frekvence je řádově nižší (zpravidla 100 Hz). Data se posílají jako v běžném režimu, ale jejich počet

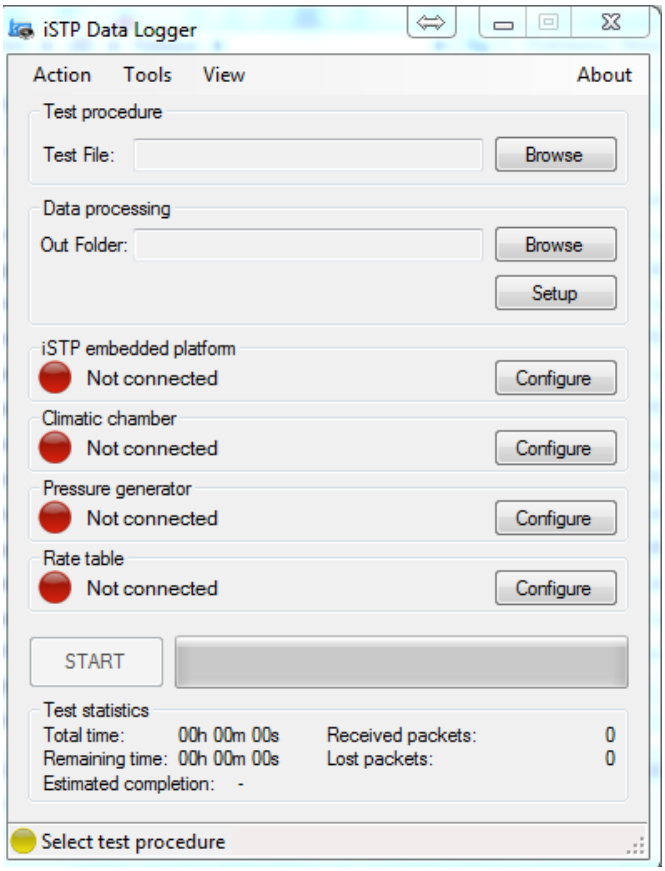
je větší. Reálně se si každý sensor uloží jiné množství dat, i když jsou jejich zaznamenávací frekvence stejné, a proto ISTP zarovnává data dle nejkratšího pole. Zarovnaná data poté ISTP odesílá do PC v pořadí dle konfigurace tak, že první číslo určuje, kolik reálných dat následuje.

Tabulka 10.1 Data v rámci FIFO režimu

Index v poli	0	1	2	3	…
Význam	2	Reálná hodnota	Reálná hodnota	2	…

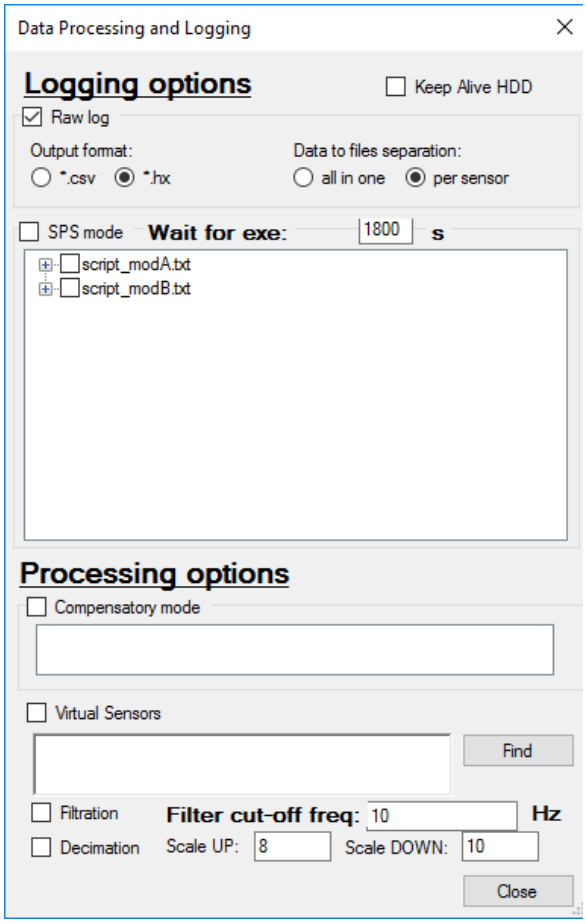
11. PROVÁDĚNÍ TESTŮ

Dle požadavků se vytvoří testovací profil, ve kterém jsou definované parametry pro všechny potřebné periferie. Aplikace nemá možnost zkontrolovat správné časování, a proto je za správné časy odpovědný ten, kdo testovací profil vytváří, na druhou stranu aplikace umí vygenerovat hlavičky potřebné pro každou periferii. Pokud by chyběla jakákoli hlavička pro jakoukoli periferii aplikace ohlásí chybu. Hlavička určuje název parametru, který je v daném sloupci nastavován, přičemž se vždy celý řádek vykonává jako celek v definovaný čas.



Obrázek 11.1 Ukázka hlavního okna PC aplikace

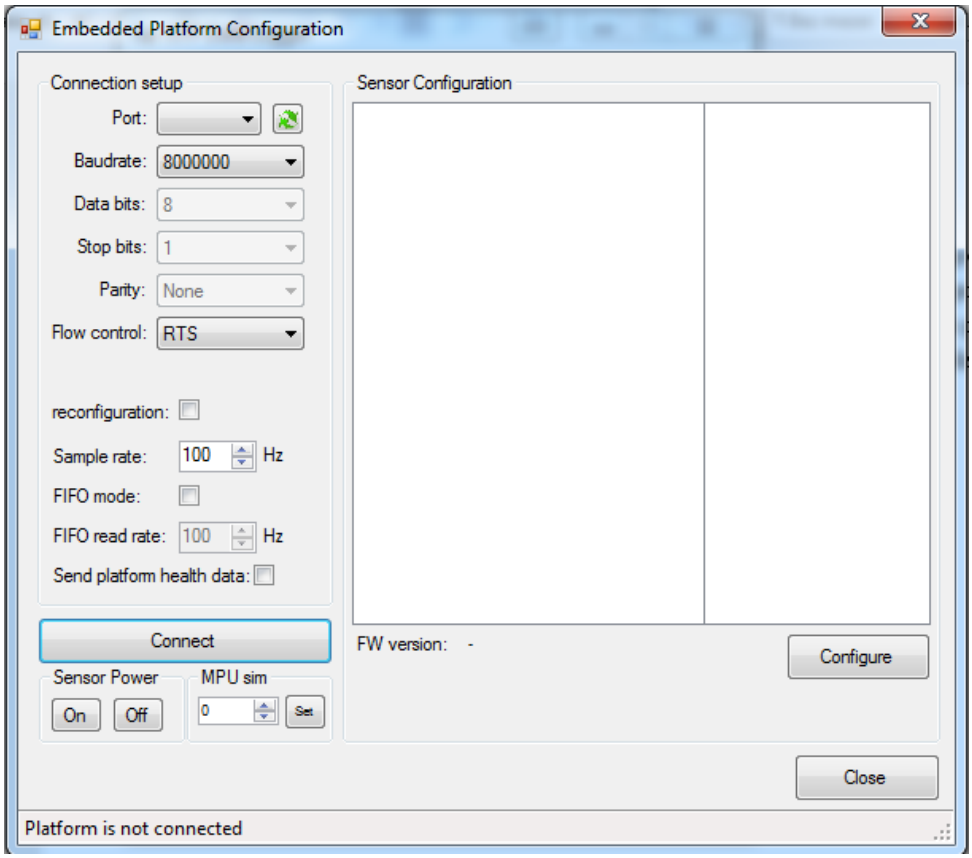
Po načtení testovacího profilu a vybrání cílové složky pro ukládání dat, je na obsluze, aby správně nastavila datové zpracování. Jednotlivé typy zpracování dat v reálném čase jsou na sobě vzájemně nezávislé, např. obsluha může povolit decimaci bez filtru. Pokud požaduje obsluha aplikaci svých externích skriptů je nutné nastavit režim ukládání na SPS, protože jedině v tomto režimu se načte uživatelem definovaný textový dokument, ve kterém je popsáno, jaké skripty, v jakém pořadí a s jakými parametry se budou volat.



Obrázek 11.2 Ukázka okna pro výběr zpracování a ukládání dat

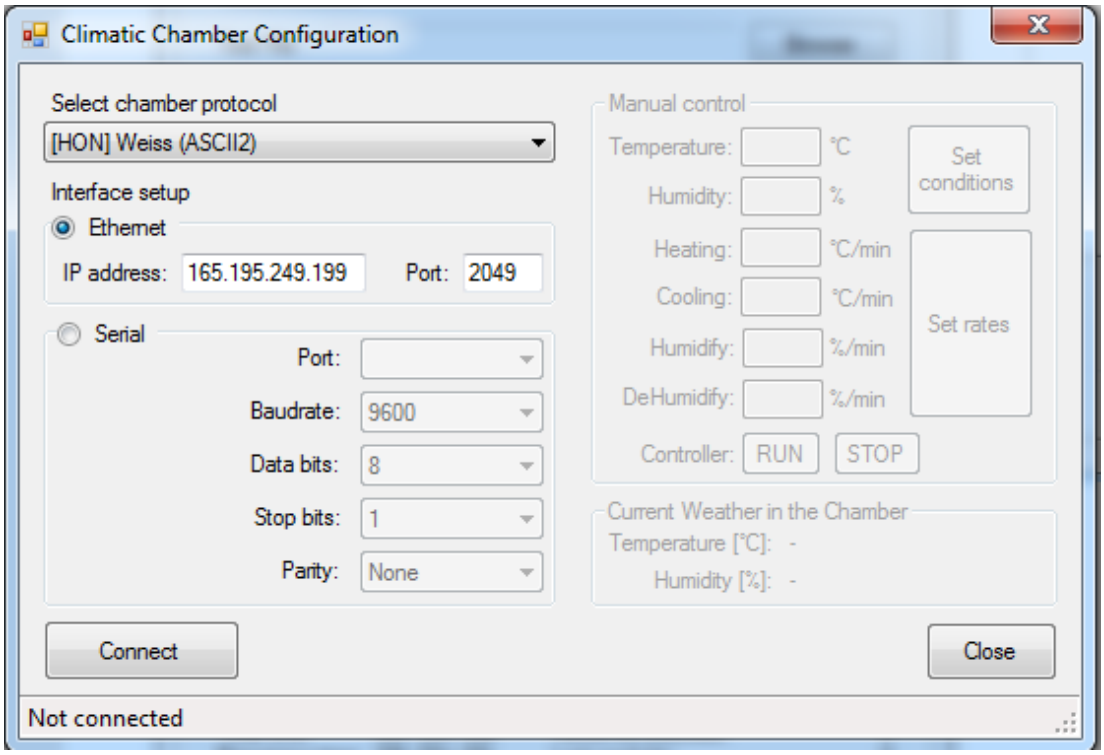
Nyni je na řadě konfigurace periferií potřebných pro správné fungování testu. Vedle každé periferie je symbol červené indikace, která značí že periferie není potřebná k danému testu, ovšem ihned poté co obsluha načte testovací profil, vedle potřebných periferií změní indikace svou barvu na žlutou. Bez nakonfigurování všech žlutých periferií nelze spustit samotný test. Pokud je periferie nakonfigurována, indikace má zelenou barvu.

V konfiguračním okně pro ISTP lze nastavit příslušnou přenosovou rychlost, ovšem ta se musí schodovat s nastavenou přenosovou rychlostí v softwaru již nahraném v ISTP. Volba RTS lze zrušit pro případ, že by obsluha použila jiný typ komunikace než RS422. Možnost ‚reconfiguration' znamená provedení konfigurace při každém zapnutí napájecích linek. Tato volba je užitečná, pokud lze očekávat, že některý z testovaných sensorů by mohl být během testu zničen. Další volbou je ‚Sample rate', která znamená frekvenci, se kterou se budou vyčítat data ze sensorů a v případě neoznačení ‚FIFOmode' jde i o odesílací frekvenci. Volba ‚FIFOmode' zapíná tzv. ‚FIFO režim', který je vysvětlen výše, s ním související ‚FIFO read rate' značí frekvenci, se kterou se budou odesílat data do PC. Nakonec ‚MPU sim' vytvoří zvolený počet simulačních sensorů, které slouží pro testování komunikace.



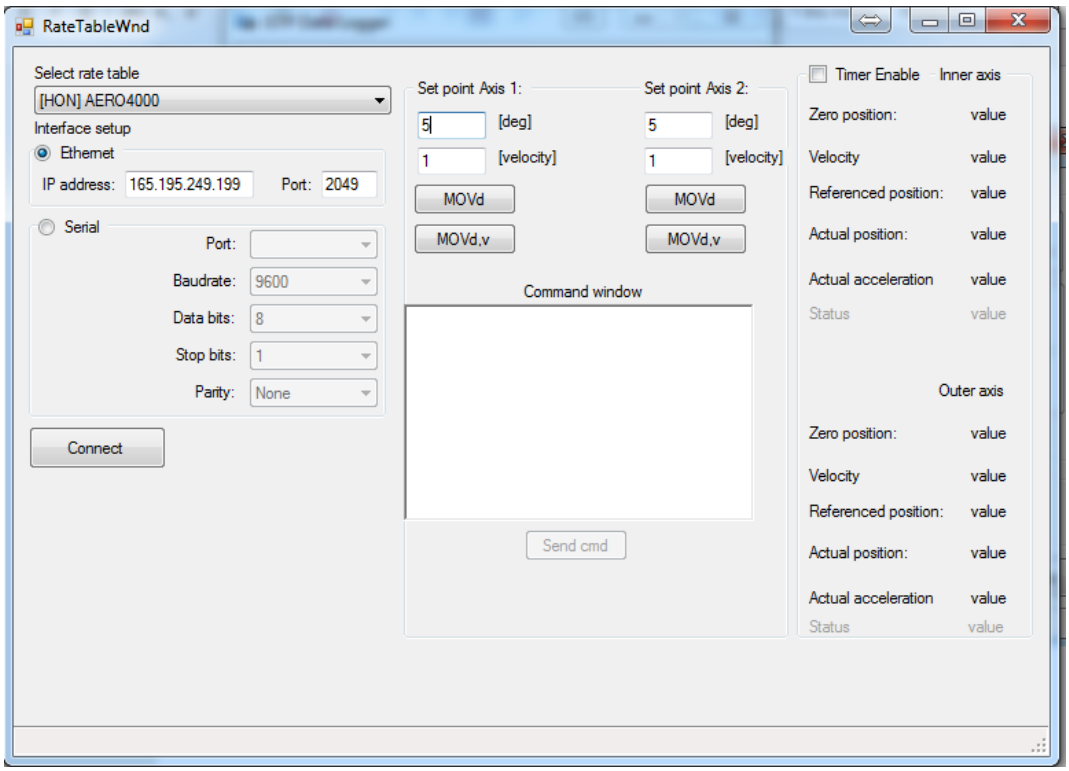
Obrázek 11.3 Ukázka okna pro nakonfigurování ISTP

Další možnou periferií je klimatická komora. Konkrétní typ záleží na všech přítomných diverech v programu. Mezi nimi zle přepínat hlavním listem umístěným v horní částí okna. Po vybrání typu komunikace RS232 nebo Ethernet a připojení se odemkne manuální řízení komory, kde si uživatel může ověřit správné fungování příslušného driveru.



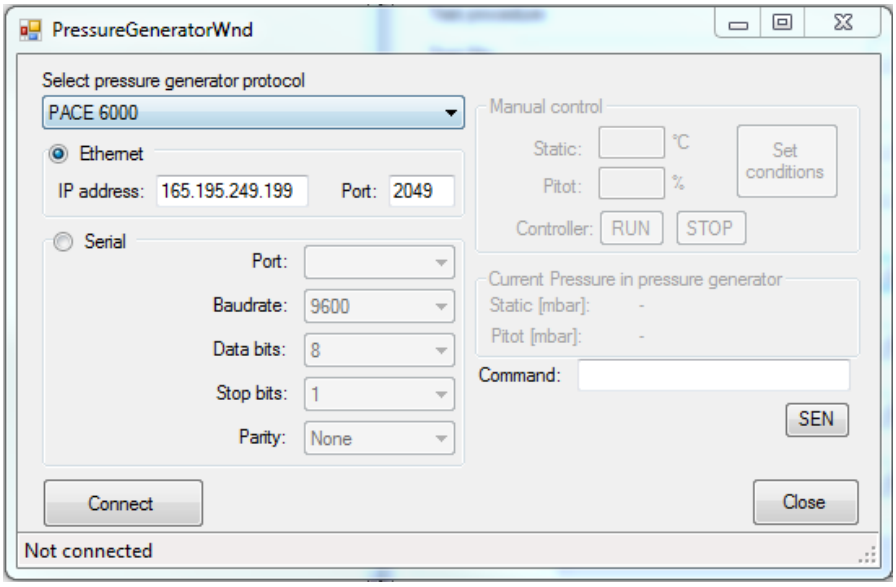
Obrázek 11.4 Ukázka okna pro nakonfigurování klimatické komory

Rotační stůl má stejnou možnost výběru jako klimatická komora. Po zvolení správného driveru, k terý záleží na typu rotačního stolu a po úspěšném připojení je možné manuálně ovládat zařízení a ověřit si správné fungování driveru.



Obrázek 11.5 Ukázka okna pro nakonfigurování rotačního stolu

Poslední možnou periferií je tlakový generátor. Princip připojení je stejný jako u ostatních periferií.



Obrázek 11.6 Ukázka okna pro nakonfigurování tlakového generátoru

11.1. SPUŠTĚNÍ TESTU

Po nakonfigurování všech potřebných periferí, načtení testovacího profilu a vybrání datového zpracování se obsluze odemkne možnost spustit test. Po jeho spuštění již vše běží automaticky dle načteného testovacího profilu a veškeré informace jsou psány do logovacího okna, které je přichyceno vpravo od hlavního okna. Obsluha si může vyžádat zobrazení ‚Debug window', které slouží k zobrazení podrobných informací v případě vzniku chyby v programu.

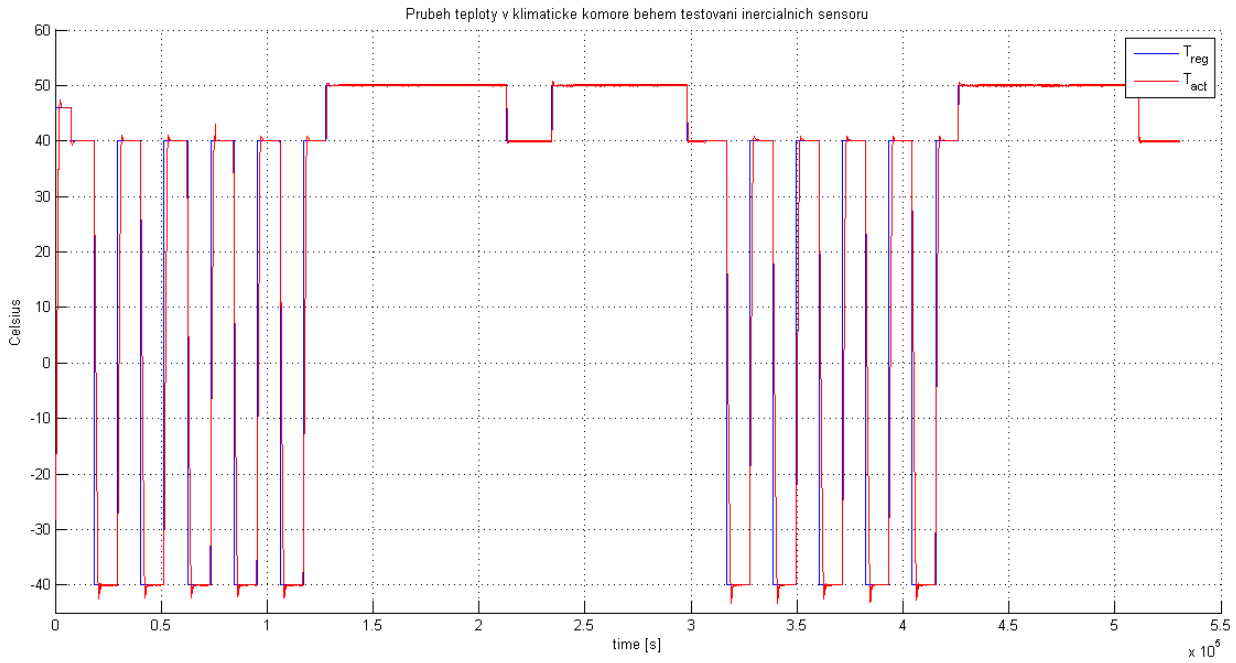
Kromě výsledných souborů s daty se automaticky vytvoří i několik souborů, které doplňují informace o testu, jak ukazuje tabulka 11.1.

Tabulka 11.1 Seznam vytvořených souborů během testu

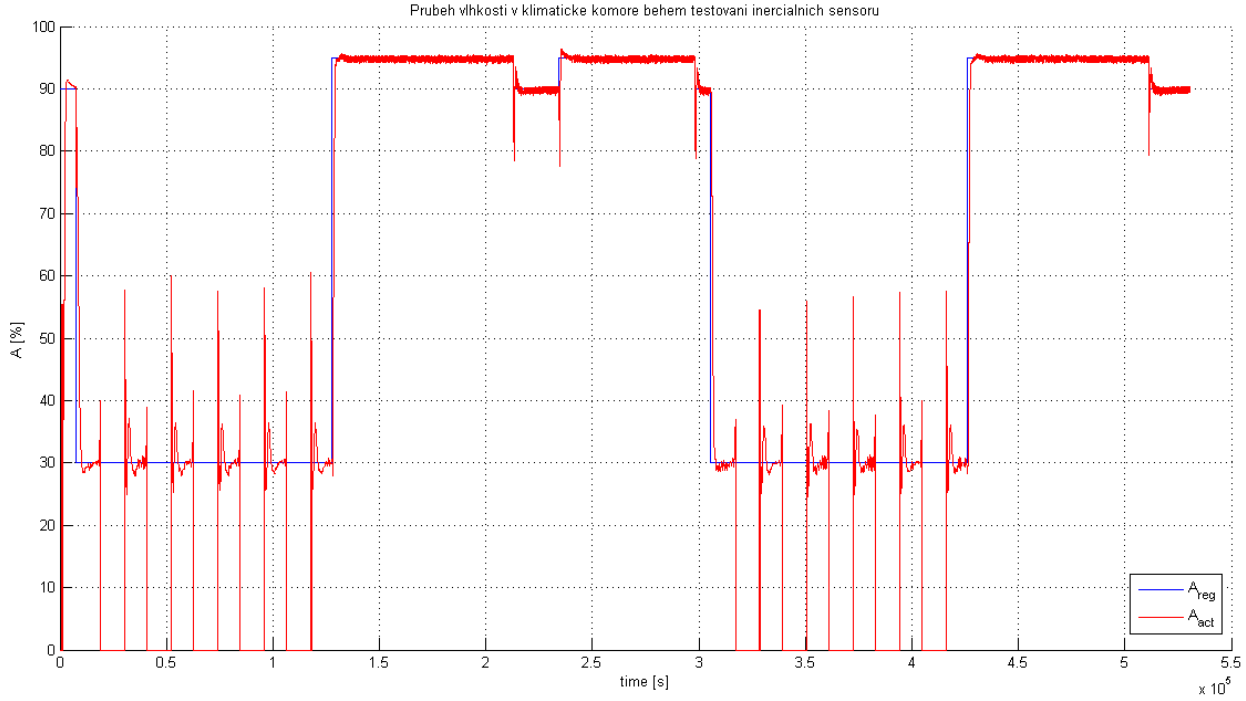
Jméno souboru s příponou	Místo vzniku	Popis
TestLog.txt	Root složka s daty	Záznam logovacího okna
EmbeddedConfig.txt	Root složka s daty, jednotlivé podložky s daty	Kompletní popis poslední konfigurace ISTP
EnviroLog.hx	Root složka s daty	Záznam hodnot s jednotlivých periferií (v ISTP jsou tyto data označována jako ‚Health data')
EnviroLog.csv		
DebugLogger.txt	Složka s programem	Záznam ‚Debug window'
TestProfile.csv	Root složka s daty	Nakopírován spuštený testovací profil

11.2. ZÁZNAM REÁLNÉHO TESTU

Test probíhal v komoře od firmy WEISS jejíchž profil ukazují dva grafy. První graf 11.1 ukazuje průběh teploty v komoře (T_{reg} … teplota požadovaná PC programem dle testovacího profilu. T_{act} … skutečná teplota v komoře). Druhý graf 11.2 zobrazuje průběh vlhkosti v klimatické komoře (A_{reg}… vlhkost požadovaná PC programem dle testovacího profilu. A_{act}… skutečná vlhkost v komoře).



Obrázek 11.7 Graf průběhu teploty v klimatické komoře.



Obrázek 11.8 Graf průběhu vlhkosti v klimatické komoře

Během testu bylo testováno 19 sensorů v SPS režimu, bez FIFO režimu a bez zpracování dat v reálném čase, se čtecí frekvencí 100 Hz. Tři sensory obsahovali jak akcelerometr tak gyroskop, ostatní byly čistě akcelerometry nebo gyroskopy a vytvořili SW program je spároval do 8 dvojic. Každému sensoru (páru) program přiřadil jméno, pod kterým jsou uložena data z něj (nich) od S01 po S11. Pro SPS byla zvolena teplotní podmínka nastavená na 80 °C s maximálním časem 3 minuty. Celkově se provedlo 20 měřicích cyklů rozdělených do dvou skupin po 10, jak lze vyčíst s obou grafů pro klimatickou komoru. Délka jednoho cyklu byla 3 hodiny rozdělených do 2 hodin pro logování dat a 1 hodinou vyhrazenou pro změnu a ustálení teploty v klimatické komoře. V příloze A jsou zobrazeny krátké výsledky sensorů S01, S02 a S03. Sensor S01 je obsažen na hlavní embedded desce vedle MCU a není externě vyhříván. Sensory S02 a S03 jsou externě vyhřívány na 80 °C.

12. ZÁVĚR

Mou prací bylo vytvoření softwarvého řešení a jeho následná implementace na již vytvořený hardware Ing. Maximiliánem Tydorem [1]. Softwarové řešení je realizováno jak pro platformu Windows, tak pro ISTP.

Hlavní program lze spustit na platformě Windows (testováno na Windows 7 Enterprise) a jeho funkcí je komplexní obsluha všech periferií, řízení celého procesu dle uživatelem definovaného testu a zpracování dat v reálném čase. Tento program je zcela nezávislý na sensorech připojených na ISTP, protože pracuje pouze s indexy v datovém poli v rámci příchozích dat ze sensorů. Komunikaci mezi hlavním programem a ISTP zajišťuje mnou vymyšlený protokol, přes který i lze měnit jednotlivá interní nastavení v sensorech. Obsluha periferií je zajištěna preš programové řídicí třídy, které komunikují přes rozhraní s konkrétním ovladačem. Díky vytvořenému rozhraní je jednoduchá budoucí implementace nového ovladače. Hlavní program umí zpracování dat v reálném čase jako je decimace, kompenzace a filtrace, kromě decimace jsou zpracování řešena spouštěním externího souboru DLL, aby je šlo měnit dle požadavků obsluhy a bez zásahu do programu. Externí skripty umí hlavní program spustit pouze v režimu SPS, ovšem parametry spuštení, včetně pořadí, lze měnit přes textový dokument. Program je schopen provádět i testy, které trvají 14 dní bez přestávky.

Jádro programu pro ISTP zajíšťuje identifikaci všech připojených sensorů, má implementovaný komunikační protokol a umí odhalit a upozornit hlavní program na vzniklé hlavní chyby. Přidání nového ovladače na sensorovou desku je snadné, protože každý ovladač je reprezentován svým unikátním číslem, které je součátí EEPROM. Jádro využívá toho, že každá sensorová deska obsahuje paměť typu EEPROM, ve které jsou uloženy veškeré informace o sensorech připojených na sensorové desce.

Zadání jsem splnil a úspěšně implenetoval jak softwarové jádro pro ISTP, tak kompletní řídicí program, který je schopen obsluhovat zároveň ISTP, klimatickou komoru, rotační stůl a tlakový generátor. Program je schopen provádět i 14 denní testy bez chyby způsobující znehodnocení výsledných dat.

13. CITACE

[1] *TYDOR, M. Univerzální senzorová testovací platforma. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2015. 57 s., 0 s. příloh. Diplomová práce. Vedoucí práce: doc. Ing. Jiří Šebesta, Ph.D.*

[2] *Atmel SAM4S Series: SMART ARM-based Flash MCU. Rev.: Atmel-11100K-ATARM-SAM4S-Datasheet_09-Jun-15. 1600 Technology Drive, San Jose, CA 95110 USA: Atmel Corporation, 2015.*

[3] *Atmel SMART Microcontrollers: SAM4S Xplained Pro - User Guide. Rev.: Atmel-42075C-SAM4S-Xplained-Pro_User Guide-06/2015. 1600 Technology Drive, San Jose, CA 95110 USA: Atmel Corporation, 2015.*

[4] *RIPKA, Pavel. a Alois. TIPEK. Modern sensors handbook. Newport Beach, CA: ISTE USA, 2007. ISBN 1905209665.*

[5] *IEEE Standard for Floating-Point Arithmetic," in IEEE Std 754-2008 , vol., no., pp.1-70, Aug. 29 2008 doi: 10.1109/IEEESTD.2008.4610935*
URL: http://bit.ly/2pLlF4h

[6] *Series F4S/D User's Manual* [PDF]. WATLOW INC. April 2004. , 152s [cit. 2017-02-24]. 0600-0032-0000 Rev G. Dostupné z: http://bit.ly/2qsatpt

[7] *Manual, ATL Interface*. Ideal Aerosmith, Inc. 2015, 112s. 231050-254.E.

[8] *AERO 4000 Controller User's Guide*. Ideal Aerosmith, Inc. 2015, 114s. 230700-251.Q.

[9] *Simply Modbus* [online]. 2015 [cit. 2017-02-24]. Dostupné z: http://simplymodbus.ca/

[10] *MODBUS APPLICATION PROTOCOL SPECIFICATION* [online]. In: MODBUS ORGANIZATION, INC. 2012, 50s [cit. 2017-02-24]. V1.1b3. Dostupné z: http://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[11] SKOG, Isaac a Peter HANDEL. *CALIBRATION OF A MEMS INERTIAL MEASUREMENT UNIT* [online]. XVII IMEKO WORLD CONGRESS, Rio de Janeiro, Brazil, 2006 [cit. 2017-03-16]. Dostupné z: http://bit.ly/2mxxJRw

[12] HÁJEK, L. a T. HAMBÁLEK. *Gyroskopy*. Praha, 2010. Dostupné také z: http://bit.ly/2nAepar . Fakulta jaderná a fyzikálně inženýrská.

[13] Gyroscope: How a Gyro Works. *SparkFun Electronics* ® [online]. [cit. 2017-03-19]. Dostupné z: http://bit.ly/2nzYLM7

[14] PROF. DR. FÖLL, Helmut. *Semiconductor Technology: Semiconductor Technology and Nano Electronics*. 262 s. Dostupné také z: http://bit.ly/2n3nffx . University of Kiel, Faculty of Engineering, AMAT. Str. 181 - 183.

[15] MÜNCHOW, Andreas, Ph.D. *Geophysical Fluid Dynamics*. Prentice Hall Inc., Englewood Cliffs, NJ, 1994, 322 s. Dostupné také z: http://bit.ly/2nalr4D . College of Earth, Ocean, and Environment University of Delaware.

[16] ĎAĎO, Stanislav a Marcel KREIDL. *Senzory a měřicí obvody*. Praha: Vydavatelství ČVUT, 1996. ISBN 80-01-01500-9.

[17] ESWARAN, P a S MALARVIZHI. *MEMS Capacitive Pressure Sensors: A Review on Recent Development and Prospective* [online]. Department of Electronics and Communication Engineering, SRM University Faculty of Engineering and Technology, Chennai, India, 2013 [cit. 2017-03-23]. ISSN 0975-4024. Dostupné z: http://bit.ly/2mvDCmg

[18] RIPKA, Pavel. *Senzory a převodníky*. 2. vyd. V Praze: České vysoké učení technické, 2011. ISBN 978-80-01-04696-8.

[19] BIDGOLI, Hossein, ed. *Encyclopedia of information systems*. San Diego: Academic Press, c2003. ISBN 0-12-227244-7.

[20] *Direct Industry: The Online Industrial Exhibition* [online]. 2017 [cit. 2017-03-23]. Dostupné z: http://www.directindustry.com/

[21] BORRELLI, Chris. *IEEE 802.3 Cyclic Redundancy Check* [online]. In: . 2001 [cit. 2017-03-28]. Dostupné z: http://bit.ly/2nfCIXx

[22] CYPRESS SEMICONDUCTOR CORPORATION. *Cyclic Redundancy Check (CRC)*. Document Number: 001-62889 Rev. D. 2016. Dostupné také z: http://bit.ly/2ne9qI1

[23] EPPLER, Barry. *A beginner's guide to SCPI*. Reading, Mass.: Addison-Wesley, c1991. ISBN 0-201-56350-9.

[24] *Pressure Automated Calibration Equipment: SCPI Remote Communications Manual* [online]. K0472 Revision A. General Electric Company, 2015 [cit. 2017-03-30]. Dostupné z: http://bit.ly/2odVkLn

[25] ANDREJAŠIČ, Matej. *MEMS ACCELEROMETERS* [online]. University of Ljubljana, 2008 [cit. 2017-03-30]. Dostupné z: http://bit.ly/2oCCQAn. Seminární práce.

University of Ljubljana, Faculty for mathematics and physics, Department of physics. Vedoucí práce Doc. dr. Igor Pobera.

[26] WOODMAN, Oliver J. *An introduction to inertial navigation* [online]. In: . UCAM-CL-TR-696. University of Cambridge, Computer Laboratory, 2007, s. 37 [cit. 2017-04-07]. ISSN 1476-2986.

[27] Weiss Umwelttechnik GmbH *Ovládací jednotka Touchpanel 8*". Verze 64636051 cs.

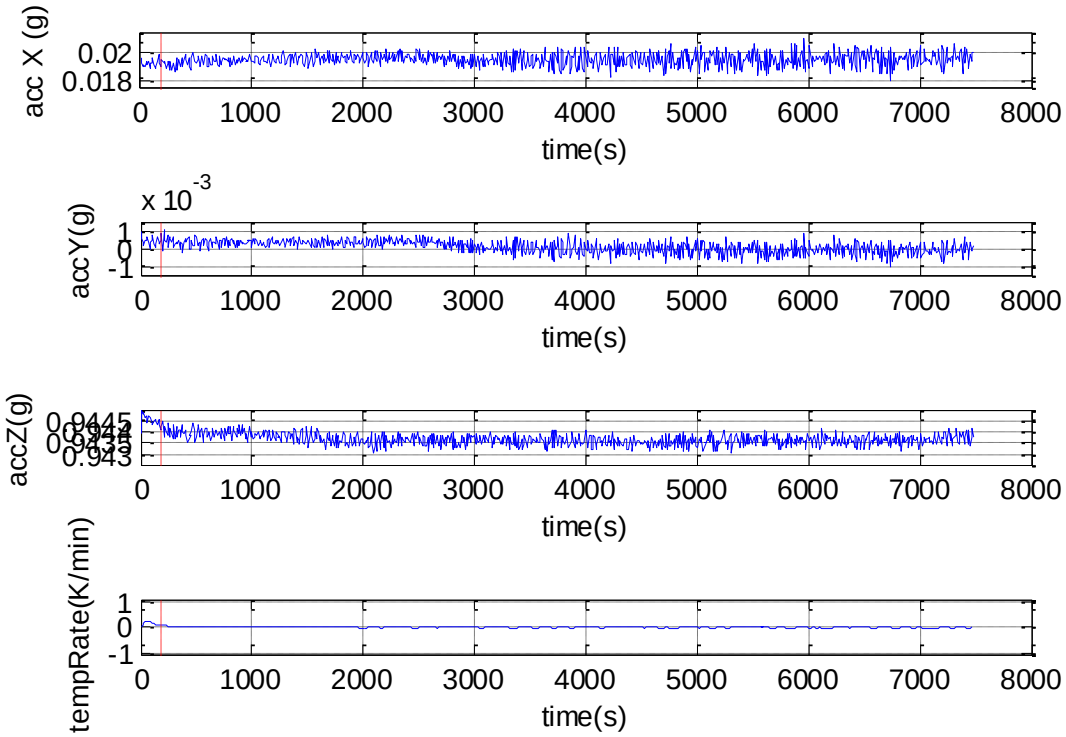
14. SEZNAM ZKRATEK

ACK	-	Acknowledgement
AD	-	Analog to Digital Converter
ATL	-	Aerosmith Table Language
BIN	-	Binární formát vytvořený pro SPS režim
CRC	-	Cyclic Redundancy Check
CRCCU	-	Cyclic Redundancy Check Calculation Uni
CSV	-	Comma-separated values
DMA	-	Direct memory access
DP	-	Diplomová práce
EEPROM	-	Electrical Erasable Programmable Read-Only Memory
FIFO	-	First in First out
GUI	-	Graphical User Interface
HW	-	Hardware
HX	-	Datový formát Honeyx
ID	-	Identifikační číslo
IMU	-	Inertial Measurement Unit
INS	-	Inertial Navigation System
ISTP	-	Inertial Sensor Test Platform
MCU	-	Microcontroller
MEMS	-	Micro-Electro-Mechanical Systems
MISO	-	Master In, Slave Out
MOSI	-	Master Out, Slave In
NSS	-	Slave Select
PC	-	Osobní počítač
PLC	-	Programmable controller
RAM	-	Random-Access Memory
SPCK	-	Serial Clockmpu
SPI	-	Serial Peripheral interface
SPS	-	Sensor Preselection Station
SW	-	Software
TC0	-	Timer Counter 0
UI	-	User Interface

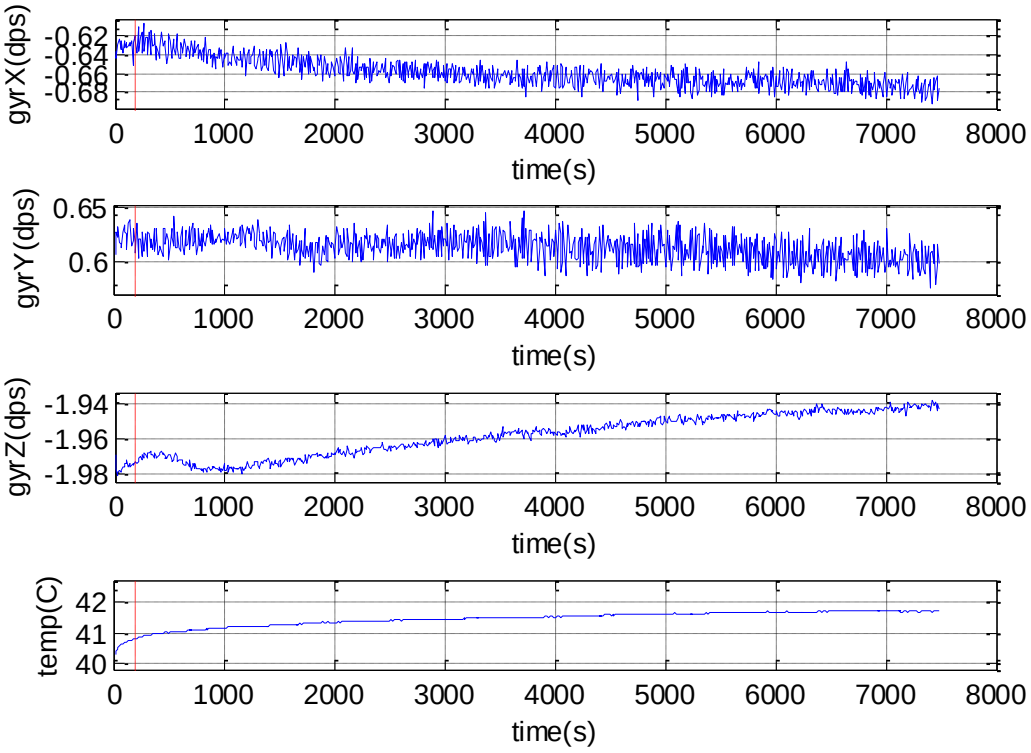
15. SEZNAM PŘÍLOH

Příloha A	-	Reálná data ze sensorů zaznemanená vytvořeným PC a embedded programem a vytisknuta externím skriptem v reřimu SPS
Obsah CD	-	xusnul00.pdf – DP iSTP_Application – PC aplikace v C# ISTP_embeddedCode – Kód pro ISTP v C

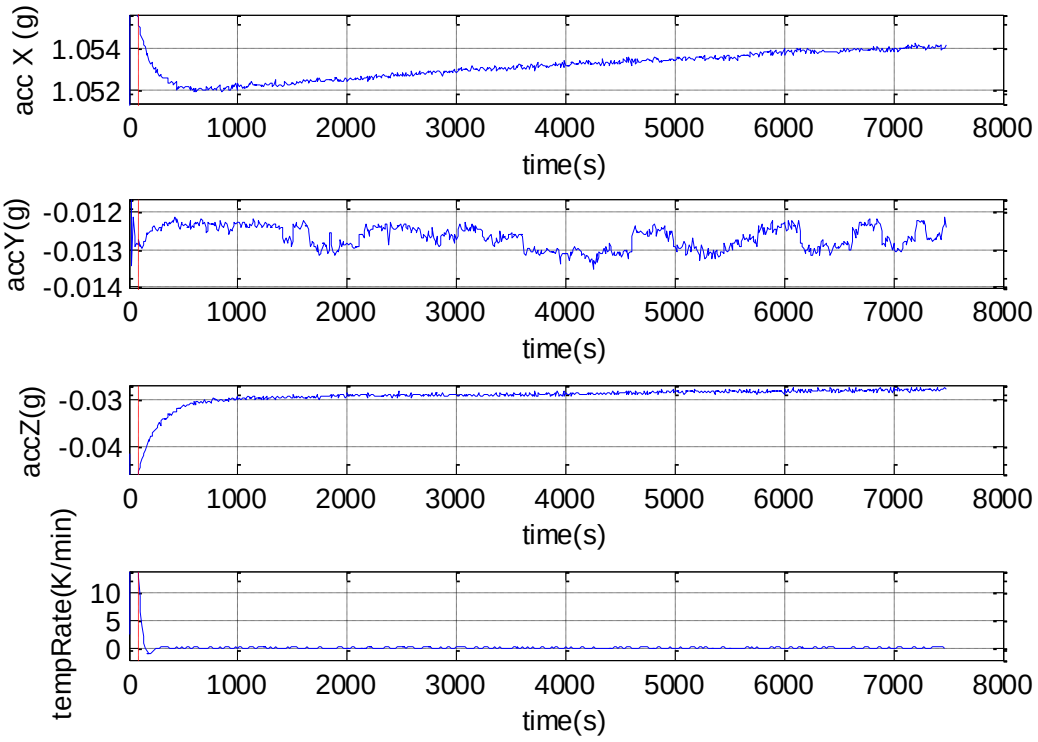
16. PŘÍLOHA A



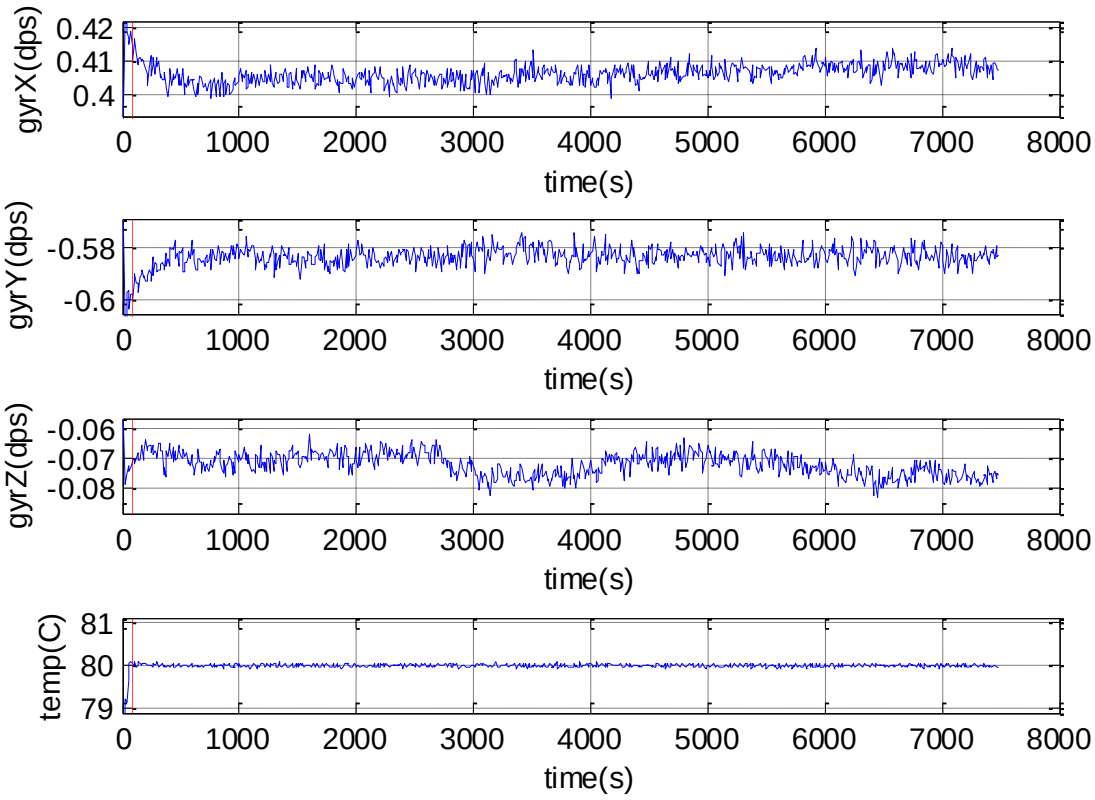
Obrázek 16.1 Průměrné hodnoty z akcelerometru sensoru S01



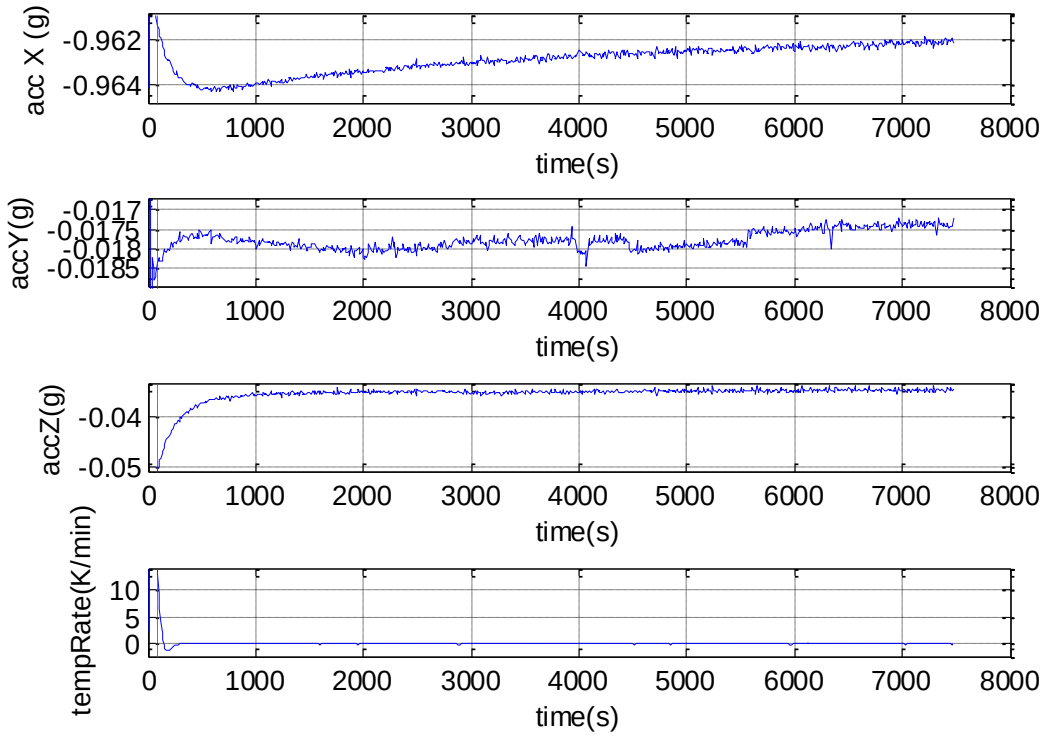
Obrázek 16.2 Průměrné hodnoty z gyroskopu sensoru S01



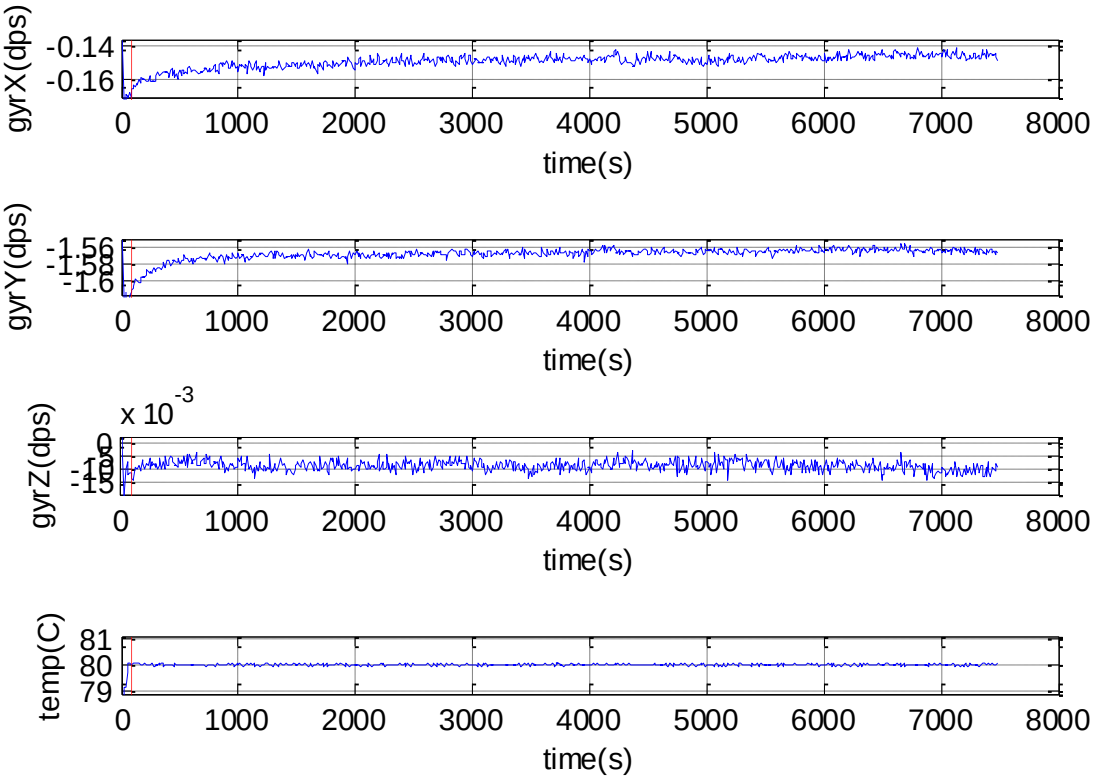
Obrázek 16.3 Průměrné hodnoty z akcelerometru sensoru S02



Obrázek 16.4 Průměrné hodnoty z gyroskopu sensoru S02



Obrázek 16.5 Průměrné hodnoty z akcelerometru ze sensoru S03



Obrázek 16.6 Průměrné hodnoty z gyroskopu ze sensoru S03