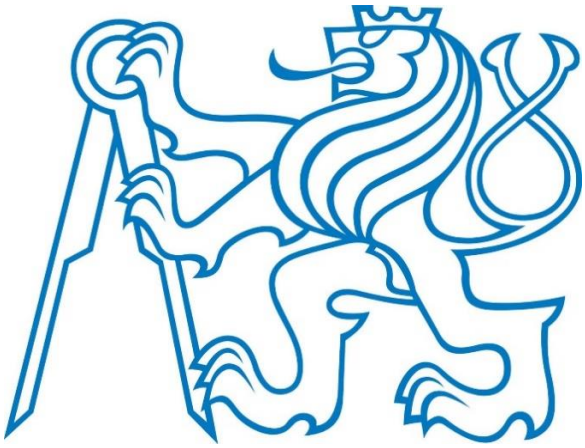
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra telekomunikační techniky



Kyberfyzikální bezpečnost pseudosatelitů

Cyberphysical Security of Pseudo-satellites

Diplomová práce

Bc. Filip Bělohlávek

Studijní program: Elektronika a komunikace
Specializace: Komunikační sítě a internet

Vedoucí práce: Ing. Bc. Marek Neruda, Ph.D.

Květen 2022
Praha

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bělohlávek** Jméno: **Filip** Osobní číslo: **474228**

Fakulta/ústav: **Fakulta elektrotechnická**

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

Studijní program: **Elektronika a komunikace**

Specializace: **Komunikační sítě a internet**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Kyberfyzikální bezpečnost pseudosatelitů

Název diplomové práce anglicky:

Cyberphysical Security of Pseudo-satellites

Pokyny pro vypracování:

Seznamte se s konceptem pseudo satelitů HAPS. Proveďte analýzu kyberfyzikálních hrozeb pseudo satelitů HAPS a zhodnoťte rizika potenciálních útoků. Ve spolupráci se zadavatelem se zaměřte na jednu, či více identifikovaných hrozeb, navrhněte vhodné bezpečnostní mechanismy a vybrané bezpečnostní mechanismy realizujte na embedded PC na úrovni „proof of concept“.

Seznam doporučené literatury:

[1] Karabulut Kurt, Gunes, Mohammad G. Khoshkholgh, Safwan Alfattani, Ahmed Ibrahim, Tasneem S. J. Darwish, Md. Sahabul Alam, Halim Yanikomeroglu and Abbas Yongaçoglu. “A Vision and Framework for the High Altitude Platform Station (HAPS) Networks of the Future.” IEEE Communications Surveys & Tutorials 23, 2021: p.729-779.
[2] Cárdenas, Alvaro A. and Nils Ole Tippenhauer. “Cyber-Physical Systems Security Knowledge Area.” University of California, Santa Cruz (2019).

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Marek Neruda, Ph.D. katedra telekomunikační techniky FEL

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

Datum zadání diplomové práce: **21.01.2022** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **30.09.2023**

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

III. PŘEVZETÍ ZADÁNÍ

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

. | Datum převzetí zadání

Podpis studenta

Prohlášení

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

V Praze, dne 20. 5. 2022 ……

Bc. Filip Bělohlávek

Poděkování

Rád bych tímto poděkoval Ing. Marku Nerudovi, Ph.D. za vedení této práce, za jeho trpělivost a velmi cenné rady a poznámky, stejně jako za přátelskou atmosféru, v jejíž duchu se tvorba této diplomové práce nesla. Dále mé poděkování patří Ing. Tomáši Vaňkovi, Ph.D. za přínosné a velice profesionální konzultace z oboru kyberbezpečnosti. V neposlední řadě chci poděkovat svým nejbližším, rodině a přátelům za jejich podporu, která mi byla velice potřebnou oporou.

Abstrakt

Tato diplomová práce pojednává o problematice kyberfyzikální bezpečnosti stratosferických pseudosatelitů (HAPS). HAPS zde figuruje jako zástupce tzv. kyberfyzikálních systémů, tedy systémů, které spojují fyzický svět se světem kybernetickým. S konceptem HAPS se počítá při návrhu budoucích mobilních 6G sítí. Problém kyberfyzikální bezpečnosti HAPS je vnímán zejména v zajištění vhodné fúze kybernetické a fyzické bezpečnosti. Představeným řešením tohoto problému je návrh zabezpečené architektury HAPS. Navržená architektura spojuje metody kybernetické i fyzické bezpečnosti. Jádrem zabezpečené architektury HAPS je navržená komponenta SCC, která zajišťuje setrvání HAPSu a jeho komponent v definovaném bezpečném stavu. Navržená architektura je podrobena analýze hrozeb pomocí modelu STRIDE, identifikované hrozby jsou ohodnoceny pomocí modelu DREAD. Na základě identifikovaných hrozeb jsou navržena další vylepšení představené architektury posilující její bezpečnost. Dále se práce věnuje teoretickému návrhu komponenty SCC. Komponenta SCC je následně realizována na úrovni „proof of concept“ a je otestována její funkčnost.

Klíčová slova

Architektura HAPS, bezpečnost HAPS, HAPS, kyberfyzikální bezpečnost, kyberfyzikální systémy, pseudosatelity

Summary

This thesis deals with the problematic of cyberphysical security of stratospheric pseudo-satellites (HAPS). HAPS is seen as a representative of so-called cyberphysical systems, i.e. systems that connect the physical world with the cyber world. The HAPS concept is being considered in the design of future 6G mobile networks. The problem of cyberphysical security of HAPS is seen primarily in ensuring an appropriate fusion of cybersecurity and physical security. The presented solution to this problem is the design of a secure HAPS architecture. The proposed architecture combines methods of both cyber and physical security. The core of the secure HAPS architecture is the proposed SCC component, which ensures that HAPS and its components remain in a defined safe state. The proposed architecture is subjected to threat analysis using the STRIDE model, and the identified threats are evaluated using the DREAD model. Based on the identified threats, further enhancements to the proposed architecture are suggested to strengthen its security. Furthermore, the theoretical design of the SCC component is discussed in the thesis. The SCC component is then implemented at the ”proof of concept” level and its functionality is tested.

Index Terms

HAPS architecture, HAPS security, HAPS, cyberphysical security, cyberphysical systems, pseudo-satellites

Obsah

Seznam obrázků a tabulek

Obrázky

Tabulky

Seznam zkratek

ACL	Access Control List
AES	Advanced Encryption Standard
AES-NI	AES New Instructions (instruction set)
API	Application Programming Interface
BLE	Bluetooth Low Energy
CA	Certificate authority
CAN	Controller Area Network
CPS	Cyber Physical System
CPU	Central Processing Unit
DFD	Data Flow Diagram
DNN	Deep Neural Network
DoS	Denial of Service
DPI	Deep Packet Inspection
FW	Firmware
GCM	Galois Counter Mode
GPIO	General-purpose input/output
GPS	Global Positioning System
GS	Ground Station
GSHU	Ground Station Human User
GSPC	Ground Station PC
GSPC2	PC in GS LAN
GUI	Graphical User Interface
HAPS	High-altitude pseudo-satellite || High-altitude platform station
HFC	HAPS Flight Control
HFS	HAPS File System
HSC	HAPS System Control
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol
HW	Hardware
IAEA	International Atomic Energy Agency

IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPS	Intrusion Prevention System
ITU	International Tlecommunication Union
JSON	JavaScript Object Notation
LAN	Local Area Network
LEO	Low Earth Orbit
MEC	Multi-Access Edge Computing
MEMS	Micro Electro Mechanical Systems
MITM	Man in the Middle
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MTTF	Mean Time to Failure
OS	Operating System
P2P	Peer-to-peer
PKI	Public Key Infrastructure
PL	Path Loss
PLC	Programmable Logic Controller
PoC	Proof of Concept
PWM	Pulse Width Modulation
RCE	Remote Code Execution
SAN	Sensors/Actuators Network
SCADA	Supervisory Control And Data Acquisition
SCC	Secure CPS Controller
SPOF	Single Point of Failure
SQL	Structured Query Language
SW	Software
TC	Threat Consequence
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UAV	Unmanned Aerial Vehicle

UE	User Equipment
USB	Universal Serial Bus
VLAN	Virtual LAN
VM	Virtual Machine
WPA2	Wi-Fi Protected Access 2
WSN	Wireless Sensor Network

Kapitola 1
Úvod

Propojení počítačového a fyzického světa se stává trendem poslední doby. Snahy o toto propojení vedly ke vzniku konceptu kyberfyzikálních systémů (CPS – Cyber-Physical Systems). V kyberfyzikálních systémech jsou počítačové a síťové části systému úzce propojeny s určitým fyzickým procesem. Existuje zde zpětná vazba mezi světem fyzickým a digitálním v podobě senzorů. Digitální část systému je schopná reagovat a ovlivnit fyzický svět pomocí akčních členů. Fyzické procesy jsou tedy schopné ovlivnit digitální procesy, a naopak [1].

CPS nejsou limitované svou velikostí či rozsáhlostí. Může se tedy jednat o jednoduché malé zařízení, stejně tak se může jednat o rozlehlý systém, například v podobě inteligentní elektrické sítě. Myšlenka CPS tak nabízí velice široké spektrum užitečných aplikací. Například v [2] jsou uvedeny již zmiňované inteligentní elektrické sítě, inteligentní transportní systémy, či aplikace v aeronautice. Zmiňován je i přínos kyberfyzikálních systémů v oblasti zdravotnických zařízení, kde mohou bezdrátové senzorové sítě (WSN – Wireless Sensor Network) sbírat data o zdravotním stavu pacientů. Systém v případě potřeby může odeslat upozornění zdravotnímu personálu, či v některých případech autonomně vykonat na pacientovi drobný úkon – například v podobě podání léků. Aplikace CPS se však také nalézt v chytrých domácnostech, městech, či v autonomních vozidlech.

Hlavní podstata CPS, kterou je ono úzké propojení počítačů, sítí a fyzických procesů, přináší nová úskalí. Jedním z velkých problémů je bezpečnost CPS. Je to právě ono propojení digitálního a fyzického světa, díky němuž se otevírají zcela nové hrozby, které by mohly mít až katastrofální důsledky ve světě, ve kterém žijeme. Například již zmíněná teoretická aplikace CPS v podobě podávání léků pacientovi, by mohla mít fatální následky, pokud by byl útočník schopen získat kontrolu nad akčním členem vykonávající samotné podávání léků. Kybernetická bezpečnost řeší bezpečnost digitálních systémů, bezpečnost informace. Naproti tomu zde je klasické pojetí fyzické bezpečnosti různých strojů, produktů, či systémů v reálném světě – tj. zajištění takové bezpečnosti systému, aby nebyl nebezpečný pro své okolí, a aby při poruše, či jiné nečekané události, nezpůsobil systém další škody. Problémem bezpečnosti CPS je fakt, že tyto dva oddělené přístupy, tedy kybernetická a fyzická bezpečnost, nejsou dostatečné.

V rámci kyberfyzikální bezpečnosti je potřeba uvažovat komplexní systém jako celek a bezpečnost CPS musí být integrována již v návrhu. V návrhu CPS je potřeba implementovat takové bezpečnostní mechanismy, které v duchu myšlenky CPS svazují kybernetickou a fyzickou bezpečnost. Absolutní bezpečnost neexistuje ani ve světě digitálním, ani v tom fyzickém. Důležité je minimalizovat rizika. Kybernetickou bezpečnost můžeme velkou mírou ovlivnit, ale ani ty nejlépe navržené systémy nejsou absolutně bezpečné, a jejich bezpečnost klesá s časem. Díky tomuto faktu je při návrhu CPS potřeba myslet na to, že digitální část systému může být útočníkem napadnuta a potenciálně nad ní i může získat kontrolu. Je ale nutné zajistit, aby v těchto případech byly minimalizovány možnosti útočníka v kontrole fyzického systému.

Nově vznikajícím CPS systémem, navrhovaným v 6G mobilních sítích, je „High-altitude pseudo-satellite“ nebo také „High-altitude platform station“ (HAPS). Dle definice ITU článku *1.66A* je HAPS „stanice přítomná na objektu, který se nachází ve výšce od 20 km do 50 km na specifikovaném fixním bodě relativně vůči zemi“ [3]. Jedná se tedy o platformu vznášející se ve stratosféře, která operuje jako satelit. Koncept HAPS má nejčastěji buď podobu jakéhosi balónu, který se vznáší pomocí lehkých plynů, či lehkého letounu. Díky energii ze solárních panelů může HAPS bez jakýchkoliv emisí zůstat ve stratosféře týdny až několik měsíců. Oproti satelitům je jejich vypouštění do místa, ve kterém operují výrazně levnější a ekologičtější záležitostí. Využití nachází HAPS v mnoha oblastech, a to od pozorování země, přes astronomii, až po telekomunikační aplikace. Právě ty jsou hlavním předmětem zájmu. HAPS bude schopen poskytovat pokrytí mobilní internetem, komunikaci se satelity, či další telekomunikační služby [4]. V této práci tak bude jako důvod nasazení HAPSu uvažována obecná telekomunikační aplikace.

HAPS, jakožto příklad kyberfyzikálního systému, bude muset disponovat počítačem, celou řadou senzorů a akčních členů, které umožní autonomní řízení letu. Nutným předpokladem je také zajištění síťové konektivity k pozemní stanici.

Kyberfyzikální bezpečnost HAPS je tématem této práce. Jak již bylo řečeno, útoky na CPS mohou mít dalekosáhlé důsledky na reálný fyzický svět. U HAPS tomu není jinak. Díky umístění HAPSu na úroveň stratosféry je fyzický útok na HAPS velice nepravděpodobný, ne-li téměř nemožný. Útočník tedy pro útoky využije síťovou komunikaci a útočit bude tzv. z digitálního světa. Bez návrhu odpovídajících bezpečnostních mechanismů by při úspěšném útoku bylo například možné získat kontrolu nad řízením HAPS. Pokud by se toto stalo, scénářů, které pak může útočník realizovat, je vícero. Jedním z těchto scénářů by teoreticky mohl být i nekontrolovatelný sestup HAPSu na zem, který by například ve městech mohl způsobit škody na majetku, či dokonce újmu na zdraví lidí. Hlavním cílem této diplomové práce je tedy navrhnout takový soubor řešení, který posiluje kyberfyzikální bezpečnost systému HAPS.

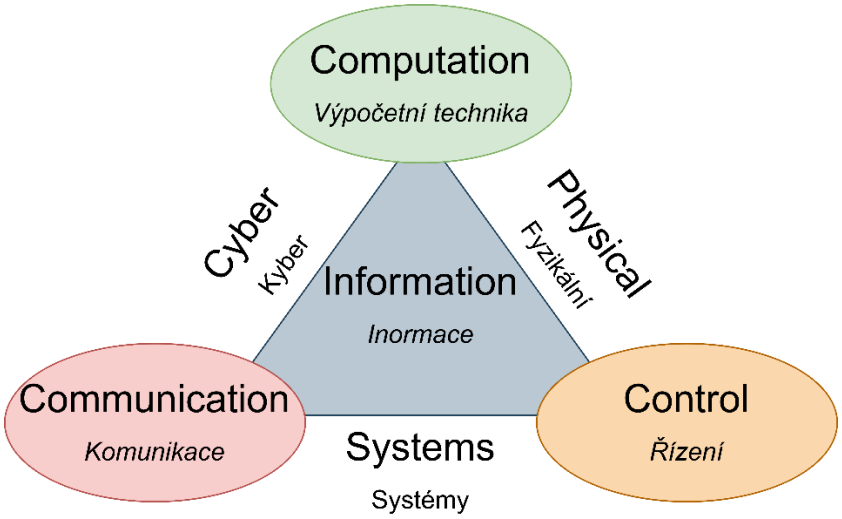
Obr. 1.1 *Koncept HAPS* [5]

Kapitola 2
Kyberfyzikální systémy

2.1 Obecná architektura CPS

Termín CPS byl poprvé použit Dr. Hellen Gillovou v roce 2006. Jak již bylo zmíněno v podkapitole 1.1, CPS představuje propojení fyzických procesů s počítačovými systémy a síťovou konektivitou. Toto propojení je obecně znázorněno na *obr. 2.1*. Tento diagram značí základní architekturu, někdy též zvanou jako 3C – „Computation, Communication, Control“. Interakce výpočetních a fyzických procesů je znázorněna hranou „Physical“ pomyslného 3C trojúhelníku CPS. Jedná se o klíčovou interakci, která počítačovému systému prostřednictvím senzorů umožňuje získávat informace o fyzickém světě, tyto informace zpracovat a na základě těchto informací učinit rozhodnutí, ovlivňující fyzický svět pomocí akčních členů.

Další nezbytnou interakcí v CPS je interakce počítačových systémů mezi sebou. Jedná se o síťovou komunikaci zařízení. Tato interakce je čistě v digitální doméně a tvoří tak druhou hranu v pomyslném trojúhelníku CPS označenou jako „Cyber“. Díky síťové komunikaci počítačových systémů je možné sdílet data navzájem mezi jednotlivými zařízeními. Komunikace je zásadním prvkem 3C architektury umožňující tvorbu inteligentnějších systémů. Tuto inteligenci tvoří právě sdílení dat, či možnost získat data třetích stran pomocí připojení k internetu. Třetí hranou v pomyslném trojúhelníku je „Systems“. Ten je tvořen právě komunikací mezi jednotlivými články [1], [6].



Obr. 2.1 *CPS – 3C architektura*

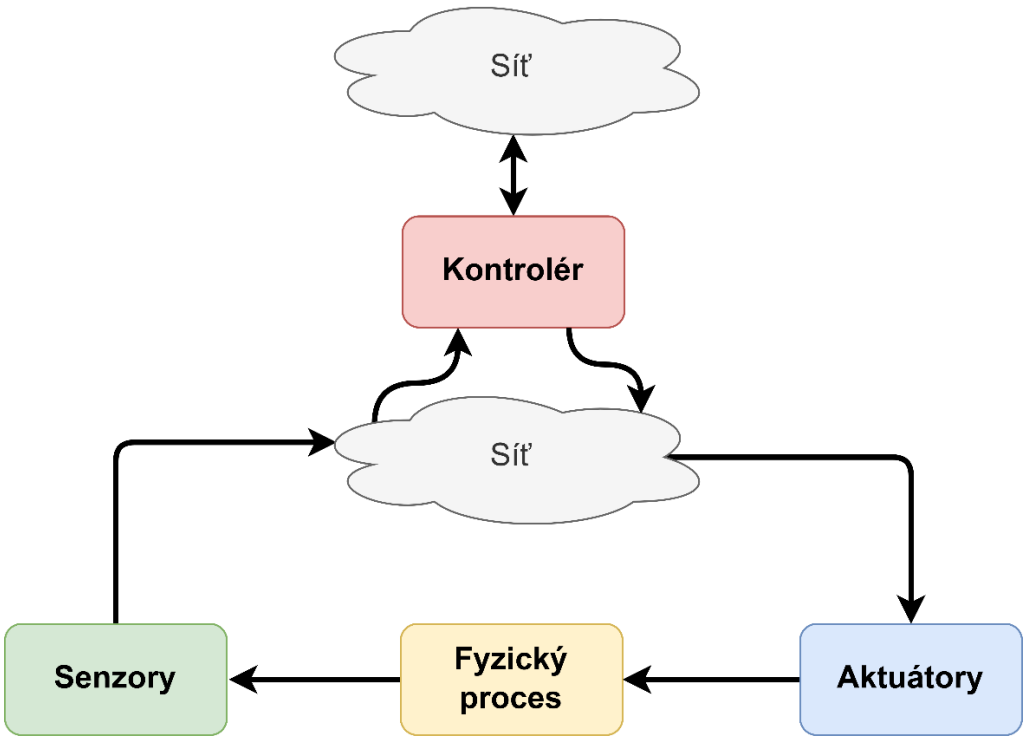
Konkrétně si lze obecnou architekturu CPS znázornit schématem na *obr. 2.2*. Zde vidíme, že obecnou architekturu CPS lze popsat několika jednoduchými prvky. Celkově se dá CPS považovat za zpětnovazební systém. Jádrem systému je kontrolér – řídící počítač, zodpovědný za zpracování senzorických dat a za ovládání akčních členů. Obecně může být kontrolér dostupný senzorům a akčním členům skrze síť, kde sítí si lze představit cokoliv od IP sítí po síť zařízení

komunikujících po sériové sběrnici. Důležitou roli zde také hraje fakt, že kontrolér může být obecně připojen do další sítě, což umožňuje již zmíněnou komunikaci zařízení, například i s dalšími kontroléry. Kontroléry také mohou být distribuovaným systémem.

Dalším prvkem obecné architektury CPS jsou akční členy. Akčním členem rozumíme zařízení, které převádí informaci, či nějaký kontrolní signál na mechanickou akci ve fyzickém světě. Příkladem akčních členů mohou být různé druhy elektrických motorů. Akční členy dostávají v CPS příkazy od kontrolérů, vykonávají je a tím ovlivňují určitý fyzický proces.

Fyzický proces je dalším dílem architektury, nicméně jedná se spíše o její pasivní součást. Fyzickým procesem je myšlen určitý proces ve fyzickém světě, který je akčními členy přímo ovlivňován. Pokud je například akčním členem elektrický motor pohánějící elektromobil, pak fyzickým procesem je v tomto kontextu otáčení kol mající za důsledek pohyb auta samotného. Je důležité si uvědomit, že na fyzický systém mohou mít vliv i jiné fyzické procesy mimo zpětnovazební smyčku CPS. Pokud se vrátím k příkladu elektromobilu, je možné, že jej někdo nebo něco bude tlačit, čímž opět ovlivňuje fyzický proces otáčení kol. Tím se dostáváme k nutnosti zpětné vazby v podobě senzorů.

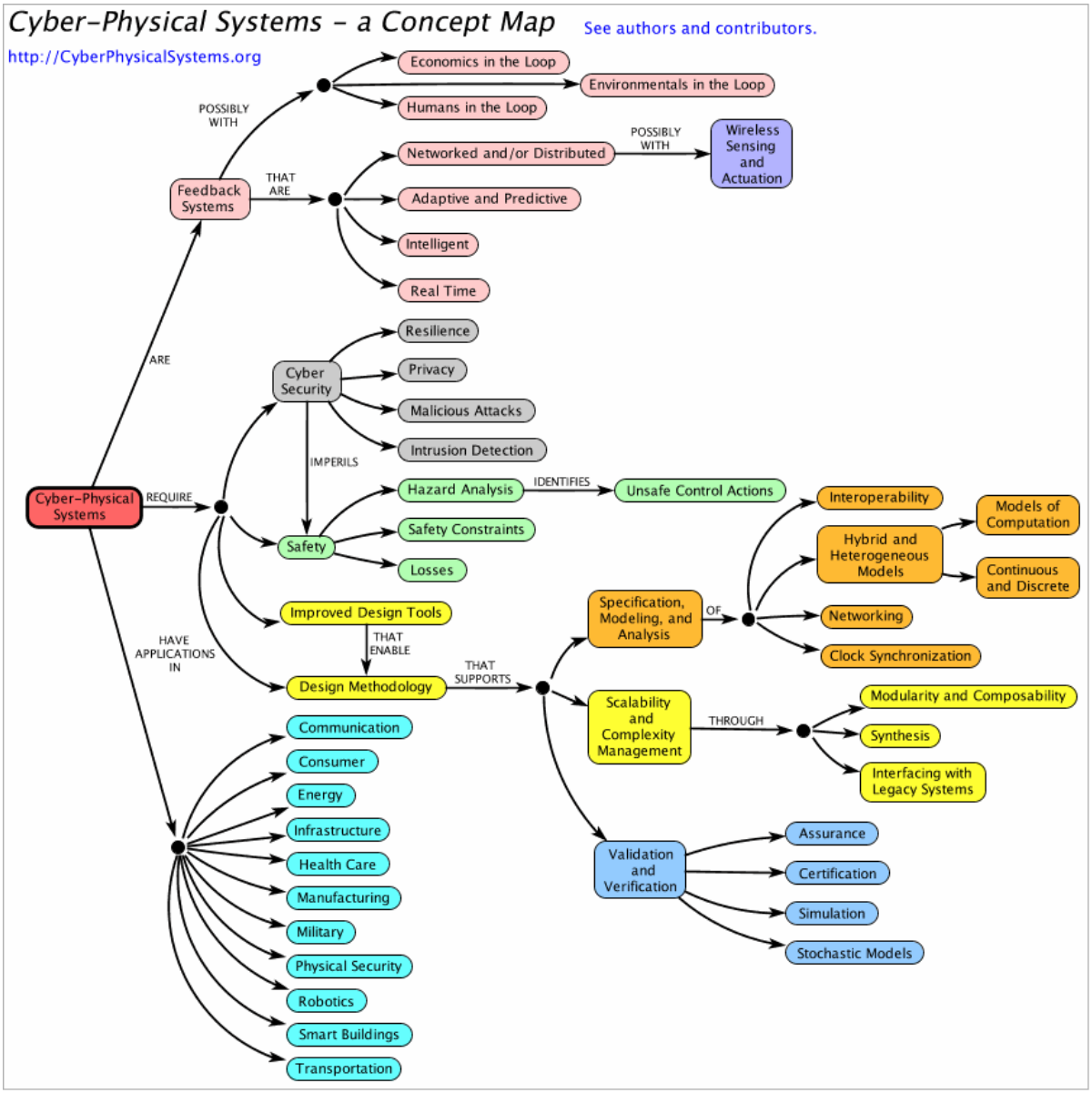
Senzory jsou posledním dílem v obecné architektuře CPS. Senzor je zařízení, které je schopné převádět fyzikální veličinu na informaci. Jsou tedy určené k získávání informace o vlastnostech fyzického procesu. Tato informace se v digitální podobě posílá na kontrolér, který informaci zpracuje. Zpracováním je možné rozumět analýzu senzorických dat, logování, či odeslání dat jinému zařízení prostřednictvím síťové konektivity. Důležité je, že jsou to právě senzorická data, na základě jejichž interpretace je kontrolér schopen činit rozhodnutí ohledně ovládání akčních členů.



Obr. 2.2 *Obecná CPS architektura*

Zajímavý a poměrně kompletní popis CPS nabízí také myšlenková mapa, která je vyobrazena na *obr. 2.3* a převzatá ze [7]. Autoři této myšlenkové mapy uvádějí, že se jedná o zpětnovazební systémy, které disponují síťovou konektivitou, jsou adaptivní a prediktivní, inteligentní a operují v reálném čase.

V myšlenkové mapě také autoři uvádějí, že CPS vyžadují kyberbezpečnost. Kyberbezpečnost (security) má podle nich také přímý vliv na fyzickou bezpečnost (safety) daného CPS. Tento poznatek je velice klíčový, protože jak již bylo zmíněno v podkapitole 1.1 – fyzickou bezpečnost a kybernetickou bezpečnost nelze v CPS oddělit a je nutné k tomuto problému přistupovat komplexně. Kyberfyzikální bezpečnost bude tématem podkapitoly 2.3.



Obr. 2.3 *Myšlenková mapa konceptu CPS* [7]

2.2 Technologie v CPS

Vzhledem k povaze CPS lze rozdělit použité technologie do několika skupin, které korespondují s 3C architekturou z *obr.* 2.1. Souhrn těchto technologií použitých v CPS lze najít v [8]. V této části jsou uvedeny technologie, které lze přiřadit k „cyber“ části pomyslného trojúhelníka.

2.2.1 Softwarové technologie

Pod pojmem „softwarové technologie“ si je možné představit softwarové vybavení komponent v CPS, jakožto i nástroje, kterými lze tento software vytvářet. Jelikož jsou v CPS často používány procesory od malých mikrokontrolerů, až po ty, které lze nazvat dnešním „high-end“, jsou použity i různé softwarové technologie, korespondující s limitacemi hardwaru. Ty nejjednodušší zařízení obsluhující několik senzorů, či aktuátorů, dokáže spolehlivě obsloužit malý 8-bit mikrokontroler. Tato malá zařízení obsluhuje pouze firmware, který je nahrán do paměti mikrokontroleru. Zpravidla se tento firmware programuje v nízkoúrovňových programovacích jazycích, jako jsou C a C++. Jelikož technický vývoj jde neustále kupředu, mikrokontrolery mají dnes více paměti a větší výpočetní výkon, než tomu bylo v minulosti. I z toho důvodu se začínají používat i více „high-level“ programovací jazyky, jako je např. MicroPython. Jejich nevýhodou je pomalejší běh programu a menší paměťová efektivita [9]. Na výkonnějších a složitějších systémech již najdeme využití operačních systémů. Nejčastěji se setkáme s některou distribucí Linuxu. Důvodem pro využití Linuxových systémů je spolehlivost, nenáročnost, efektivita a podpora ze strany open-source komunity.

V celém komplexním systému je nutné nasazení celé řady softwarového vybavení. Zásadní je monitoring dat a funkčnosti celého systému. Senzorická data a události v systému je nutné ukládat. Zde nachází uplatnění databázové systémy. Tradiční relační databáze využívající jazyk SQL se v poslední době setkávají s konkurencí v podobě NoSQL databází. Známými zástupci ze strany SQL databází jsou např. MySQL, či PostgreSQL. Často využívaným databázovým systémem v CPS a IoT je pak InfluxDB, který se řadí mezi NoSQL databáze a je speciálně navržen pro data časových řad. Dalším příkladem NoSQL databáze může být MongoDB. Nelze jednoznačně určit, který druh databází je pro využití v CPS výhodnější. Vždy záleží na konkrétní aplikaci. Tomuto tématu se věnovala i studie [10], která přinesla nekonkluzivní výsledky.

Vizualizace nasbíraných dat je další neméně důležitou částí systému. Vhodná vizualizace dat umožňuje získat přehledné informace o sledovaném systému. Součástí vizualizace mohou být i upozornění na negativní vývoj v průběhu dat, značící potenciální problém ve sledovaném systému. Často používanými platformami pro datovou vizualizaci jsou platformy Grafana a Kibana. Umožňují přehlednou vizualizaci dat uložených v databázových systémech. Ukázka tzv. „dashboardu“ je na *obr. 2.4.*

Obr. 2.4 *Ukázka vizualizace dat pomocí platformy Grafana* [11]

2.2.2 Síťové a komunikační technologie

Komunikační technologie jsou jednou z nejdůležitějších částí CPS. Pro maximální přínos CPS je komunikace jednotlivých zařízení podmínkou. V CPS jsou často využívány jak drátové, tak bezdrátové komunikační technologie, přičemž o nasazení dané technologie rozhodují potřeby konkrétní aplikace. Pokud je to možné, komunikační technologie využívající metalická vedení jsou preferovány. Nabízejí vyšší spolehlivost, odolnost vůči rušení a bezpečnost. Velmi ověřenou technologií je sériová sběrnice RS-485. Tato sběrnice umožňuje komunikaci po metalickém vedení až na vzdálenost 1200 m, kde při těchto vzdálenostech dosahuje rychlosti až 100 kbps. Při kratších vzdálenostech se rychlost pohybuje okolo 10 Mbps [12]. Vzhledem k primárnímu nasazení v senzorických sítích se jedná o dostatečnou přenosovou rychlost. Dalšími komunikačními technologiemi, používanými zejména v průmyslu, jsou např. sběrnice CAN, či 1-Wire. Poslední dobou také můžeme i v CPS sítích více sledovat nasazení technologie Ethernet a přechod na komunikaci založené na TCP/IP. Klasické síťové technologie jsou často využity zejména pro komunikaci složitějších zařízení s operačními systémy.

Bezdrátové technologie nacházejí využití v situacích, kdy je nasazení drátových technologií nepraktické, či ekonomicky neefektivní. Tato zařízení jsou často napájená baterií, proto je energetická náročnost technologie klíčovou vlastností. Technologie Wi-Fi založená na standardech 802.11 není pro tyto účely ideální zejména kvůli vysoké spotřebě energie a až zbytečně vysokou přenosovou rychlostí (pro většinu aplikací ve WSN). Standard 802.15.4, který je základem pro technologie jako Zigbee, či 6LoWPAN, je daleko vhodnější volbou pro tyto aplikace. Dalšími bezdrátovými technologiemi nacházející využití v aplikacích WSN jsou Bluetooth, BLE, či Z-Wave [13].

2.3 Kyberfyzikální bezpečnost

Základní myšlenka kyberfyzikální bezpečnosti, neboli zajištění bezpečnosti kyberfyzikálních systémů, byla představena v úvodu v kapitole 1. Tato myšlenka je dále rozvedena v několika dalších odstavcích. Jak již bylo řečeno v bezpečnosti CPS je nutné nastavit takový soubor opatření, který svazuje kybernetickou a fyzickou bezpečnost systému. Klasická kybernetická bezpečnost zahrnuje bezpečnostní mechanismy jako jsou např. šifrování, autentizace, nastavení firewallů, či využití IDS/IPS (Intrusion Detection/Prevention System). Lze sem ale zahrnout i bezpečný návrh sítě jako takové, vývoj bezpečného softwarového vybavení pro komponenty systému a v neposlední řadě i penetrační testování výsledného systému, na jehož základě bude jeho bezpečnost posílena odstraněním zjištěných zranitelností. Tyto metody mají nepochybně svůj prostor i v bezpečnosti CPS, ale díky unikátním vlastnostem CPS jsou sami o sobě nedostatečné [14]. Využití standardních metod a technologií pro zabezpečení komunikace a zabezpečení systémů je vhodné z důvodu, že se již jedná o ověřené a široce používané technologie.

Metody kybernetické bezpečnosti ale nejsou stoprocentní. Zajištění kybernetické bezpečnosti je kontinuální proces a žádný systém není absolutně bezpečný. Útočníci mohou i v dobře zabezpečeném systému nalézt zranitelnosti a zneužít je. Často je ale problémem i strana provozovatele systému. I v případě CPS se bavíme o systémech, které jsou například průmyslově nasazeny, a které spravuje často jeden administrátor, či úzká skupina osob. Ne vždy mají tyto osoby dostatečné zkušenosti, aby byly schopny zhodnotit bezpečnostní riziko a implementovat vhodná bezpečnostní opatření, či jsou zde jiné překážky (např. finanční, pohodlnost, atd.), které dosažení tohoto cíle brání. Zaměstnanci jsou často také pod tlakem a nejsou si plně vědomi rizik při kliknutí na různé odkazy v emailech, což může vést k zavlečení malwaru do firemní sítě, jejíž součástí může být i CPS. Výsledkem jsou neideálně zabezpečené systémy, které jsou snadnějším cílem pro útočníky [15]. Jednou z nejčastějších zranitelností, se kterými se penetrační testeři setkávají, jsou slabá hesla, či dokonce defaultní hesla z výroby. Dalším velkým problémem je neaktualizování softwaru a používání zastaralých verzí, které často obsahují méně, či více závažné zranitelnosti [16]. Tyto problémy jsou přitom příklady těch, které lze velice jednoduše odstranit. Bohužel zajištění bezpečnosti bývá kompromisem mezi bezpečností a pohodlím, kde v nezanedbatelném množství případů vyhrává právě pohodlí.

V případě fyzického zabezpečení systémů se jedná spíše o problematiku inženýrského návrhu. Selhání systému se nedá vyloučit, důležité je zajistit selhání bezpečné. Pokud budeme jako příklad CPS uvažovat těžké průmyslové stroje, lze si představit různá nebezpečí, která u takových strojů existují. Návrh stroje musí počítat s různými poruchami a hraničními stavy, do kterých se stroj může dostat a musí být zajištěné pojistky pro bezpečné odstavení stroje v případě poruchy, kdy není stroj schopen dále pokračovat ve funkčním stavu [17]. Díky monitorování stroje pomocí senzorů ale nemusí k poruše vůbec dojít. Opotřebení komponent se projevuje změnou některých veličin. Tyto veličiny mohou být monitorovány pomocí senzorů a můžeme tak získávat data o stavu monitorovaného objektu. Poté je možné díky prediktivním algoritmům odhalit opotřebení dříve, než dojde k poruše. Tento přístup je efektivnější a levnější než potenciálně vážné následky poruchy dané komponenty [18]. Stejně jako popsané metody kyberbezpečnosti jsou i tyto metody fyzického zabezpečení již zaběhnutou praxí.

Kyberfyzikální bezpečnost ale otevírá nové problémy. Zatímco kybernetické systémy jsou spíše abstraktním pojmem, CPS interagují s reálným světem a s lidmi, kteří v něm žijí. Nebezpečí se tedy přesouvá čistě z abstraktního prostoru do prostoru reálného světa. Pokud selžou systémy kybernetické bezpečnosti a útočník získá přístup do systému, již nehrozí pouze ztráta citlivých dat a vyřazení systémů z provozu, ale hrozí i reálné nebezpečí v podobě ohrožení zdraví osob a majetku. Kyberfyzikální bezpečnost by měla odpovídat na takové otázky, jako jsou například:

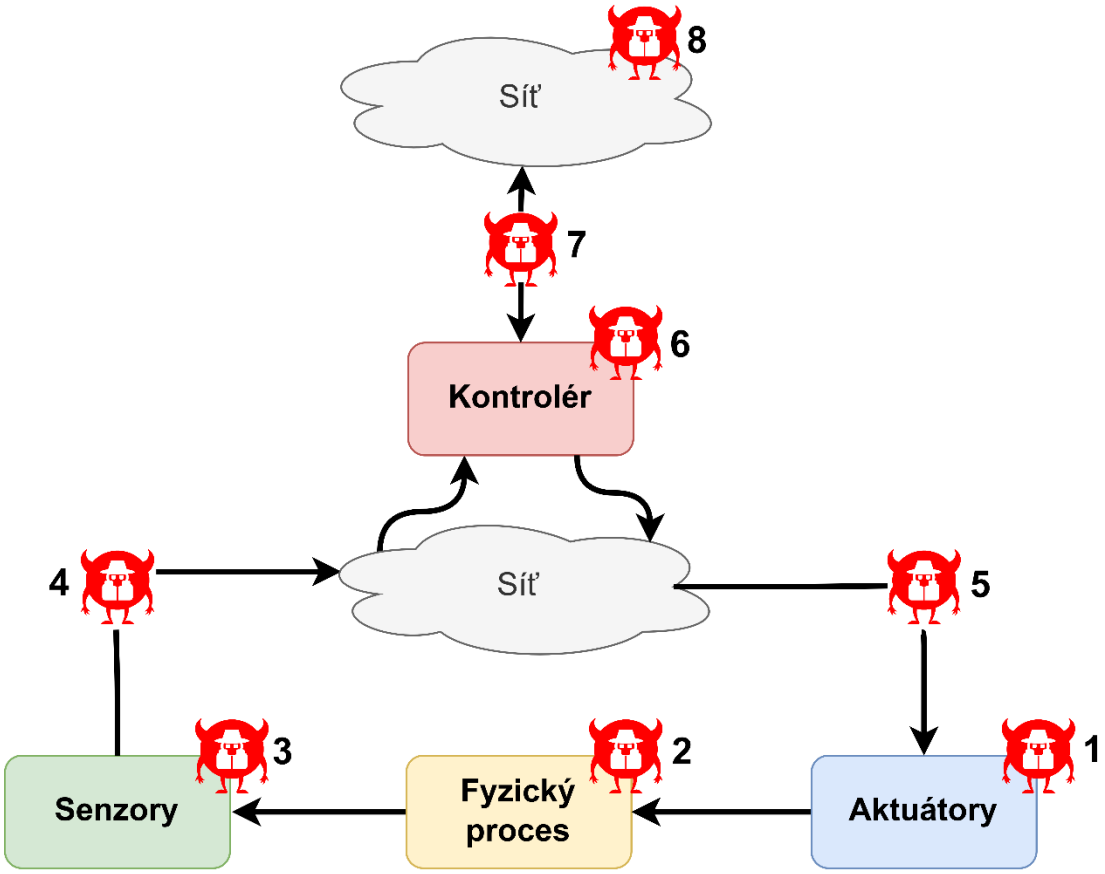
- Jak útočníkovi znemožnit přístup ke kontrole CPS?
- Pokud již útočník má přístup do systému, jak lze zabránit tomu, aby napáchal škody?
- Jaké útoky může útočník provést na vrstvě senzorů a aktuátorů? Jak tyto útoky detekovat a zabránit jim?
- Jaké jsou následky různých útoků na dané CPS?

Odpovědi na tyto otázky v plné šíři neposkytuje ani kybernetická bezpečnost, ani fyzické zabezpečení. Oba přístupy jsou součástí kyberfyzikální bezpečnosti, ale na to, abychom takové systémy dokázaly skutečně zabezpečit, je potřeba přemýšlet nad správnými otázkami již během návrhu. Kyberfyzikální bezpečnost totiž nelze zcela vyřešit nasazením kryptografických protokolů, ani monitorováním systému pomocí senzorů, která nám nemusí dávat pravdivá data. Zabezpečení CPS vyžaduje správný multioborový inženýrský návrh, zohledňující různé druhy hrozeb. Pro takový návrh je důležité spojit znalosti ze síťových technologií, kryptografie, síťové bezpečnosti, penetračního testování, technologií bezdrátové komunikace, mikrokontrolerů, senzorů, akčních členů a mnoho dalších.

2.3.1 Obecný model bezpečnosti CPS

Aby bylo možné plně porozumět rizikům v kyberfyzikálních systémech, je dobré uvažovat model architektury, který byl uveden na *obr. 2.2*. Téma kyberfyzikální bezpečnosti je obsáhle zpracováno v [19] a myšlenka modelu bezpečnosti CPS je převzatá z tohoto zdroje, viz *obr. 2.5*.

V modelu bezpečnosti CPS se nejprve identifikují místa, ve kterých může dojít k útokům. Útokem lze chápat úmyslné, či neúmyslné ovlivnění systému s vlivem na funkčnost systému. Tyto útoky jsou obecného charakteru. Konkrétním hrozbám, se kterými se lze setkat v CPS, se věnuje podkapitola 2.3.2.



Obr. 2.5 *Model bezpečnosti CPS s vyznačením možností útoků*

Útočník 1

Tento útočník je schopen napadnout aktuátor a kontrolovat jej nezávisle na příkazech z hlavního kontroléru. Takový útok lze realizovat například nahráním škodlivého firmwaru do mikrokontroleru ovládající daný aktuátor.

Útočník 2

Zde je symbolizován útočník, který útočí přímo na fyzický proces, který je kontrolován aktuátory a jeho stav je monitorován senzory. Může se také jednat o fyzické poškození tohoto procesu. Důsledkem takového útoku může být manipulace se systémem, či odstavení systému z funkce v případě fyzického poškození.

Útočník 3

Tento útočník představuje scénář, kdy data poskytována senzorem již nejsou důvěryhodná. Útočník může nad senzorem převzít kontrolu a manipulovat se signály a daty tak, aby data poskytovaná senzory neodpovídala realitě. Jelikož je CPS obecně zpětnovazební systém, kde je systém řízen na základě dat ze senzorů, jedná se o poměrně velký problém, protože může dojít k nepřímé kontrole aktuátorů právě skrze manipulaci senzorické zpětné vazby.

Útočník 4

V tomto případě byl útočník schopen dostat se mezi komunikaci senzoru a kontroléru. Pokud není zajištěna integrita senzorických dat, může dojít k jejich změně, což má efektivně stejné důsledky jako případ útočníka 3. Útočník ale může být také schopen signály, či zprávy ze senzorů opozdit, či zablokovat. To způsobí ztrátu informace o systému a kontrolér již není schopen adekvátně reagovat. Tento útok se řadí do kategorie „Denial of Service“ (DoS) a může být způsoben zastaralými senzorickými daty [20].

Útočník 5

Zde je reprezentován útočník, který je schopen kontrolovat tok dat z kontroléru do aktuátorů. Tím může buď příkazy modifikovat, či zpožďovat nebo dokonce blokovat. Důsledkem je buď kontrola nad aktuátory, či v druhém případě útok typu DoS, který zamezí správné funkčnosti aktuátorů.

Útočník 6

Tento útočník modeluje situaci, kdy byl kontrolér napaden. Útočníkovi se mohlo podařit získat přístup ke kontroléru a získat nad ním kontrolu. V tomto případě by útočník získal kontrolu nad celým systémem a mohl by posílat libovolné příkazy aktuátorům. Kontrolér může být také napaden takovým způsobem, že bude odstaven z provozu (DoS), čímž by byl z provozu odstaven celý systém.

Útočník 7

Komunikace mezi sítí a kontrolérem představuje zejména komunikaci mezi kontrolérem a řídícím střediskem. Může ale reprezentovat i komunikaci mezi kontrolérem a libovolným místem v síti, jelikož lze předpokládat, že kontrolér bude komunikovat i s jinými koncovými body, nežli pouze s řídícím střediskem. Právě do této komunikace se dostal útočník v tomto scénáři. Záleží na implementovaných bezpečnostních opatřeních, nicméně útočník by v tomto případě mohl být schopen blokovat síťový provoz, což by mohlo omezit funkčnost systému. Útočník by také teoreticky mohl být schopen síťový provoz modifikovat.

Útočník 8

Útočník byl schopen převzít kontrolu nad některým z koncových bodů, se kterými přes síťovou komunikaci interaguje kontrolér. Důsledkem tohoto útoku by mohla být možnost útočníka provést další útoky na kontrolér a získat tak kontrolu nad systémem. V případě kompromitace řídícího střediska by útočník přímo mohl zasílat škodlivé příkazy. Zde ale záleží na konkrétní implementaci celého systému.

2.3.2 Hrozby v CPS

Autoři [21] přišli se souhrnem hrozeb v CPS v podobě stromového diagramu, viz *obr. 2.6*. Stromový diagram si můžeme rozložit do následujících částí:

- Kořen
  - Selhání CPS → důsledek uskutečnění hrozeb
- Kmen
  - Zpětná vazba
- Větve
  - Aktuátory
  - Senzory
  - Síťová komunikace
  - Software

Obr. 2.6 *Stromový diagram hrozeb v CPS* [21]

Senzory

Útoky na senzory jsou jednou z hrozeb, které se vyskytují v kyberfyzikální bezpečnosti. Hrozby vyplývají i z přirozeného prostředí. Například vlivem okolního prostředí může být senzor poškozen, či zničen. Autoři v [21] dále uvádějí hrozby jako například manipulace s hardwarem, ztráta napájení, poškození hardwaru, či ovlivňování senzorických dat. Do poslední skupiny patří mimo jiné útoky GPS spoofing, injektování falešných radarových signálů, či oslňování kamer pomocí světla. V článku [22] autoři provedli experiment, ve kterém úspěšně dokázali kontrolovat hlasové asistenty pomocí vysílání speciálně upravených laserových signálů na MEMS mikrofony daného hlasového asistenta (Google Home, Alexa, atd.). Hlasový asistent poté interpretoval laserové signály přijímané MEMS mikrofony jako hlasové příkazy uživatele. V roce 2019 se GPS spoofing ukázal jako problém pro automobilku Tesla, kde bylo zjištěno, že některé její modely jsou náchylné na tento útok [23]. Podobné útoky se jeví jako vážné hrozby pro nasazení autonomních systémů.

Síťová komunikace

V této větvi identifikovali autoři [21] hrozby, se kterými se lze setkat v klasické síťové bezpečnosti. Tyto hrozby jsou většinou dobře známé, tedy jsou známé i způsoby, jakými se před nimi lze efektivně chránit. Jedná se o hrozby jako jsou odposlouchávání komunikace, „replay“ útoky, změna dat, různé druhy spoofingu, atd. Tyto útoky jsou řešeny pomocí metod moderní kryptografie a vhodné konfigurace síťových zařízení. I zde je možné setkat se s hrozbou, která není přímo útokem, jelikož může docházet k náhodným chybám.

Software

Útoky na výpočetní složku systému jsou další z těch, se kterými se lze setkat již v klasické kyberbezpečnosti. Patří sem například zneužití zranitelností pomocí exploitů a malware. Opět je možné se setkat i s hrozbou v podobě „přirozené“ chyby v softwaru, která zapříčiní nefunkčnost systému.

Aktuátory

Tato část je, stejně jako část senzorická, specifikem pro CPS. Autoři v [21] uvádějí útoky jako modifikace softwaru, modifikace hardwaru, či ztráta napájení. Dalším možným útokem na aktuátory je modifikace vstupních příkazů a signálů. Jedním z možných způsobů je modifikace příkazů na úrovni dat. V tomto případě budou modifikována samotná data v příkazu od kontroléru. Aktuátor tak pouze vykoná to, co je obsahem zfalšovaných příkazů. Lze ale jít o úroveň níže. V [24] se autorům podařilo modifikovat PWM signál pro servo motor pomocí elektromagnetického pole toroidní cívky. Vodiče vedoucí z mikrokontroleru do servo motoru (nesoucí PWM signál) byly omotány okolo této toroidní cívky. Díky této modifikaci PWM signálu, dokázali autoři značně ovlivnit chování servo motoru. Tyto útoky je možné nazývat „False Actuation Injection“.

Zpětná vazba

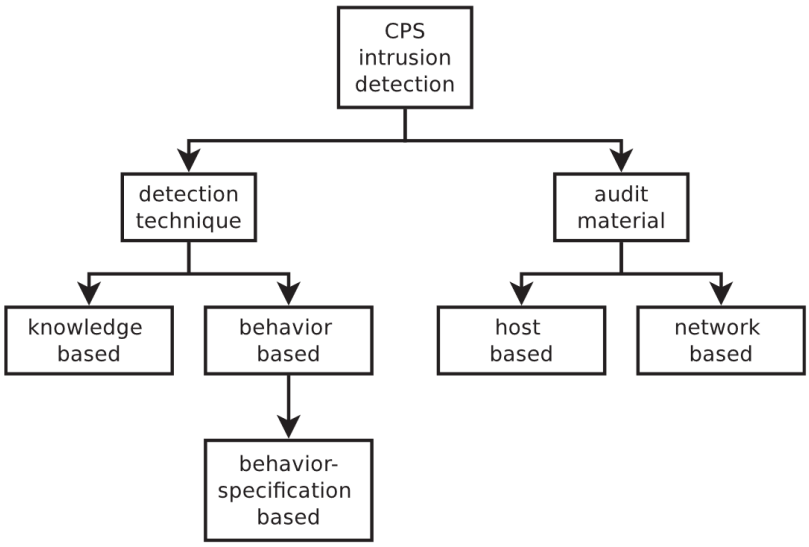
Modifikace zpětné vazby je další z útoků uvedený autory [21]. Pokud se útočníkovi podaří ovlivnit tok dat ze senzorů do kontroléru, či z kontroléru do aktuátorů, dosáhne tím stejných výsledků jako v případě ovlivnění samotných senzorů nebo aktuátorů.

2.3.3 Metody detekce a mitigace hrozeb v CPS

Ke kybernetickým hrozbám, které byly popsány v podkapitole 2.3.2 lze přistoupit známými metodami kybernetické bezpečnosti. Tyto metody již byly dříve popsány v úvodu podkapitoly 2.3.

Větší otázkou v rámci výzkumu kyberfyzikální bezpečnosti zůstávají bezpečnostní opatření proti hrozbám specifickým pro svět CPS. Hlavní metody detekce a mitigace hrozeb v CPS jsou převzaty ze souhrnu v [19].

Detekce útoků je prvním krokem pro zabránění útoku. Aby bylo možné proti útočníkovi zasáhnout, je nutné o něm nejprve vědět. Na pomezí kybernetických hrozeb stojí útoky, které lze zaznamenat pomocí IDS/IPS. Metod pro IDS v CPS je více. V článku [25] je zpracován přehled metod pro IDS v kyberfyzikálních systémech. Na *obr. 2.7* je uvedeno rozdělení technik pro IDS.



***Obr. 2.7** Techniky IDS v kyberfyzikálních systémech* [25]

Detekce anomálií je dalším způsobem jak odhalit útok v CPS. Nejjednodušším příkladem může být prahování – přesáhnutím přednastavené limitní hodnoty se vygeneruje upozornění. Existují ale i složitější algoritmy pro detekci anomálií, které jsou založené na strojovém učení. Další metoda detekce anomálií v CPS je založená na nesrovnalostech mezi senzorickými daty a fyzikálními zákony. Pomocí fyzikálních zákonů je možné vytvořit predikci, kterou by se následně měla řídit i senzorická data. Pokud je značná odchylka reality od toho, co je dle fyzikálních teorií možné, jedná se o problém. V těchto případech se jednalo o metody pasivní detekce, nicméně existují i metody aktivní detekce. Při aktivní detekci se sleduje, jak kyberfyzikální systém reaguje na testovací příkazy. Pokud kontrolér pošle testovací příkaz aktuátorům, měla by se změna jeho

stavu adekvátně promítnout ve změně senzorických dat. Pokud se tak nestane, je možné, že jsou senzorická data podvrhnutá nebo že aktuátor nedostal příkaz. [19]

Metod pro mitigaci útoku v CPS je mnoho. Vhodné je jednotlivé metody správně kombinovat pro dosažení maximální úrovně zabezpečení. V případě, že systém využívá senzorickou fúzi, je možné využít korelaci senzorických dat jako parametr pro vyhodnocení, zda některý ze senzorů poskytuje nesprávná data. Další metodou může být omezení možností aktuátorů. Pokud útočník uspěje v napadnutí systému a povede se mu převzít kontrolu, je důležité omezit jeho možnosti v tom, co může vykonat. I přes to, že má útočník plnou kontrolu nad zařízením, není schopen ho uvést do nebezpečného stavu. Podobný přístup má metoda s hlavní myšlenkou tzv. „reference monitor“. Před tím, než kontrolér odešle příkaz, využije „reference monitor“ pro kontrolu, zda při vykonání tohoto příkazu nedojde k uvedení systému do nebezpečného stavu. [19]

2.3.4 Stuxnet – příklad útoku na CPS

Jedním z prvních a nejznámějších rozsáhlých útoků na kyberfyzikální systémy byl Stuxnet. V lednu 2010 si inspektoři „International Atomic Energy Agency“ (IAEA) všimli velmi zvýšené poruchovosti centrifug v závodu na obohacování uranu v Íránském Natanz. V té době bylo velkou záhadou proč k tomuto jevu docházelo [26].

Stuxnet byl velice inovativní a sofistikovaný počítačový červ. Počítačový červ je škodlivý program, který je schopný svojí replikace a šíření na další systémy. Stuxnet se šířil pomocí USB disků a využíval několik „0-day“ exploitů, o kterých neměli vývojáři zranitelných softwarů ponětí [26], [27]. Stuxnet cílil na počítače s operačním systémem Windows, na kterých byl nainstalován software pro kontrolu Siemens SCADA systémů – konkrétně šlo o kontrolní software WinCC/PC S7 SCADA. Stuxnet zachytával komunikaci mezi počítačem s OS Windows a připojenými PLC zařízeními, které spolu byly propojeny datovým kabelem. Technicky se tak jednalo o útok typu „man-in-the-middle“ [28]. Stuxnet útočil na ovládací motory, jejichž rychlost otáček se pohybovala mezi 807 a 1210 Hz – právě v těchto otáčkách operují ony napadané plynové centrifugy a pumpy [26]. Stuxnet poté do PLC zařízení nainstaloval malware, který za určitých okolností měnil rychlost otáčení motorů centrifug na příliš vysoké, či naopak příliš nízké rychlosti. Cílem bylo pomalu zničit tyto centrifugy. Stuxnet také instaloval rootkit, který dokázal falšovat senzorická data, tudíž se vše jevilo v normě [28].

Závodu na obohacování uranu Natanz nepomohl ani fakt, že se jednalo o síť s tzv. vzduchovou mezerou – systém nebyl nijak připojen k internetu. Jelikož byl ale Stuxnet přenášen pomocí USB disků, pravděpodobně jej do systému zavlekl nic netušící zaměstnanec. Spekulovalo se také o původu Stuxnetu. Vzhledem k cíli, kterým byl de facto Íránský jaderný program, se spekulovalo, že zdrojem by mohli být armádní složky USA a Izraele. Jednalo by se tak o kybernetickou zbraň [26].

Stuxnet byl mnohem komplikovanější. Článek [28] popisuje jeho komplexnost ve větším detailu.

Důležité ponaučení, které jsme díky Stuxnetu získali je fakt, že výzkumu bezpečnosti kyberfyzikálních systémů je nutné se věnovat. Ukázal nám, že cílem se mohou stát i průmyslové továrny, jejichž vyřazení z provozu by mohlo být velkým problémem pro společnost. Tento fakt potvrzují i další události z poslední doby, kdy se útoky proti kyberfyzikálním systémům využívají jako prostředek „kyber války“. Příkladem může být útok na Ukrajinskou elektrickou rozvodnou síť z prosince 2015 [29].

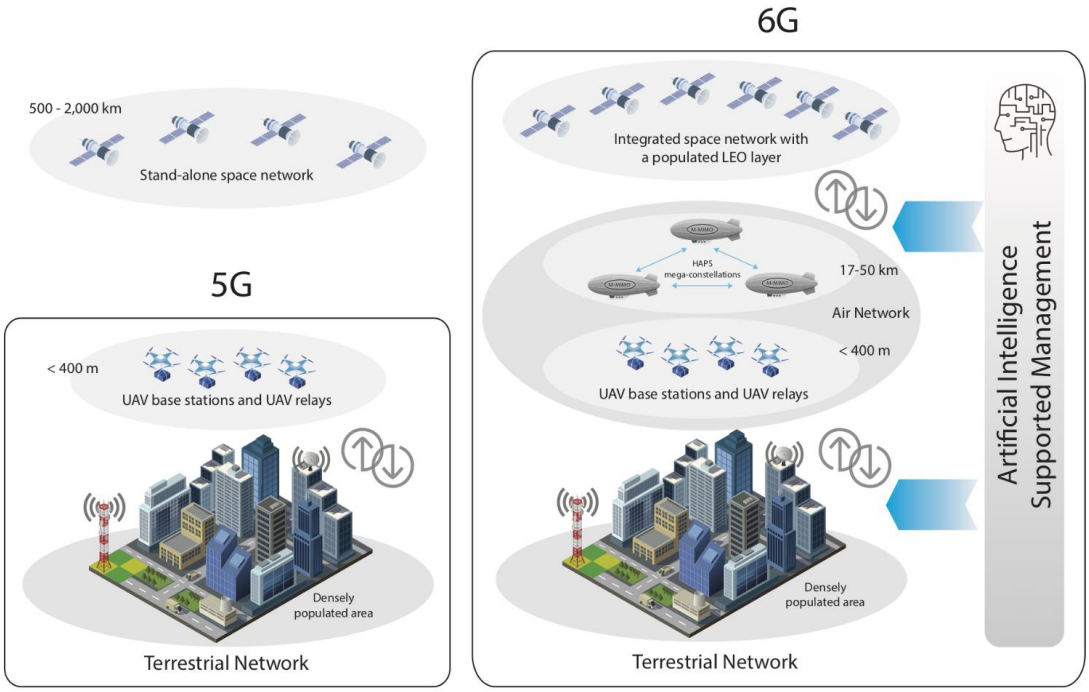
Kapitola 3
Pseudosatelity HAPS

Koncept HAPS byl již představen v úvodní kapitole. Jak již bylo řečeno, HAPS má široké pole využitelnosti. V této kapitole je detailněji rozebráno využití HAPSu v sítích 6G a také jak celý systém vypadá a z čeho se skládá.

3.1 Využití konceptu HAPS v sítích 6G

Využití nachází HAPS v sítích 6G zejména jako prostředek pro poskytování pokrytí mobilním internetem. HAPS by mohl nabízet relativně levné řešení pro poskytnutí pokrytí v odlehlých oblastech, ve kterých je ekonomicky neefektivní stavět síťovou infrastrukturu. Objevují se také názory, že HAPS by mohl být vhodný i pro nasazení v hustých městských zástavbách jako zajímavá alternativa k přetížené síťové infrastruktuře na zemi [30].

Na *obr. 3.1* můžeme vidět srovnání sítí 5G s budoucími sítěmi 6G. V sítích 5G jsou provozovány klasické terestriální sítě s využitím UAV základnových stanic. Na nízké oběžné dráze země jsou pak provozovány LEO (Low Earth Orbit) satelity. Dle autorů [32] *obr. 3.1* jsou sítě LEO satelitů provozovány odděleně od klasických terestriálních sítí. Velkým projektem, který má za cíl poskytovat rychlé satelitní internetové připojení s nízkou latencí je projekt Starlink společnosti SpaceX [31].



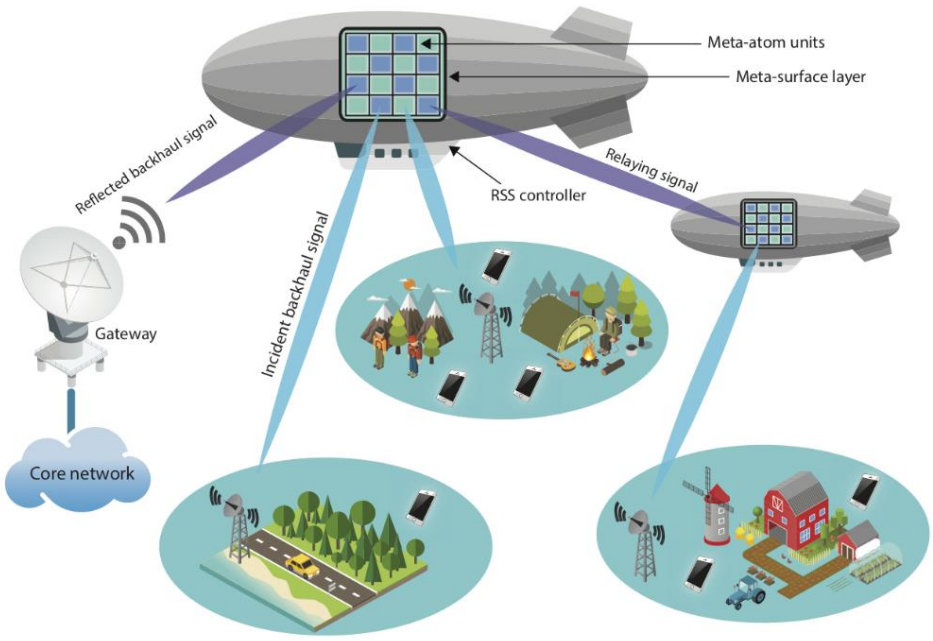
Obr. 3.1 *Vizualizace přechodu ze sítí 5G na sítě 6G* [32]

V sítích 6G by ale mohly pseudosatelity HAPS sloužit jako jakýsi mezičlánek mezi vrstvou terestriální sítě a sítě LEO satelitů. Další možnou aplikací poskytovanou v sítích 6G může být výpočetní server pro tzv. „offloading“, který by našel využití zejména v IoT. Takový princip již funguje v rámci „Multi-Access Edge Computing“ (MEC) [33]. Velmi podobné využití přináší use-case v podobě datového centra. Dále by HAPS mohl poskytovat páteřní spojení mezi základnovými stanicemi (viz *obr. 3.2*) nebo také pokrytí mobilním internetem při nečekaných událostech. V neposlední řadě je možné poskytovat podporu a řízení UAV a autonomním dopravním prostředkům. [32]

HAPS má oproti LEO satelitům několik výhod. Zřejmou výhodou je nižší provozní výška. Pseudosatelity HAPS operují ve stratosféře – ve výšce zhruba 20 km. Oproti tomu LEO satelity operují ve výškách od 350 km do 2000 km. To přináší HAPSu výhodu ve vyšším SNR ve směru downlink. Ve směru uplink je výhodou fakt, že díky nízkému parametru „Path Loss“ (PL) mohou být využity přímo „User Equipment“ (UE) bez potřeby pozemní stanice. Díky nižší výšce je také znatelně nižší latence, kvůli čemuž je HAPS vhodný i pro aplikace vyžadující velice nízkou latenci [32].

HAPS je také stacionárním pseudosatelitem, což znamená, že může operovat přesně v tom místě, ve kterém je potřeba. LEO satelity obíhají kolem Země, tudíž v určitých momentech prolétají například nad oceánem či neosídlenou oblastí – jejich využitelnost je pak tedy nižší [32].

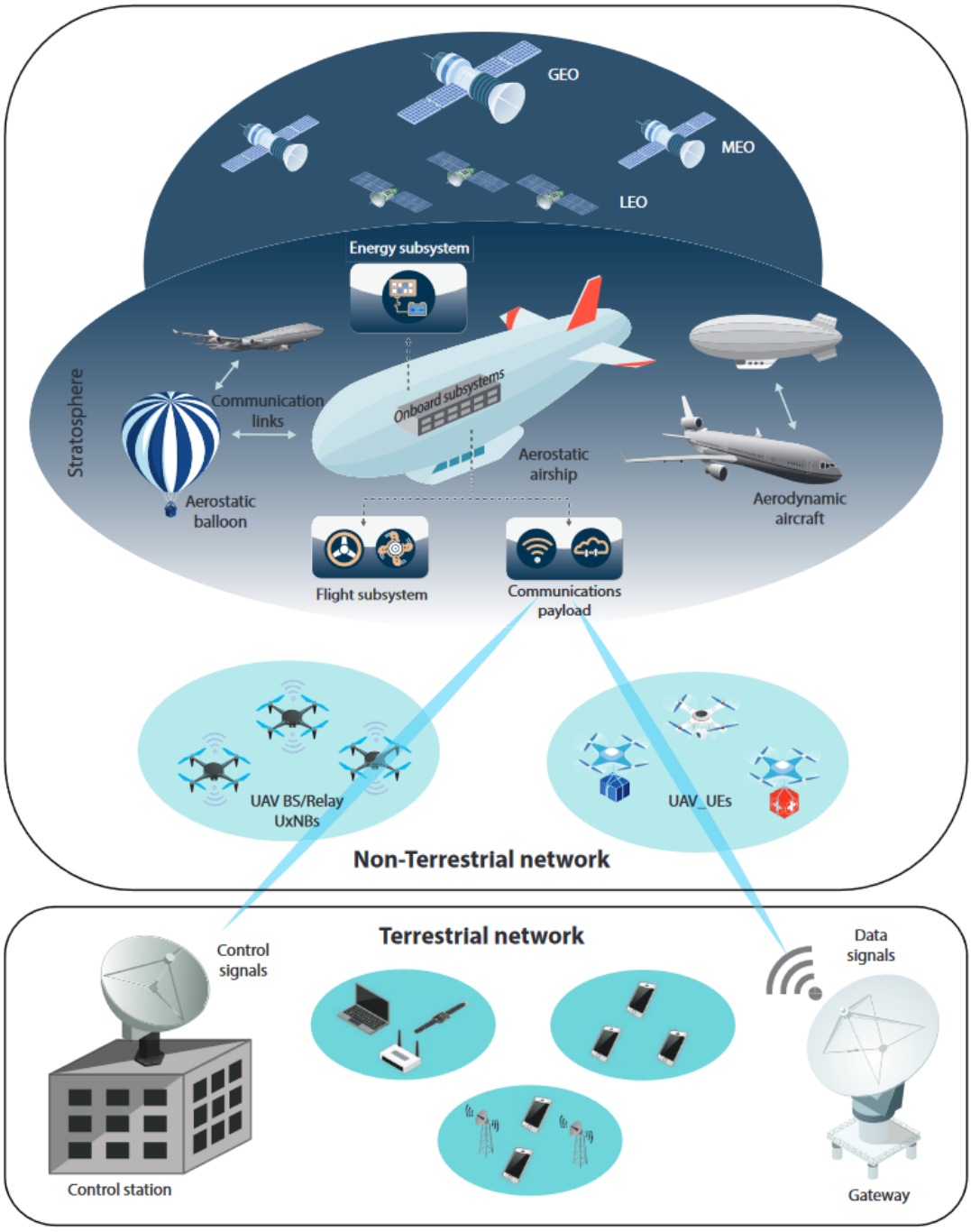
V neposlední řadě je zde výhoda ekonomická. HAPS je oproti LEO satelitům levnější, je snazší jej vynést do požadované polohy a tento proces je v případě HAPS méně rizikový. HAPS také dokáže pohodlně přistát zpět na Zemi, tudíž je na rozdíl od LEO satelitů jednoduše udržovatelný a použitelný vícekrát [32].



Obr. 3.2 *HAPS poskytující páteřní konektivitu pro základnové stanice* [32]

3.2 Komponenty systému HAPS,

Systém HAPS lze rozdělit na dvě části, viz *obr. 3.3*. První částí je pozemní kontrolní stanice, tvořící terestriální část sítě. Druhou je pak samotný HAPS ve stratosféře, tvořící neterestriální část sítě [32]. Obě tyto části jsou tvořeny dílčími komponenty. V dalších podkapitolách budou obě části detailněji popsány.



Obr. 3.3 *Systém HAPS* [32]

3.2.1 Pozemní kontrolní stanice

Úkolem pozemní kontrolní stanice je řízení HAPSu a dohled nad ním. Pozemní stanice musí sbírat data, monitorovat stav HAPSu a v případě potřeby zasáhnout. Pozemní kontrolní stanice se může skládat z následujících prvků [32]:

- Kontrolní stanice
  - Zařizuje samotnou komunikaci mezi Zemí a HAPSem
  - Zajišťuje správu HAPSu
  - Řídí vzlet a přistání HAPSu
- Komunikační brána
  - Připojuje HAPS k páteřní síti
- Síť v pozemní kontrolní stanici
  - Samotná kontrolní stanice je jeden fyzický stroj
  - Lze předpokládat, že zde bude více počítačů v rámci LAN sítě
  - Důležité zejména z bezpečnostních důvodů

V rámci terestriální části sítě fungují všechny uzly a uživatelé, kteří s HAPSem interagují. Toto zahrnuje pozemní základnové stanice, mobilní uživatele, či zařízení IoT [32].

3.2.2 HAPS

Samotný HAPS je složitější zařízení v porovnání s jednoduchým výčtem některých jeho částí. Pro představu je ale dobré uvést některé z jeho klíčových prvků [32]:

- Systém pro kontrolu letu
  - Má za úkol autonomně udržovat pozici HAPSu
  - Kontroluje přistání a vzlétnutí platformy
  - Pro řízení využívá data z několika různých senzorů
- Systém managementu energie
  - Energie je udržovaná v bateriích
  - Pro nabíjení jsou využity solární panely
  - Reguluje spotřebu elektrické energie
- Komunikační systém
  - Je zodpovědný za navázání a udržení komunikace s pozemní kontrolní stanicí a s dalšími uzly a uživateli v terestriální části sítě

Na základě těchto poznatků z [32] bude v kapitole 4 navrhnuta architektura systému HAPS.

Kapitola 4
Architektura HAPS s posílenou bezpečností

V této kapitole je představena základní architektura HAPS. V podkapitole 4.1 jsou definovány bezpečnostní nároky pro systém HAPS a v podkapitole 4.2 zjednodušená obecná architektura, která slouží pouze jako funkční základ, na kterém jsou představeny bezpečnostní nedostatky takové architektury. V podkapitole 4.3 je navržena zmíněná základní architektura HAPS doplněná o některá z bezpečnostních opatření. Nad touto architekturou s posílenou bezpečností je v kapitole 5 ukázáno modelování hrozeb. Během psaní této práce si nejsem vědom, že by taková architektura již byla k dispozici z dostupných zdrojů. Z tohoto důvodu jsem na základě znalosti předpokládané funkcionality vytvořil zjednodušenou architekturu HAPS, která bude sloužit pro studium bezpečnosti HAPS.

Z důvodu značné rozsáhlosti problému se v návrhu věnuji zejména samotnému HAPSu a jeho interakcí s pozemní kontrolní stanicí. Pro bezpečnost celého systému je ale podstatná i bezpečnost pozemní kontrolní stanice, satelitů interagujícími s HAPSem, ale i určité telekomunikační služby, kterou HAPS provozuje.

4.1 Bezpečnostní nároky

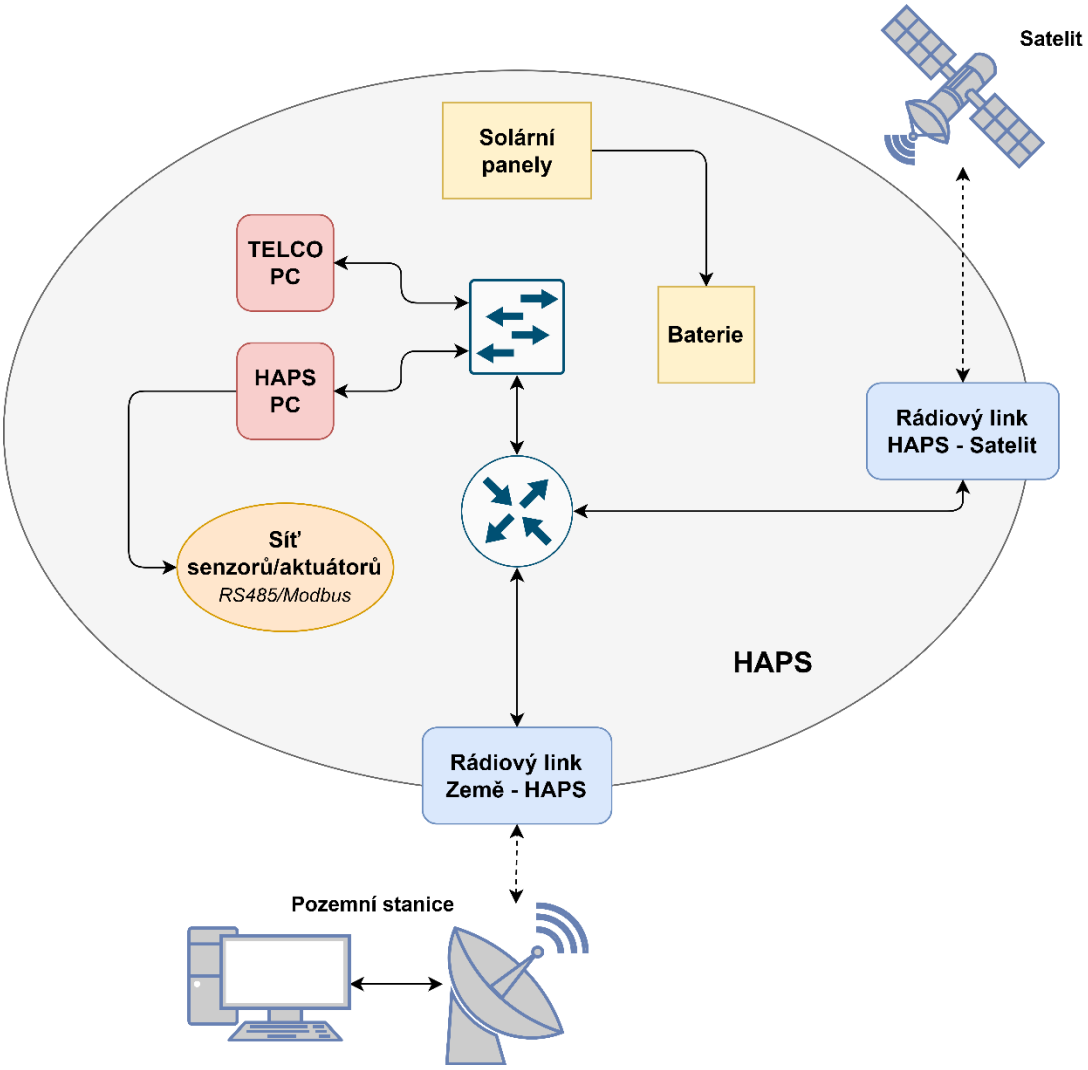
Článek [32] uvádí bezpečnost HAPSů jako jeden z otevřených problémů pro tyto systémy. Autoři v [32] uvádí obavy v případě kompromitace HAPSu a důsledků takového útoku pro integritu a soukromí dat, která HAPSem prochází. Dále uvádí problémy ohledně rušení a odposlouchávání radiového linku mezi pozemní kontrolní stanicí a HAPSem. Autoři v [32] také zmiňují hrozbu převzetí kontroly nad HAPSem, což může představovat nebezpečí – například v podobě střetu s letadly nebo pádu na zem. *V tab. 4.1* jsou uvedené klíčové nároky pro zajištění bezpečnosti HAPSu. Bezpečnostních nároků lze vymyslet více, v tomto případě se ale jedná o ty nejpodstatnější.

Tab. 4.1 *Seznam klíčových nároků pro zajištění bezpečnosti HAPS*

Bezpečnostní nárok	Popis
Zajištění bezpečného letu HAPSu	*Pokud bude HAPS napadnut, je nutné zamezit absolutnímu převzetí kontroly nad jeho letem. Útočník, který by získal kontrolu nad letem HAPSu, by jej mohl zneužít pro způsobení škod na majetku či k ohrožení zdraví a životů osob.*
Zamezení pohybu útočníka v síti HAPSu, pokud bude některá část sítě kompromitována	*HAPS se bude skládat z vícero výpočetních a síťových prvků. Je nutné zajistit, aby nebyl při kompromitaci jednoho z nich ohrožen celý systém. Jedná se zejména o počítač obsluhující telekomunikační služby a počítač obsluhující samotné řízení a funkčnost HAPSu.*
Zajištění integrity a utajení komunikace	*Pro zajištění bezpečnosti HAPS je nezbytně nutné zajistit ochranu komunikace mezi pozemní kontrolní stanicí a samotným HAPSem. V opačném případě by útočník mohl provést různé síťové útoky, které by mohly vést ke kompromitaci HAPSu.*

4.2 Návrh obecné nezabezpečené architektury HAPS

Pro získání lepší představy, jaký systém je třeba zabezpečit, jaké vazby jednotlivé prvky mají a jak spolu interagují, jsem se rozhodl vytvořit zjednodušenou nezabezpečenou architekturu HAPS. Tato architektura, která je zobrazena na *obr. 4.1*, tedy slouží jako pouhá vizualizace celého systému, nikoliv jako návrh s implementovanými bezpečnostními mechanismy.



Obr. 4.1 *Obecná nezabezpečená architektura HAPS*

V této obecné architektuře jsou uvedeny následující prvky:

- Rádiové spoje pro:
  - Země – HAPS
    - Komunikace pozemní kontrolní stanice s HAPSem
    - Komunikace uživatelů a klientů se službou poskytovanou HAPSem
  - HAPS – Satelit

- Dále uvažujeme dva počítače:
    - HAPS PC
        - Řídí autonomní let a funkčnost HAPSu
        - Ovládá aktuátory, čte senzory, reportuje status pozemní stanici, kontroluje napájení, atd.
    - TELCO PC
        - Poskytuje telekomunikační služby
- Síť senzorů/aktuátorů
    - Senzory a aktuátory nezbytné pro let a fungování HAPSu
- Solární panely a baterie
    - HAPS je napájen z baterie, která je dobíjena solárními panely
- Síťové prvky
    - Router
    - Switch

Architektura na *obr. 4.1* je sice funkčním, ale z bezpečnostního hlediska naivním návrhem. Nedostatky této architektury budou hlouběji rozebrány v další části.

4.3 Návrh architektury HAPS s posílenou bezpečností

Z důvodu zásadních bezpečnostních nedostatků architektury prezentované na *obr. 4.1* je nutné pro další postup vytvořit architekturu, která je odolnější vůči kritickým útokům a která lépe splňuje požadavky definované v podkapitole 4.1.

4.3.1 Bezpečnostní problémy obecné architektury HAPS

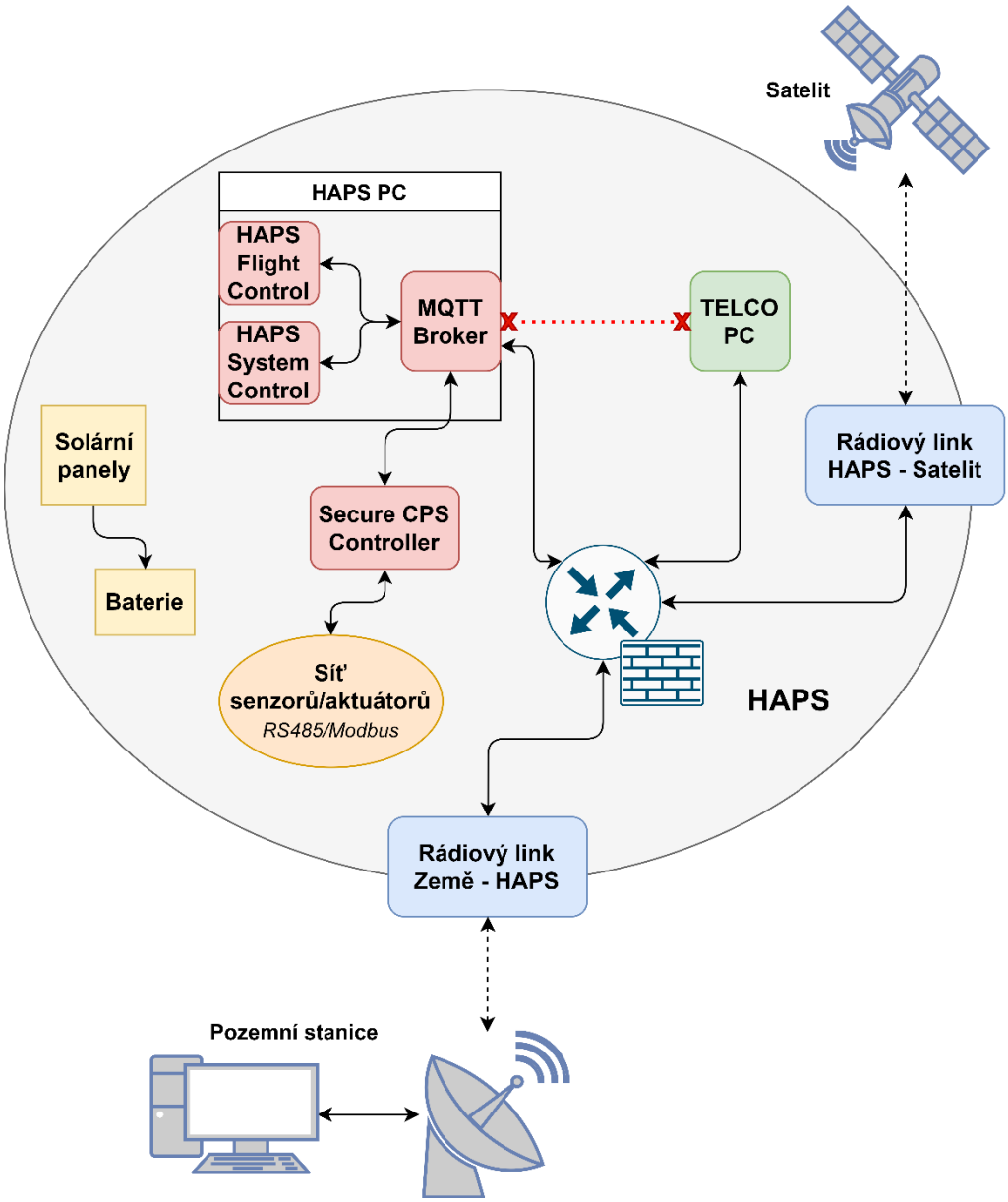
Architektura prezentovaná na *obr. 4.1* má několik zásadních bezpečnostních nedostatků. Tyto nedostatky jsou v rozporu s nároky z *tab. 4.1*.

Prvním zásadním nedostatkem je absence síťové segmentace HAPS PC a TELCO PC. TELCO PC je serverem poskytujícím telekomunikační služby pro určitou skupinu uživatelů, což jej de facto činí veřejně přístupným. V případě, že by byl tento veřejně přístupný server napaden, útočník má díky absenci síťové segmentace možnost laterálního pohybu v síti, čímž potenciálně hrozí i napadení HAPS PC. Tím by byla ohrožena funkčnost a let HAPSu jako takového, což je v rozporu s bezpečnostním požadavkem z *tab. 4.1*. Opačně by při napadení HAPS PC hrozilo mimo jiné i napadení TELCO PC, čímž by byla v ohrožení integrita uživatelských dat a jejich soukromí.

Druhým zásadním problémem je fakt, že k HAPS PC jsou přímo připojené senzory a aktuátory. V případě kompromitace HAPS PC není v architektuře zakomponován žádný prvek, který by zaručil bezpečný let HAPSu. Díky přímému připojení aktuátorů k napadenému HAPS PC by tak útočník získal plnou kontrolu nad letem HAPSu, což je opět v rozporu s bezpečnostním požadavkem z *tab. 4.1*. Jelikož je útočník schopen manipulovat HAPSem, může jej zneužít ke způsobení škod. Vzhledem k manipulaci s HAPSem z jeho zamýšlené pozice je možný i problém s dodržením SLA poskytovaných služeb.

4.3.2 Architektura HAPS s posílenou bezpečností

Architektura HAPS s posílenou bezpečností řeší dva hlavní problémy popsané v předchozí podkapitole. Logické schéma této vylepšené architektury je vyobrazeno na *obr. 4.2*.



Obr. 4.2 *Logická architektura HAPS s posílenou bezpečností*

Na schématu výše je již detailněji popsána stavba HAPS PC. HAPS PC se v tomto návrhu skládá z těchto hlavních komponentů:

- HAPS Flight Control (HFC)
- HAPS System Control (HSC)
- MQTT Broker

HAPS Flight Control (HFC)

Hlavním úkolem HFC je autonomní let a stabilizace HAPSu. HFC řídí HAPS skrze požadavky na SCC na základě zpětné vazby ze senzorů, kterou HFC také získává z SCC.

HAPS System Control (HSC)

Úkolem HSC je snímat data o HAPSu, řídit energetický stav HAPSu a kontrolovat bezchybnost celého systému. HFC řídí pouze let, kdežto HSC je zodpovědný za vše ostatní.

MQTT Broker

Tato komponenta slouží k posílání zpráv pomocí protokolu MQTT, který vznikl pro zařízení připojené do Internetu věcí. Protokol MQTT byl zvolen zejména pro svoji jednoduchost, spolehlivost a malý „overhead“, což šetří síťový provoz, paměť i výkon zařízení. Například ve srovnání s protokolem HTTP se vzhledem k potřebám IoT ukazuje MQTT jako vhodnější volba [34].

Z bezpečnostního hlediska byly v této architektuře implementovány dvě hlavní změny:

- Síťová segmentace HAPS PC a TELCO PC
- Oddělení senzorů a aktuátorů od HAPS PC pomocí „Secure CPS Controller“ (SCC)

Síťová segmentace HAPS PC a TELCO PC

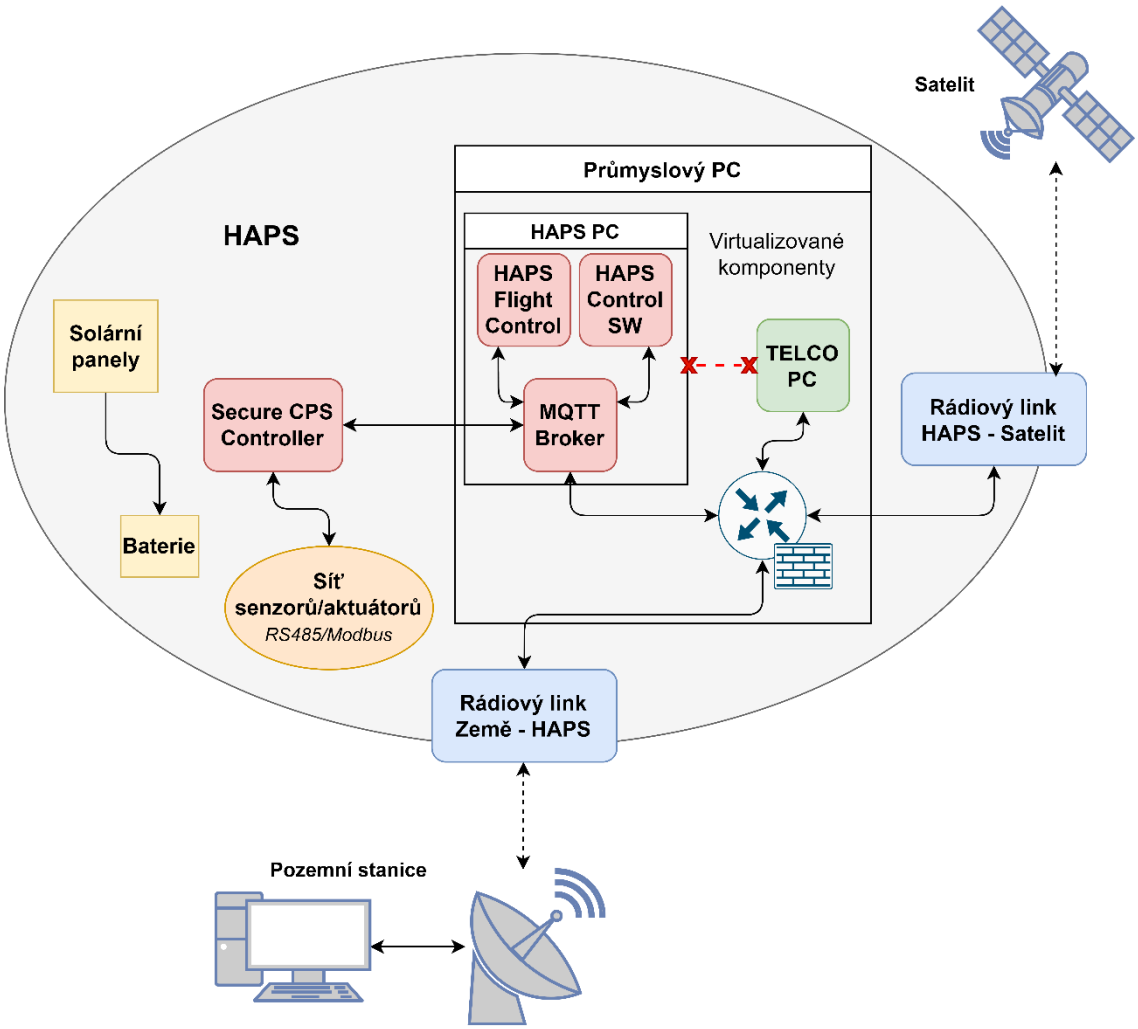
Z architektury byl odstraněn síťový přepínač. HAPS PC i TELCO PC jsou připojeny přímo k routeru. Síťová segmentace na L2 pomocí VLAN není totiž z bezpečnostního hlediska dostatečná díky útoku „VLAN Hopping“ [35]. Na L3 jsou již možnosti síťové segmentace větší. Na routeru s firewallem je možné udělat takovou konfiguraci, aby požadavek segmentace HAPS PC a TELCO PC byl splněn. Firewall na routeru dále filtruje komunikaci k HAPS PC a povoluje provoz tímto směrem pouze z pozemní kontrolní stanice.

Fyzické oddělení senzorů/aktuátorů od HAPS PC

Na *obr. 4.2* přibyla ve srovnání s *obr. 4.1* i nová komponenta „Secure CPS Controller“ (SCC). Tato komponenta představuje způsob fyzického oddělení senzorů/aktuátorů od HAPS PC. V případě, že bude HAPS PC kompromitován útočníkem, nebude mít útočník fyzický přístup k aktuátorům a senzorům, což umožňuje větší stupeň ochrany kontroly letu. Detailnějšímu představení komponenty SCC bude věnovaná další podkapitola.

Dalším důležitým aspektem, který na *obr. 4.2* není vyobrazen, je fyzická realizace celé architektury. Vzhledem k limitované nosnosti HAPSu je vhodné minimalizovat počet fyzických zařízení, které musí HAPS nést. Díky tomu je vhodné využití průmyslového počítače a většinu komponent virtualizovat. Tímto krokem bude ušetřena váha a vznikne větší prostor pro implementaci případných dalších komponent. Fyzická architektura s posílenou bezpečností je vyobrazena na *obr. 4.3*.

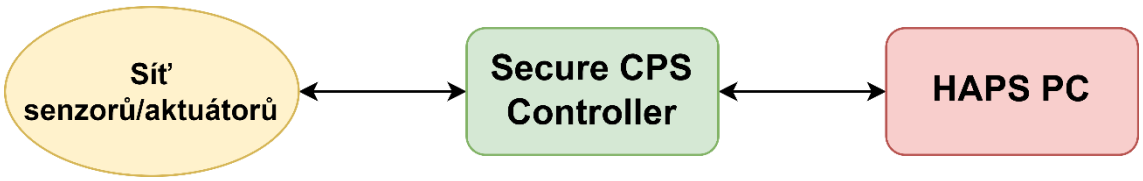
SCC musí zůstat separátním fyzickým zařízením. Přestože logické oddělení by díky virtualizaci bylo zachováno, existují útoky „Virtual Machine Escape“ [36], díky kterým by mohl být ohrožený i virtualizovaný modul SCC v případě kompromitace některé z jiných VM (Virtual Machine), což není žádoucí.



Obr. 4.3 *Fyzická architektura HAPS s posílenou bezpečností*

4.3.3 Koncept Secure CPS Controller (SCC)

Komponenta SCC má za cíl fyzicky oddělit aktuátory a senzory od kontrolního softwaru v HAPS PC, viz *obr. 4.4*. Jedná se tedy o samostatný fyzický prvek, který komunikuje s HAPS PC a ke kterému jsou přímo připojeny senzory a aktuátory. SCC je k HAPS PC připojeno pomocí komunikačního rozhraní a není přístupné z jiné části HAPSu, pouze z HAPS PC.



Obr. 4.4 *Koncept SCC*

SCC je postavené na premise důvěryhodnosti a bezpečnosti. Myšlenka je taková, že do SCC bude extrémně složité proniknout. Tento předpoklad bude splněn zejména díky jednoduchosti SCC. Díky jednoduché struktuře SCC bude minimalizován „attack surface“, čímž budou možnosti útočníka značně omezené.

Jak již bylo uvedeno, senzory a aktuátory budou připojeny přímo k SCC, nikoliv k HAPS PC. Toto fyzické oddělení je nutné, jelikož je HAPS PC, jakožto komplexní systém, potenciálně napadnutelný. V případě napadnutí by hrozilo přímé převzetí kontroly nad letem HAPSu. Fyzickým oddělením je toto riziko minimalizováno.

Komunikace HAPS PC s SCC bude fungovat podle architektury klient – server. HAPS PC, resp. HFC a HSC, budou v roli klienta a SCC v roli serveru. Procesy na HAPS PC tedy budou vysílat požadavky na kontrolu aktuátorů a čtení senzorů a SCC tyto požadavky bude vykonávat a odpovídat na ně. Tyto požadavky ale mohou být pro HAPS škodlivé a proto bude v SCC implementován jakýsi filtr, který tyto požadavky bude zkoumat a povolí pouze ty, které neuvedou systém do nebezpečného stavu. Tento filtr můžeme nazývat jako CPS IPS. Detailněji bude tato myšlenka popsána v další podkapitole.

V neposlední řadě bude SCC zastávat i pozici jakési pojistky. V případě selhání ostatních systémů bude mít SCC implementován algoritmus, díky kterému bude HAPS schopen bezpečně přistát.

4.3.4 Koncept CPS IPS

Koncept CPS IPS (CPS Intrusion Prevention System) je součástí SCC. V případě, že útočník získá kontrolu nad HAPS PC, bude teoreticky schopen vysílat libovolné příkazy pro SCC, nehledě na zprávy generované z řídících procesů na HAPS PC. To znamená, že by mohl vygenerovat takové požadavky, které by mohly uvést HAPS do nebezpečného stavu. Příkladem nebezpečného požadavku může být manipulace s výškou HAPSu z 20 km do cca 10 km, což je výška, ve které létají civilní letadla – je zde tedy riziko kolize.

Úkolem CPS IPS je tyto požadavky identifikovat a blokovat. Pro HAPS bude definována množina stavů, které odpovídají stavům bezpečným. Jakýkoliv požadavek, který by uvedl HAPS mimo tuto množinu povolených bezpečných stavů, bude zamítnut. Pokud opět uvážíme příklad s manipulací výšky letu HAPSu a definujeme povolenou množinu hodnot výšky letu v rozmezí například 19 km až 21 km, pak právě příkaz, kterým by útočník chtěl dostat HAPS mimo tyto hodnoty, bude zablokován a SCC jej nevykoná. Přestože útočník má určité možnosti manipulace s HAPSem, nebude schopen provést akce, které by dostaly HAPS do nebezpečných stavů. Tato funkcionalita je předpokladem pro zajištění bezpečnostního nároku z *tab. 4.1* v podobě bezpečného letu HAPSu.

Dalším úkolem CPS IPS je provádět analýzu senzorických dat. Tato analýza bude zaměřená na detekci anomálií v senzorických datech. Díky této analýze budou detekovány jak pokusy o úmyslné ovlivnění senzorických dat, tak reálná senzorická data, která vypovídají například o poruchovém stavu HAPSu. V těchto případech bude vygenerováno upozornění, které bude moci vyhodnotit obsluha řízení provozu na pozemní kontrolní stanici.

Kapitola 5
Analýza kyberfyzikálních hrozeb v systému HAPS

Tato kapitola se zabývá zejména modelováním bezpečnostních hrozeb a bezpečnostní analýzou. Hlavním úkolem je identifikovat přítomné hrozby v architektuře navržené v podkapitole 4.3 pomocí běžně užívaných modelů. Identifikovaným hrozbám bude dále přidělena závažnost. U nejzávažnějších hrozeb budou nastavena taková bezpečnostní opatření, aby byly tyto hrozby mitigovány.

5.1 Úvod do modelování hrozeb

Hlavní smysl modelování hrozeb spočívá v metodickém a strukturovaném způsobu přemýšlení o zkoumaném návrhu, který vede k identifikování hrozeb. Hrozba může být zranitelnost zneužitelná útočníkem, ale také může existovat hrozba samovolného incidentu, jako je například porucha nějakého prvku systému. Identifikováním a uvědoměním si těchto hrozeb lze navrhnout opatření, která tyto hrozby mitigují [37].

Při modelování hrozeb si lze položit následující čtyři otázky [37]:

- **Na jakém systému pracujeme?**
  - Klíčové je porozumět rozsahu problému, který chceme analyzovat
  - Rozložení aplikace/systému na menší části
- **Co se může pokazit?**
  - Úkolem bezpečnostního analytika je identifikovat hrozby
  - Je vhodné použít strukturované modely pro systematičtější přístup
- **Co s tím můžeme udělat?**
  - Nejprve je vhodné uvědomit si míru rizika, které konkrétní hrozby představují
  - Pokud je riziko nepřijatelné, navrhnou se taková bezpečnostní opatření, která snižují riziko na přijatelnou mez
  - Pokud je riziko hrozby nízké, mohu toto riziko akceptovat
- **Odvedli jsme dobrou práci?**
  - Kontrola zpracované bezpečnostní analýzy a zhodnocení naší práce

Prvním krokem v modelování hrozeb a současně odpovědí na první z otázek, je porozumět samotné aplikaci a tomu, jak funguje. Je vhodné vytvořit diagram datových toků (DFD – Data Flow Diagram), kde jsou znázorněny interakce jednotlivých procesů a entit v celé aplikaci/systému [38].

Po dekompozici aplikace/systému se dále identifikují hrozby a hodnotí se rizika [38]. Pro systematický přístup k identifikaci hrozeb byl použit model STRIDE. Ke zhodnocení rizik, které jednotlivé hrozby představují byl využit model DREAD.

5.1.1 Model STRIDE

Model STRIDE byl vyvinut inženýry ve společnosti Microsoft, kde byl používán pro modelování hrozeb [39]. Tento model je využíván pro systematický přístup k identifikaci hrozeb. Může být použit při modelování hrozeb širokého spektra systémů – od programů, webových aplikací, fyzických zařízení, či právě kyberfyzikálních systémů.

Slovo „STRIDE“ představuje mnemotechnickou pomůcku, kde každé písmeno reprezentuje jeden druh hrozby. Shrnutí těchto hrozeb je uvedeno v následující tabulce, která je inspirovaná podobnou tabulkou z [40].

Tab. 5.1 *Hrozby v modelu STRIDE*

Hrozba	Popis hrozby	Požadovaná vlastnost
Spoofing	*Vydávání se za něco/někoho jiného*	*Authentication*
Tampering	*Manipulace obsahu informace*	*Integrity*
Repudiation	*Popření provedení úkonu*	*Non-repudiation*
Information disclosure	*Přístup k informaci neautorizovanou entitou*	*Confidentiality*
Denial of service	*Odepření služby legitimním uživatelům*	*Availability*
Elevation of privilege	*Získání přístupu neautorizovanou entitou*	*Authorization*

Detailnější informace o modelu STRIDE je možné získat z [38], [39] a [40].

5.1.2 Model DREAD

Model DREAD je využíván pro zhodnocení rizik, které představují již identifikované hrozby. Podobně jako model STRIDE, byl i tento model vyvinut a používán společností Microsoft [41]. Další podobnost s modelem STRIDE je fakt, že se opět jedná o mnemotechnickou pomůcku, kde každé písmeno reprezentuje metriku pro hodnocení rizika.

Hlavním problémem modelu DREAD je absence jasných definicí jednotlivých metrik a fakt, že samotné hodnocení je z velké části subjektivní. Zkratka DREAD reprezentuje tyto metriky [41]:

***Tab. 5.2** Metriky k hodnocení hrozeb v modelu DREAD*

Metrika	Popis metriky
Damage	*Jak velké škody způsobí zneužití hrozby?*
Reproducibility	*Jak snadné je pro útočníka reprodukovat útok?*
Exploitability	*Kolik úsilí stojí útočníka využít hrozbu k provedení útoku?*
Affected Users	*Kolik uživatelů je ovlivněno touto hrozbou?*
Discoverability	*Jak snadné je hrozbu objevit?*

Každá metrika uvedená v *tab. 5.2* musí být ohodnocena určitým počtem bodů. Způsob hodnocení se v jiných zdrojích liší. Zatímco v [38] je každá metrika hodnocena maximálně 10 body a finální skóre je určeno průměrem všech metrik, v [41] a [42] je systém jednodušší. Každé metrice jsou přiděleny body od 1 do 3 podle toho, jak závažné riziko daná hrozba u konkrétní metriky představuje – **1** (Low), **2** (Medium), **3** (High). Finální skóre je dáno jako součet všech bodů, tudíž 5-15. Celkové riziko hrozby je pak rozděleno do tří kategorií – Low, Medium a High. Bodové meze pro toto rozdělení jsou definovány v [42] a jsou uvedeny v následující tabulce. Pro hodnocení rizik navržené architektury byl použit právě tento systém.

***Tab. 5.3** Hodnocení rizik u modelu DREAD*

Hodnocení rizika	Výsledné skóre
High	**12 – 15**
Medium	**8 – 11**
Low	**5 – 7**

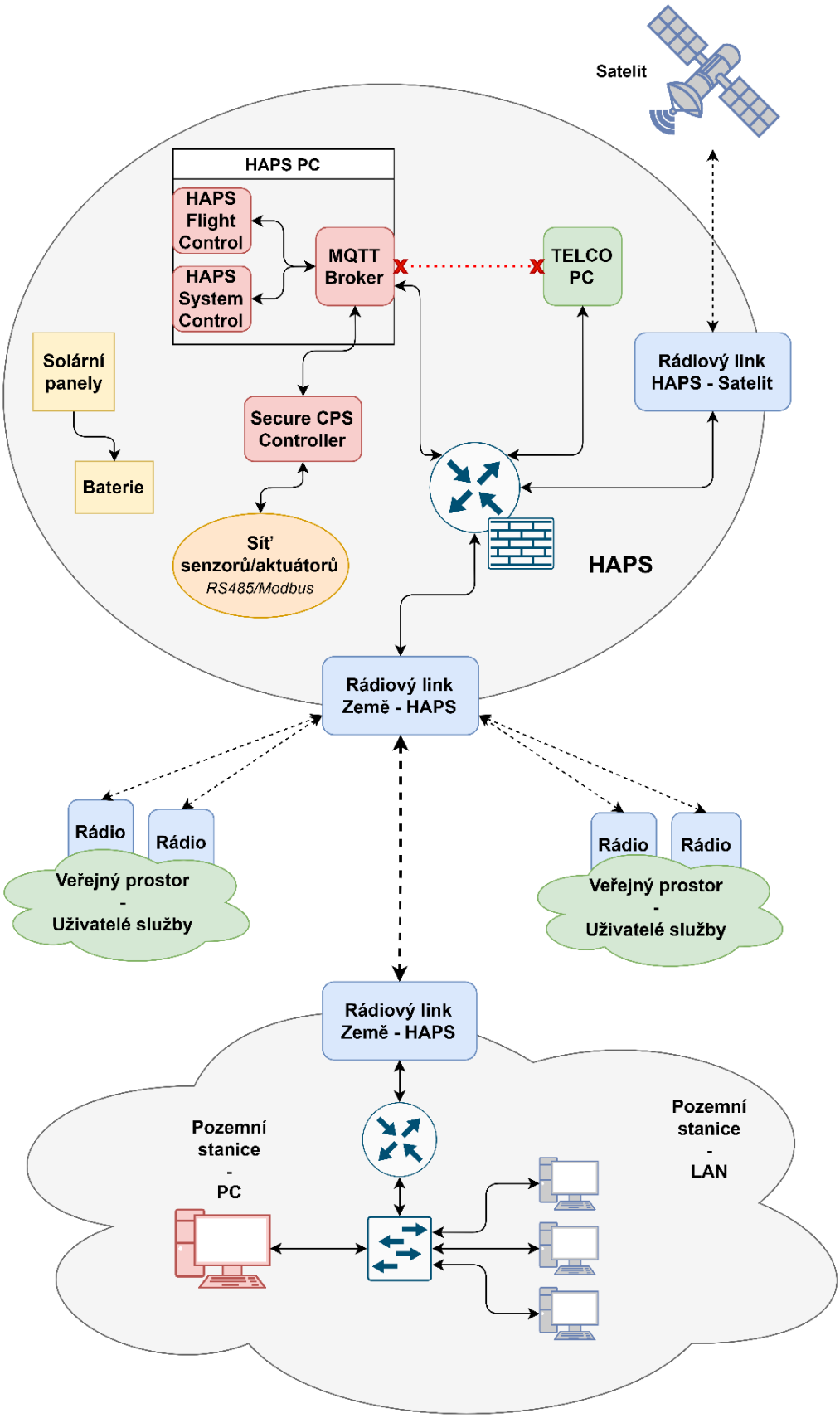
V podkapitole 5.5.1 jsou definovány metriky modelu DREAD pro použití v prostředí HAPS. Následně jsou navržena pravidla, podle kterých jsou jednotlivé metriky bodovány. Cílem je minimalizovat vliv subjektivního vnímání jednotlivých rizik na celkové hodnocení.

5.2 Uvažovaná architektura – rozšíření o pozemní kontrolní stanici

Pro účely modelování hrozeb byla architektura navržená v podkapitole 4.3 doplněna o model pozemní kontrolní stanice, viz *obr. 5.1*. Je nutné si uvědomit, že kompromitace pozemní stanice je z bezpečnostního hlediska další možností útočníka jak napadnout HAPS. Při získání kontroly nad pozemní stanicí hrozí, že by útočník mohl získat citlivé informace, částečnou kontrolu nad HAPSem či cestu, skrze kterou lze napadnout přímo HAPS a kompromitovat jeho systémy. Z těchto důvodů je nezbytné brát pozemní kontrolní stanici v potaz v případě modelování hrozeb.

Doplněná pozemní kontrolní stanice je pro potřeby modelování hrozeb reprezentována jednoduchou LAN se směrovačem a přepínačem. V síti bude přítomno více koncových stanic, kde jedna z těchto koncových stanic bude sloužit jako kontrolní stanice. Tento případ je úmyslně zvolen jakožto případ síťového návrhu bez hlubších bezpečnostních prvků – není tedy z bezpečnostního hlediska vhodný. Příkladem úmyslně špatného bezpečnostního návrhu může být absence síťového oddělení kontrolní stanice od zbytku sítě. Je důležité si uvědomit další hrozby, které v tomto případě představují potenciálně napadnutelné stanice v LAN.

Dále byla architektura doplněna o reprezentaci uživatelů telekomunikačních služeb, které HAPS poskytuje skrze TELCO PC.



Obr. 5.1 *Architektura HAPS s pozemní kontrolní stanicí*

5.3 Modelování hrozeb pomocí modelu STRIDE

Pro modelování hrozeb byl využit nástroj „Microsoft Threat Modeling Tool“ [43]. V tomto nástroji byl na základě architektury z *obr. 5.1* vytvořen diagram datových toků, který je hlouběji rozebrán v další části textu.

Následně byly identifikovány hrozby v jednotlivých interakcích. Metodika použití modelu STRIDE v prostředí kyberfyzikálních systémů byla inspirována prací [44] vědců z Queen's University Belfast. Konkrétně byla použita tzv. metodika „STRIDE-per-interaction“.

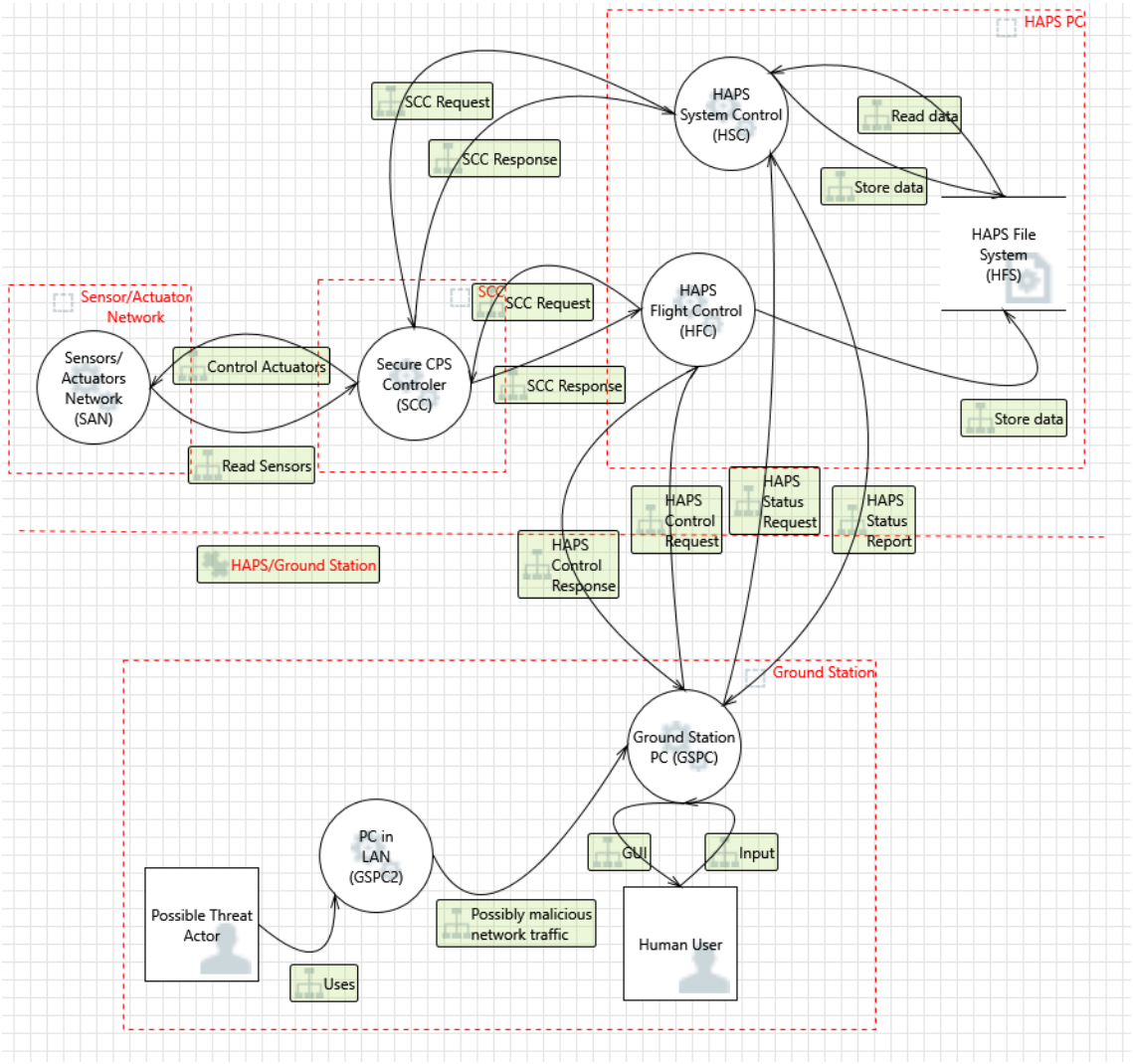
Pro přílišnou složitost výsledného modelování hrozeb nebyly některé části architektury při tvorbě DFD brány v potaz. Do DFD nebyl uvažován TELCO PC, a to z důvodu, že je již síťově oddělen od zbytku sítě HAPS a jeho případná kompromitace nepředstavuje pro HAPS přímé nebezpečí. Tato práce se zabývá bezpečností samotného systému HAPS, nikoliv bezpečností telekomunikačních služeb, které na něm budou provozovány. Tato problematika je pak v gesci provozovatelů dané telekomunikační služby.

Podobně byl kvůli složitosti vynechán i rádiový link „HAPS – Satelit“. Přestože útok skrze satelit není možné zcela vyloučit, je však jeho pravděpodobnost nižší oproti případům, na které se model bude zaměřovat. Lze ale konstatovat, že hrozby identifikované na tomto linku budou mít analogický charakter jako hrozby, které mohou být identifikovány na rádiovém linku „Země – HAPS“ a stejně tak je analogický i způsob mitigace těchto hrozeb.

5.3.1 Diagram datových toků navrženého systému HAPS

Na *obr. 5.2* je zobrazen uvažovaný DFD, na kterém bude prováděna analýza hrozeb. Je nutné uvést, že se jedná pouze o modelový příklad interakcí v systému a nikoliv o reálný komplexní návrh celého systému. V době psaní této práce nejsou známé přesné specifikace a požadavky na systém HAPS, tudíž se při bezpečnostní analýze opírám o pravděpodobné scénáře, které by měly být blízké reálnému řešení.

Pro přehlednost byl také v HAPS PC vynechán MQTT Broker. Jelikož se jedná de facto o přeposílací prvek, uvažováním MQTT Brokeru v DFD bychom ztratili přehled o tom, jaké komponenty spolu reálně interagují. MQTT Broker je ale stále nutné uvažovat pro další bezpečnostní návrhy.



Obr. 5.2 *Diagram datových toků systému HAPS*

Celý diagram na *obr. 5.2* je rozdělen do čtyř zón – Pozemní Stanice, HAPS PC, SCC, Síť senzorů/aktuátorů.

Ground Station (GS) – Pozemní stanice

- Ground Station PC (GSPC)
  - Počítač fungující jako pozemní kontrolní stanice
  - Komunikuje s HAPS PC procesy přes MQTT Broker
  - Posílá řídící příkazy HAPS PC
  - Ovládá ho člověk – nejedná se o plně automatizovaný program pro autonomní řízení
- PC in GS LAN (GSPC2)
  - Reprezentuje další počítač v LAN pozemní stanice
  - Může být napaden a zneužit k útokům v této síti
  - Může být ovládán potenciálním útočníkem
- Ground Station Human User (GSHU)
  - Lidský uživatel pozemní kontrolní stanice
  - Pracuje s GSPC pomocí GUI

HAPS PC

- HAPS Flight Control (HFC)
  - Řídí autonomní létání HAPSu
  - Řízení provádí skrze požadavky na SCC
  - Přijímá požadavky od GSPC
- HAPS System Control (HSC)
  - Řeší systémové věci jako např.:
    - Hlídání napájení
    - Status všech senzorů – stav celého systému
  - Informace o systému reportuje na pozemní stanici
  - Informace loguje do souborového systému (úložiště)
- HAPS File System (HFS)
  - Slouží pro ukládání logů a jiných potřebných dat

SCC

- SCC
  - Komunikuje s HAPS PC
  - Přijímá požadavky na řízení aktuátorů a tyto požadavky vykonává
  - Přijímá požadavky na čtení senzorických dat a poskytuje senzorická data
  - Řídí síť senzorů a aktuátorů

Sensor/Actuator Network – Síť senzorů/aktuátorů

- Sensor/Actuator Network (SAN)
  - Obsahuje senzory a aktuátory nutné pro funkčnost HAPSu
  - Senzory a aktuátory jsou připojeny k SCC a řízeny skrze SCC

5.3.2 Množina následků hrozeb

Pro další analýzu byly identifikovány následky uskutečnění hrozeb (TC - Threat Consequences). Zde je uveden jejich seznam:

- **TC-1** - Převzetí kontroly nad řízením HAPSu
- **TC-2** - Ovlivnění řízení HAPSu
- **TC-3** - Ztráta kontroly nad HAPSem
- **TC-4** - Neoprávněný přístup k datům o systému HAPS
- **TC-5** - Systém nemá pravdivé informace o skutečném stavu HAPS
- **TC-6** - Systém nemá žádné informace o aktuálním stavu HAPS
- **TC-7** - HAPS není schopen autonomního provozu
- **TC-8** - Získání neautorizovaného přístupu do systému
- **TC-9** - Elevace privilegií v rámci systému

5.4 Identifikované hrozby

V následujících podkapitolách jsou identifikovány interakce, u kterých je relevantní příslušný druh hrozby. Shrnutí této analýzy je k dispozici v podkapitole 5.4.7.

5.4.1 Spoofing

Spoofing u interakce SCC → SAN není technicky možný, protože žádný jiný prvek není připojen přímo k SAN. To znamená, že i kdyby se někdo za SCC vydával, nebude schopen SAN ovládat. V opačném směru, tzn. SAN → SCC je možný například spoofing GPS, jak již bylo uvedeno v podkapitole 2.3.2 (TC-2, TC-5).

Závažné důsledky může mít spoofing u interakce HFC → SCC, či HSC → SCC. Pokud by se útočníkovi podařilo vydávat se za HFC nebo HSC, mohl by získat kontrolu nad letem HAPSu a přístup k senzorickým datům o stavu HAPSu skrze generování falešných požadavků na SCC (TC-1, TC-4).

V případě spoofingu v interakci SCC → HFC, či SCC → HSC by bylo možné odesílat falešná data. To může vést k tomu, že systém nezná reálný stav HAPSu. Díky autonomnímu řízení letu na základě zpětné vazby ze senzorických dat je teoreticky možné ovlivnit let HAPSu (TC-1, TC-2, TC-5).

Díky tomu, že komunikace s GS probíhá přes rádiový link, je tato komunikace ohrožená více nežli předchozí příklady. Spoofing interakcí GSPC → HFC a GSPC → HSC je závažná hrozba. Pokud se útočník dokáže vydávat za GSPC, je efektivně možné posílat příkazy HFC, ovládat tak let HAPSu a získávat cenné informace o systému skrze reporty z HSC (TC-1, TC-4).

V interakci GS – HAPS je možný spoofing také na straně HAPSu. Pokud se útočník bude vydávat za HAPS (HFC, HSC), je možné provést útok MITM, kdy reálné GSPC bude posílat příkazy falešnému HFC, ten následně může falešné příkazy posílat skutečnému HFC a zároveň GSPC ubezpečovat, že jeho požadavky byly v pořádku vykonány. Analogická situace nastává u spoofingu HSC (TC-1, TC-3, TC-4, TC-5, TC-6).

5.4.2 Tampering

U interakce SAN → SCC existuje riziko modifikace senzorických dat. Tato hrozba je sice nepravděpodobná, ale pokud by se útočníkovi podařilo externím vlivem ovlivnit senzorická data, systém by měl nepravdivé informace o reálném stavu HAPSu (TC-2, TC-5).

Interakce HFC → SCC a HSC → SCC je potenciálně modifikovatelná. Při kompromitaci HAPS PC může útočník měnit vyslané požadavky na SCC. Tím je ohroženo řízení HAPSu. Útočník také může získat citlivé údaje o systému (TC-1, TC-4).

V případě interakcí HSC → HFS a HFC → HFS je teoreticky možné změnit, nebo dokonce mazat data, která jsou ukládána do logů (TC-5).

Díky možnosti MITM útoku na linku GS – HAPS je možné měnit zprávy v interakcích mezi GSPC, HFC a HSC. Tato hrozba má závažné důsledky v podobě převzetí řízení HAPSu,

potenciální nemožnosti ovládat HAPS, přístup k citlivým datům o systému a nemožnosti získat reálná data o systému (TC-1, TC-3, TC-4, TC-5, TC-6).

V případě napadení GSPC je teoreticky možné zobrazovat uživateli obsluhující GSPC pomocí GUI falešná data – jedná se o interakci GSPC → GSHU (TC-5).

5.4.3 Repudiation

V HAPS PC je nastaveno logování událostí v rámci vnitřních procesů (HSC, HFC, HFS). Popření činů (repudiation) je možné v interakcích, kde figuruje SCC.

V interakci mezi GS a HAPS je nutné nastavit logování interakcí. Přišlé požadavky a odchozí odpovědi je nutné logovat a to jak na straně HAPS, tak na straně GS.

5.4.4 Information disclosure

Pokud je útočník schopen zachytávat komunikaci mezi HFC, HSC a SCC, je schopen získat přístup k citlivým datům o systému (TC-4).

Přístup k datům uloženým v HFS musí být náležité chráněn. V případě, že přístup k uloženým datům nebude dostatečně dobře ošetřen, hrozí neoprávněný přístup k datům systému v interakci HFS → HSC (TC-4).

Jelikož komunikace GS a HAPS probíhá po rádiovém spojení, je možné tuto rádiovou komunikaci odposlouchávat. Útočník je tímto schopen získat přístup k obsahu této komunikace (TC-4).

Na GSPC se předpokládá použití monitorovacího systému s vizualizací dat. Pokud nebude přístup k tomuto systému (aplikaci) zabezpečen, je teoreticky možné, že k těmto datům dostane přístup neautorizovaná osoba s přístupem do GS (TC-4).

5.4.5 Denial of Service

K Denial of Service (DoS) může dojít díky cílenému útoku, ale i díky chybě v programu, či mechanické poruše. Je nutné nastavit takové mechanismy, aby se byl systém v těchto případech schopen dostat zpět do provozuschopného stavu.

V případě interakce SAN → SCC je možné, že na některém ze senzorů/aktuátorů nastane porucha. Také může nastat situace, kdy senzor bude reportovat takovou hodnotu (např. díky chybě senzoru), na kterou program na SCC není připraven a díky tomu dojde k chybě v programu SCC vedoucí k jeho pádu (TC-3, TC-6, TC-7).

Interakce v opačném směru SCC → SAN může vyslat takový požadavek, který nebyl očekáván firmwarem daného senzoru či aktuátoru, což může mít za následek pád programu běžící na mikrokontroleru daného senzoru/aktuátoru (TC-3, TC-6, TC-7).

U interakcí HFC, HSC s SCC je možné, že dojde k softwarové chybě a tyto programy se zastaví. Dále je možné, že pokud útočník získá přístup k některému z procesů, může případnou

softwarovou chybu zneužít a tento chybový stav vyvolat k provedení útoku DoS. V případě kompromitace HAPS PC může útočník odstavit celý systém. Tento scénář by měl fatální následky, jelikož komunikace GSPC s SCC probíhá nepřímo skrze HAPS PC. Efektivně tak dojde ke ztrátě kontroly nad HAPSem (TC-3, TC-6, TC-7).

Rádiová komunikace mezi GS a HAPS může být náchylná na úmyslné rušení rádiového signálu – tzv. „jamming“. V případě MITM útoku je také útočník schopen zablokovat požadavky jdoucí z pozemní stanice k HAPSu (TC-3, TC-6).

Specifický případ DoS je teoreticky možné vyvolat u interakce GSPC2 → GSPC. Pokud bude mít útočník přístup ke GSPC2, lze potenciálně zneužít případnou zranitelnost na GSPC a způsobit tak DoS na GSPC (TC-3).

5.4.6 Elevation of Privilege

Díky potenciálním zranitelnostem ve zpracování požadavků na SCC je teoreticky možné na SCC realizovat Remote Code Execution (RCE). Pokud by byl kompromitovaný HAPS PC, RCE je potenciálně možné provést v interakcích HFC → SCC a HSC → SCC pomocí vhodně upraveného obsahu komunikace. Útočník tak může získat přístup do SCC a zvýšit tak svá privilegia v rámci sítě HAPSu (TC-8, TC-9).

Obdobný útok v opačném směru by neměl být možný díky předpokladu, že útočník nemá k SCC přístup. Aby získal přístup k SCC, musel by již mít přístup do HAPS PC a muselo by se mu podařit zneužít závažné zranitelnosti v SCC.

V případě kompromitace GSPC se může útočník pokusit získat přístup k HAPS PC zneužitím případných zranitelností v interakcích GSPC → HFC a GSPC → HSC (TC-8, TC-9).

V opačném směru – pokud již nějakým způsobem získal útočník přístup do HAPS PC, může se pokusit útočit na GSPC a získat k němu přístup (TC-8).

Pokud má útočník přístup ke GSPC2 v LAN GS, je potenciálně možné napadnout přímo GSPC (TC-7, TC-8).

Další hrozbou (která není uvedena v tabulce níže) je hrozba elevace privilegií uvnitř jedné entity, například HAPS PC. Pokud útočník získá přístup k určitému systému, hrozí, že se útočníkovi podaří zvýšit svá práva v rámci tohoto uzavřeného systému. Typicky se jedná o elevaci privilegií ze standardního uživatele systému na uživatele „root“. Toho může útočník dosáhnout díky zranitelnostem některých služeb v systému, zranitelnostem v kernelu OS či špatnou bezpečnostní konfigurací systému.

5.4.7 Tabulka identifikovaných hrozeb pomocí modelu STRIDE

V *tab. 5.4* se nachází identifikovaných hrozeb pomocí modelu STRIDE.

Tab. 5.4 *Identifikované hrozby pomocí modelu STRIDE*

Interakce	Popis interakce	S	T	R	I	D	E
SAN → SCC	*Senzorická data, odpověď od aktuátorů*	X	X	X		X	
SCC → SAN	*Ovládání aktuátorů, čtení senzorických dat*			X		X	
HFC → SCC	*Požadavky – čtení senzorických dat, ovládání aktuátorů*	X	X		X	X	X
SCC → HFC	*Odpověď na požadavky, senzorická data*	X		X	X	X	
HSC → SCC	*Požadavky – senzorická data, stavy aktuátorů*	X	X		X	X	X
SCC → HSC	*Odpovědi na požadavky, senzorická data*	X		X	X	X	
HSC → HFS	*Ukládání dat o stavu HAPSu + logování*		X				
HFC → HFS	*Ukládání dat o stavu HAPSu + logování*		X				
HFS → HSC	*Čtení dat o stavu HAPSu, čtení logů*				X		
GSPC → HFC	*Požadavek k vykonání požadavků v rámci SCC*	X	X	X	X	X	X
HFC → GSPC	*Odpovědi na požadavky, senzorická data*	X	X		X	X	X
HSC → GSPC	*Reporty o stavu HAPSu*	X	X		X	X	X
GSPC → HSC	*Požadavek o zaslání reportu o stavu HAPSu*	X	X	X	X	X	X
GSPC → GSHU	*Zobrazení uživatelského GUI uživateli*		X		X		
GSPC2 → GSPC	*Obecná interakce PC v LAN s GSPC*					X	X

5.5 Hodnocení identifikovaných hrozeb pomocí modelu DREAD

Metodika hodnocení rizik identifikovaných hrozeb byla popsána v podkapitole 5.1.2. V této podkapitole je také uveden jeden z hlavních nedostatků modelu DREAD, který spočívá v příliš subjektivním hodnocení. Způsob minimalizování tohoto nedostatku spatřuji ve vytvoření tabulky, definované v podkapitole 5.5.1, s objektivními kritérii pro hodnocení daných metrik.

Dále jsou vyhodnoceny rizika identifikovaných hrozeb u jednotlivých uzlů z DFD z *obr. 5.2*. Hodnocení hrozeb se věnují podkapitoly 5.5.2 až 5.5.8.

V podkapitole 5.5.9 je uvedena krátká statistika shrnující závažnosti identifikovaných hrozeb.

5.5.1 Definice metrik

Cílem této části je vytvořit tabulku, ve které je jasně definováno, při jakých podmínkách je určité metrice udělen daný počet bodů (1 až 3). K vytvoření této tabulky je nejprve vhodné najít správné otázky, které se při hodnocení budou klást. Odpověďmi na tyto otázky pak získáme jasná kritéria, podle kterých jsme schopni metriky ohodnotit.

Jednotlivé metriky jsou popsány v podkapitole 5.1.2 v *tab. 5.2*. Níže budou uvedeny otázky pro jednotlivé metriky.

Damage – *Jak velké škody způsobí zneužití hrozby?*

- Jak citlivé informace unikly a kolik jich bylo?
- Je ohrožen autonomní let HAPSu?
- Jak moc je útočník schopen zasahovat do kontroly letu HAPSu?
- Je útočník schopen poškodit části HAPSu? Jaký je rozsah škod?
- V případě, že hrozí ztráta dat, kolik dat bude ztraceno?
- Pokud je útočník schopen falšovat informace o systému, jaký dopad to bude mít?

Reliability / Reproducibility – *Jak snadné je pro útočníka reprodukovat útok?*

- Může útočník použít útok kdykoliv?
- Potřebuje k tomu nějaké speciální podmínky?

Exploitability – *Kolik úsilí stojí útočníka využít hrozbu k provedení útoku?*

- Jak složité je útok provést?
- Jak zkušený musí útočník být?
- Kolik času potřebuje útočník k provedení útoku?
- Vyžaduje útok nějaké specifické nástroje, které nejsou běžně dostupné?
- Je útok možné provést automatizované?

Affected Users - *Kolik uživatelů je ovlivněno touto hrozbou?*

- Dokáže útočník ovlivnit služby poskytované HAPSem? Jak moc?
- Dojde k porušení SLA – ano/ne?

Discoverability - *Jak snadné je hrozbu objevit?*

- Jak těžké je zranitelnost objevit?
- Je nutné mít přístup ke zdrojovému kódu?

Dále je uvedena tabulka specifikující kritéria pro hodnocení metrik skrze odpovědi na výše položené otázky. Vzhledem k faktu, že se stále jedná o teoretickou rovinu, nelze při hodnocení metrik ověřit, že skutečnost odpovídá kritériu uvedenému v tabulce. Určitá míra subjektivního zkreslení je zde tedy stále přítomna.

***Tab. 5.5** Definice kritérií pro hodnocení metrik v modelu DREAD*

Metrika	Low (1)	Medium (2)	High (3)
Damage	• Přístup k obecným informacím • Let HAPSu není ohrožen • Poškození HAPSu nehrozí • Nehrozí ztráta důležitých dat • Falšovaná data mají minimální dopad na systém	• Přístup k důležitým informacím • Let HAPSu může být ovlivněn • Hrozí poškození nekritických částí systému (omezení SCC) • Nehrozí ztráta důležitých dat • Falšovaná data mohou mít znatelný dopad na systém	• Přístup ke kritickým informacím • Útočník může kontrolovat HAPS (omezení SCC) • Let HAPSu může být paralyzován • Hrozí poškození kritických částí systému (omezení HAPS) • Hrozí ztráta dat • Falšovaná data mají znatelný dopad na funkčnost celého systému
Reproducibility / Reliability	• Zranitelnost je nestabilní a špatně využitelná k útoku • Vyžaduje velmi specifické podmínky pro úspěšný útok • Náhodná porucha pouze v nepravděpodobných a uměle vyvolaných případech	• Zranitelnost lze využít k útoku za splnění určitých podmínek, které mohou nastat při běžném provozu • Náhodná porucha s velmi malou pravděpodobností i v normálním provozu	• Zranitelnost je triviálně využitelná za každé situace • Náhodná porucha může s nemalou pravděpodobností nastat za normálního provozu
Exploitability	• Útok vyžaduje extrémně zkušeného útočníka a velice detailní informace o celém systému • Útok je velice náročný na čas	• Zkušený útočník je schopen zneužít tuto zranitelnost v rozumném čase s detailními informacemi o funkčnosti systému • Středně dlouhý čas na spuštění útoku	• Zranitelnost je zneužitelná i málo zkušeným útočníkem v krátkém čase, je veřejně dostupný funkční exploit • Krátký čas na spuštění útoku
Affected Users	• Zranitelnost neohrožuje poskytované služby	• Zranitelnost může v malé míře ohrozit kvalitu poskytovaných služeb pro malou skupinu uživatelů	• Zranitelnost může ohrozit kvalitu poskytovaných služeb pro větší skupinu uživatelů
Discoverability	• Zranitelnost je velice těžké najít i pokud má útočník přístup k celému systému a všem informacím o něm	• Zranitelnost je těžké najít, lze jí objevit náhodou, či detailním studováním celého systému	• Zranitelnost lze jednoduše objevit, informace o této zranitelnosti již mohou být ve veřejném prostoru

5.5.2 Hodnocení rizik identifikovaných hrozeb

V této části budou vyhodnoceny rizika identifikovaných hrozeb u interakcí uzlů z DFD. Do hrozeb uzlu uvažujeme ty hrozby, jejichž interakce z daného uzlu vychází. Finální hodnocení je rozřazeno do kategorií Low, Medium a High na základě bodového hodnocení, jehož meze jsou definovány v *tab. 5.3* v podkapitole 5.1.2.

Hodnocení hrozeb uzlu „Sensor/Actuator Network“ – SAN

Tab. 5.6 *Hodnocení hrozeb uzlu SAN*

#	Interakce	Hrozba	D	R	E	A	D	∑	Rating
1	SAN → SCC	Spoofing senzorů (např. GPS Spoofing)	2	2	2	1	2	9	Medium
2	SAN → SCC	Vnější ovlivnění senzorických dat	2	2	1	1	1	7	Low
3	SAN → SCC	Absence logování na mikrokontrolerech	1	1	1	1	1	5	Low
4	SAN → SCC	Výpadek senzorů/aktuátorů	3	2	1	1	1	8	Medium

Hodnocení hrozeb uzlu „Secure CPS Controller“ – SCC

Tab 5.7 *Hodnocení hrozeb uzlu SCC*

#	Interakce	Hrozba	D	R	E	A	D	∑	Riziko
1	SCC → SAN	Absence logování příkazů z SCC na SAN	1	1	1	1	1	5	Low
2	SCC → SAN	Výpadek procesu SCC při interakci se SAN	3	1	1	2	1	8	Medium
3	SCC → HFC	Spoofing SCC při interakci s HFC	3	2	2	2	3	12	High
4	SCC → HFC	Absence logování příchozích požadavků	1	1	1	1	1	5	Low
5	SCC → HFC	Sniffing odpovědí od SCC pro HFC	2	2	2	1	3	10	Medium
6	SCC → HFC	Výpadek procesu SCC při interakci s HFC	3	1	1	2	1	8	Medium
7	SCC → HSC	Spoofing SCC při interakci s HSC	2	2	2	2	3	11	Medium
8	SCC → HSC	Absence logování příchozích požadavků	1	1	1	1	1	5	Low
9	SCC → HSC	Sniffing odpovědí od SCC pro HSC	2	2	2	1	3	10	Medium
10	SCC → HSC	Výpadek procesu SCC při interakci s HSC	3	1	1	2	1	8	Medium

Hodnocení hrozeb uzlu „HAPS Flight Control“ – HFC

Tab. 5.8 *Hodnocení hrozeb uzlu HFC*

#	Interakce	Hrozba	D	R	E	A	D	∑	Riziko
1	**HFC → SCC**	Spoofing HFC při interakci s SCC	3	2	2	2	3	**12**	**High**
2	**HFC → SCC**	Modifikace požadavků pro SCC	3	2	2	2	2	**11**	**Medium**
3	**HFC → SCC**	Sniffing požadavků pro SCC	1	2	2	1	3	**9**	**Medium**
4	**HFC → SCC**	Výpadek procesu HFC při interakci s SCC	3	2	1	2	2	**10**	**Medium**
5	**HFC → SCC**	Elevace privilegií pomocí RCE na SCC	3	1	1	3	1	**9**	**Medium**
6	**HFC → HFS**	Změna ukládaných dat z HFC do HFS	1	3	2	1	2	**9**	**Medium**
7	**HFC → GSPC**	Spoofing HFC při interakci s GSPC	2	2	2	1	3	**10**	**Medium**
8	**HFC → GSPC**	Modifikace obsahu komunikace HFC s GSPC	2	2	2	1	3	**10**	**Medium**
9	**HFC → GSPC**	Sniffing komunikace HFC s GSPC	1	3	3	2	3	**12**	**High**
10	**HFC → GSPC**	Jamming komunikace HFC s GSPC	2	3	2	2	3	**12**	**High**
11	**HFC → GSPC**	Elevace privilegií pomocí RCE na GSPC	3	1	1	2	1	**8**	**Medium**

Hodnocení hrozeb uzlu „HAPS System Control“ – HSC

Tab. 5.9 *Hodnocení hrozeb uzlu HSC*

#	Interakce	Hrozba	D	R	E	A	D	∑	Riziko
1	**HSC → SCC**	Spoofing HSC při interakci s SCC	2	2	2	2	3	**11**	**Medium**
2	**HSC → SCC**	Modifikace požadavků pro SCC	2	2	2	2	2	**10**	**Medium**
3	**HSC → SCC**	Sniffing požadavků pro SCC	1	2	2	1	3	**9**	**Medium**
4	**HSC → SCC**	Výpadek procesu HSC při interakci s SCC	2	2	1	2	2	**9**	**Medium**
5	**HSC → SCC**	Elevace privilegií pomocí RCE na SCC	3	1	1	3	1	**9**	**Medium**
6	**HSC → HFS**	Změna ukládaných dat z HSC do HFS	2	3	2	2	2	**11**	**Medium**
7	**HSC → GSPC**	Spoofing HSC při interakci s GSPC	2	2	2	1	3	**10**	**Medium**
8	**HSC → GSPC**	Modifikace obsahu komunikace HSC s GSPC	2	2	2	1	3	**10**	**Medium**
9	**HSC → GSPC**	Sniffing komunikace HSC s GSPC	2	3	3	1	3	**12**	**High**
10	**HSC → GSPC**	Jamming komunikace HSC s GSPC	2	3	2	2	3	**12**	**High**
11	**HSC → GSPC**	Elevace privilegií pomocí RCE na GSPC	3	1	1	2	1	**8**	**Medium**

Hodnocení hrozeb uzlu „HAPS File System“ – HFS

Tab. 5.10 *Hodnocení hrozeb uzlu HFS*

#	Interakce	Hrozba	D	R	E	A	D	∑	Riziko
1	**HFS → HSC**	Neoprávněný přístup k uloženým datům	3	2	2	1	3	**11**	**Medium**

Hodnocení hrozeb uzlu „Ground Station PC“ – GSPC

Tab. 5.11 *Hodnocení hrozeb uzlu GSPC*

#	Interakce	Hrozba	D	R	E	A	D	∑	Riziko
1	**GSPC → HFC**	Spoofing GSPC při interakci s HFC	3	3	2	3	3	**14**	**High**
2	**GSPC → HFC**	Modifikace obsahu komunikace GSPC s HFC	3	2	2	3	3	**13**	**High**
3	**GSPC → HFC**	Nedostatečné logování událostí na GSPC	1	1	1	1	1	**5**	**Low**
4	**GSPC → HFC**	Sniffing komunikace GSPC s HFC	2	3	3	1	3	**12**	**High**
5	**GSPC → HFC**	Jamming komunikace GSPC s HFC + DoS	3	3	2	2	3	**13**	**High**
6	**GSPC → HFC**	Elevace privilegií pomocí RCE na HFC	3	2	2	2	2	**11**	**Medium**
7	**GSPC → HSC**	Spoofing GSPC při interakci s HSC	2	3	2	1	3	**11**	**Medium**
8	**GSPC → HSC**	Modifikace obsahu komunikace GSPC s HSC	1	2	2	1	3	**9**	**Medium**
9	**GSPC → HSC**	Nedostatečné logování událostí na GSPC	1	1	1	1	1	**5**	**Low**
10	**GSPC → HSC**	Sniffing komunikace GSPC s HSC	2	3	3	1	3	**12**	**High**
11	**GSPC → HSC**	Jamming komunikace GSPC s HSC + DoS	2	3	2	1	3	**11**	**Medium**
12	**GSPC → HSC**	Elevace privilegií pomocí RCE na HSC	3	2	2	1	2	**10**	**Medium**
13	**GSPC → GSHU**	Modifikace zobrazených informací v GUI	1	2	1	1	1	**6**	**Low**
14	**GSPC → GSHU**	Nezabezpečený přístup ke GUI	1	2	3	1	3	**10**	**Medium**

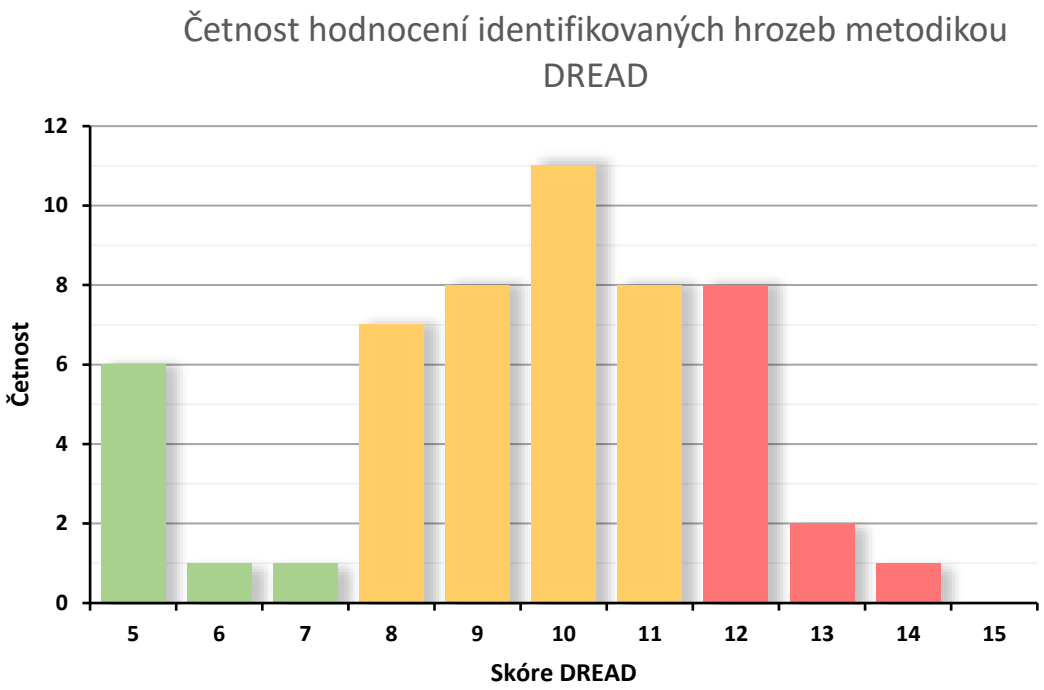
Hodnocení hrozeb uzlu „PC in GS LAN“ – GSPC2

Tab. 5.12 *Hodnocení hrozeb uzlu GSPC2*

#	Interakce	Hrozba	D	R	E	A	D	∑	Riziko
1	**GSPC2 → GSPC**	Zneužití GSPC2 k DoS útoku na GSPC	1	2	2	1	2	**8**	**Medium**
2	**GSPC2 → GSPC**	Zneužití GSPC2 k útoku na GSPC pomocí RCE	3	2	2	1	2	**10**	**Medium**

5.5.3 Statistika závažnosti rizik identifikovaných hrozeb

V podkapitole 5.4 bylo identifikováno celkem 53 hrozeb. Rozložení bodového hodnocení identifikovaných hrozeb je ukázáno na *obr. 5.3*.



Obr. 5.3 *Četnost hodnocení identifikovaných hrozeb metodikou DREAD*

Většina identifikovaných hrozeb byla hodnocena jako středně závažná. Celkem bylo hodnoceno 9 hrozeb jako málo závažné, 33 hrozeb jako středně závažné a 11 hrozeb jako vysoce závažné.

Hrozby, které byly identifikovány jako vysoce závažné je nutné považovat za priority při dalším návrhu bezpečnostních opatření architektury HAPS.

Kapitola 6
Návrh bezpečnostních opatření v architektuře HAPS

Tato kapitola se zabývá návrhem bezpečnostních opatření k mitigování identifikovaných hrozeb. Vzhledem k rozsahu takového úkolu jsou nejprve uvedena pouze obecná bezpečnostní řešení pro všechny identifikované hrozby. Vybrané nejzávažnější hrozby jsou detailně analyzovány a je předložen detailnější návrh bezpečnostních opatření pro jejich odstranění. Poté je představena finální podoba architektury HAPS, do které jsou již zakomponovány přidané bezpečnostní návrhy.

6.1 Obecná bezpečnostní opatření pro identifikované hrozby

V následujících částech jsou pro každý uzel systému HAPS uvedena obecná bezpečnostní řešení identifikovaných hrozeb.

6.1.1 Sensor/Actuator Network (SAN)

Hrozba 1 – Spoofing senzorů (např. GPS Spoofing)

- Riziko: Medium
- Řešení: Různé druhy senzorů vyžadují různé přístupy pro mitigaci hrozby „sensor spoofing“. V [45] popisují autoři spoofing útok na senzor infuzní pumpy pro podávání medikace. Mitigace takové hrozby spočívá v kombinaci detekčních metod a prevence v podobě fyzické izolace senzorů pro znemožnění externího ovlivnění senzorů. Takový přístup lze nasadit i v případě HAPS. Ochrana proti GPS spoofingu se liší na základě druhu GPS spoofing útoku. Ochranou mohou být detekční algoritmy na základě přijatých signálů, využití senzorické fúze pro konfirmaci údajů z GPS, či nasazení kryptografie [46]. Další přehled metod pro mitigaci GPS spoofingu je k dispozici v [47].

Hrozba 2 – Vnější ovlivnění senzorických dat

- Riziko: Low
- Řešení: SCC musí správně kontrolovat, zda jsou senzorická data ve stanovených mezích. Musí být nastaven kontrolní algoritmus, který zkoumá vývoj dat v čase. Tento algoritmus může být založen na vícero metodách v závislosti na zdroji a typu dat – může se tedy jednat o pouhé prahování, využití regresní analýzy pomocí Deep Neural Network (DNN) či o kontrolu dodržení fyzikálních zákonů. Dále lze také využít detekce ovlivnění senzorických dat na základě konfirmace dat z několika zdrojů pomocí senzorické fúze.

Hrozba 3 – Absence logování na mikrokontrolerech

- Riziko: Low
- Řešení: Logování příkazů na SCC.

Hrozba 4 – Výpadek senzorů/aktuátorů

- Riziko: Medium
- Řešení: Aby bylo možné výpadkům senzorů a aktuátorů předejít, je zásadní odolný softwarový návrh firmwaru senzorů/aktuátorů a důsledné testování navrženého FW/SW. Součástí návrhu musí být i způsob restartování zařízení při detekci kritické chyby v softwaru. Pro ošetření mechanických poruch je vhodné kritické senzory/aktuátory zdvojit (pokud je to možné), aby byl odstraněn problém „Single Point of Failure“ (SPOF).

6.1.2 Secure CPS Controller (SCC)

Hrozba 1, 4, 8 – Absence logování příkazů z SCC na SAN, Absence logování příchozích požadavků na SCC z HFC a HSC

- Riziko: Low
- Řešení: Logování odchozích příkazů pro prvky v SAN. Logování příchozích požadavků a odpovědí na ně. Ukládání logů musí být nastaveno s takovými bezpečnostními pravidly, aby běžný uživatel systému nemohl tyto logy upravovat. Některé logy mohou být posílány přímo na pozemní kontrolní stanici. Tyto logy musí být ošetřeny digitálním podpisem, aby byla zajištěna autenticita a integrita dat v logu.

Hrozba 2, 6, 10 – Výpadek procesu SCC při interakci se SAN, HFC a HSC

- Riziko: Medium
- Řešení: Aby bylo možné předejít softwarovým poruchám způsobující DoS, je nutný odolný softwarový návrh aplikace běžící na SCC a důsledné testování naprogramovaného SW, včetně testování systému jako celku. Dále je také nutné, aby v návrhu komponenty byl integrován mechanismus pro autonomní restart v případě detekce softwarové chyby. Hardware, na kterém bude SCC postaven musí být vhodně vybrán, zejména s ohledem na provozní podmínky ve stratosféře. Je také nutné uvažovat dostatečný „Mean Time to Failure“ (MTTF). Ekonomicky a technicky náročnějším řešením by bylo vytvoření zálohy SCC k odstranění problému SPOF.

Hrozba 3, 7 – Spoofing SCC při interakci s HFC, Spoofing SCC při interakci s HSC

- Riziko: High (3), Medium (7)
- Řešení: Nasazení autentizačních mechanismů. Autentizace SCC vůči HFC a HSC.

Hrozba 5, 9 – Sniffing odpovědí od SCC pro HFC a HSC

- Riziko: Medium
- Řešení: Šifrování komunikace mezi SCC a HFC, HSC.

6.1.3 HAPS PC Flight Control (HFC)

Hrozba 1, 7 – Spoofing HFC při interakci s SCC, Spoofing HFC při interakci s GSPC

- Riziko: High (1), Medium (7)
- Řešení: Autentizace HFC vůči SCC. Autentizace HFC vůči GSPC.

Hrozba 2, 8 – Modifikace požadavků pro SCC, Modifikace obsahu komunikace HFC s GSPC

- Riziko: Medium
- Řešení: Zajištění integrity dat, pomocí vhodných kryptografických protokolů.

Hrozba 3, 9 – Sniffing požadavků pro SCC, Sniffing komunikace HFC s GSPC

- Riziko: Medium (3), High (9)
- Řešení: Šifrování komunikace mezi SCC a HFC. Šifrování komunikace mezi HFC a GSPC.

Hrozba 4 – Výpadek procesu HFC při interakci s SCC

- Riziko: Medium
- Řešení: Odolný softwarový návrh a testování SW. Vhodný výběr HW HAPS PC.

Hrozba 5, 11 – Elevace privilegií pomocí RCE na SCC, Elevace privilegií pomocí RCE na GSPC

- Riziko: Medium
- Řešení Návrh a vývoj aplikace na SCC, resp. GSPC musí probíhat dle vhodných bezpečnostních doporučení. Vhodné je podrobit SCC, resp. GSPC důkladnému penetračnímu testování pro odhalení zranitelností. Filozofie vývoje softwarových komponent by měla být v souladu s minimalizací tzv. „attack surface“. HFC musí být taktéž bezpečně vyvíjeno a podrobeno penetračnímu testování, aby nemohl být tento proces v první řadě zneužit pro další útočníkův pohyb v systému.

Hrozba 6 – Změna ukládaných dat z HFC do HFS

- Riziko: Medium
- Řešení: Konfigurace přístupu do HFS. Mechanismy autorizace.

Hrozba 10 – Jamming komunikace HFC s GSPC

- Riziko: High
- Řešení: Nasazení opatření proti rádiovému jammingu, jako např. „frequency hopping over spread spectrum“, atd.

6.1.4 HAPS PC System Control (HSC)

Hrozby a způsoby jejich mitigace jsou analogické jako v případě HFC.

6.1.5 HAPS PC File System (HFS)

Hrozba 1 – Neoprávněný přístup k uloženým datům

- Riziko: Medium
- Řešení: Autorizace, kontrola přístupu do úložiště.

6.1.6 Ground Station PC (GSPC)

Hrozba 1, 7 – Spoofing GSPC při interakci s HFC, Spoofing GSPC při interakci s HSC

- Riziko: High (1), Medium (7)
- Řešení: Autentizace GSPC vůči HFC a HSC.

Hrozba 2, 8 – Modifikace obsahu komunikace GSPC s HFC a HSC

- Riziko: High (2), Medium (8)
- Řešení: Zajištění integrity dat, pomocí vhodných kryptografických protokolů.

Hrozba 3, 9 – Nedostatečné logování událostí na GSPC

- Riziko: Low
- Řešení: Bezpečné logování událostí na GSPC.

Hrozba 4, 10 – Sniffing komunikace GSPC s HFC a HSC

- Riziko: High
- Řešení: Šifrování komunikace GSPC s HFC a HSC.

Hrozba 5, 11 – Jamming komunikace GSPC s HFC + DoS, Jamming komunikace GSPC s HSC + DoS

- Riziko: High (5), Medium (11)
- Řešení: Nasazení opatření proti rádiovému jammingu, jako např. „frequency hopping over spread spectrum“, atd. Dále může docházet k cíleným DoS útokům vlivem softwarových zranitelností. Nutností je penetrační testování HFC a HSC pro odhalení těchto zranitelností. Dalším opatřením je autentizace a autorizace komunikace.

Hrozba 6, 12 – Elevace privilegií pomocí RCE na HFC a HSC

- Riziko: Medium
- Řešení: Zabezpečení GSPC a celé sítě pozemní stanice, aby nemohl být GSPC využitý pro tyto účely. Penetrační testování HFC a HSC pro odhalení zranitelností vedoucí k RCE.

Hrozba 13 – Modifikace zobrazených informací v GUI

- Riziko: Low
- Řešení: Aby tato situace nastala, musel by být GSPC infikovaný. Důležité je nastavit bezpečnostní mechanismy, aby k tomu nedošlo. GUI aplikace by také měla být podrobena bezpečnostní analýze.

Hrozba 14 – Nezabezpečený přístup ke GUI

- Riziko: Medium
- Řešení: Přístup ke GUI aplikaci musí být poskytnut pouze autorizovaným osobám. Autorizace pomocí přihlašovacích údajů. Kladení důrazu na dodržení silných uživatelských hesel, případně využití biometrie.

6.1.7 Ground Station PC 2 (GSPC2)

Hrozba 1, 2 – Zneužití GSPC2 k DoS útoku na GSPC, Zneužití GSPC2 k útoku na GSPC pomocí RCE

- Riziko: Medium
- Řešení: Tato hrozba zdůrazňuje důležitost zabezpečení ostatních koncových a síťových zařízení v celé síti pozemní stanice. Zabezpečení celé sítě a všech zařízení v ní je zásadním požadavkem pro bezpečnost celého systému. Zabezpečení sítě by mělo obsahovat síťovou segmentaci GSPC od zbytku LAN pro eliminace těchto útoků. Přístup ke GSPC by měl být umožněn pouze autorizovaným osobám.

6.2 Návrh konkrétních bezpečnostních opatření

Úkolem této podkapitoly je předložit návrh konkrétních bezpečnostních opatření. Návrh opatření je rozdělen do dvou částí. První částí je návrh bezpečnostních opatření na úrovni samotného HAPSu, viz podkapitola 6.2.1. Tato sada opatření slouží k mitigování některých identifikovaných zranitelností přímo uvnitř sítě HAPSu.

Druhou částí je sada bezpečnostních opatření zaměřující se na bezpečnost interakce HAPSu a pozemní kontrolní stanice, viz podkapitola 6.2.2.

6.2.1 Diskuse k návrhu bezpečnostních opatření na úrovni HAPSu

Mezi hrozby s největším vyhodnoceným rizikem, které byly identifikovány uvnitř HAPSu (SCC, HFC, HSC), patřily zejména hrozby týkající se zajištění následujících vlastností:

- Utajení (Confidentiality) – ochrana proti přístupu k datům neautorizovaným aktérem
- Integrita (Integrity) – ochrana proti změně obsahu dat neautorizovaným aktérem
- Autentizace (Authentication) – ověření totožnosti komunikujících stran

Navržená sada opatření by tedy měla cílit zejména na dosažení těchto bezpečnostních vlastností.

Nutnost zajištění těchto bezpečnostních prvků může být na první pohled diskutabilní, a to z několika důvodů. Aby útočník mohl tyto útoky provádět, musel by získat přístup do sítě HAPSu, což se může jevit jako poměrně náročné z dnešního úhlu pohledu. Nicméně v návrhu je nutné počítat s tím, že je infiltrace HAPSu útočníkem reálným scénářem.

Dalším faktem je, že pokud se již útočník zmocní kontroly nad např. HAPS PC, již bude aktérem, který by byl schopen dešifrovat a modifikovat komunikaci a také by byl vůči SCC autentizovaný. Ochranná opatření by v tomto případě nebyla tolik účinná. Lze ale dojít k několika hypotetickým situacím, které by nasazení takových bezpečnostních mechanismů ospravedlňovaly. Jako příklad lze uvést následující situace:

- Napadení TELCO PC a následné obcházení, či selhání síťové segmentace
    - V tomto případě by měl útočník z TELCO PC přístup k HAPS PC a SCC.
    - Teoreticky hrozí spoofing útok z infikovaného TELCO PC.
- Útočník dostane fyzický přístup k HAPS (na zemi, při vývoji…)
    - To mu teoreticky umožní provést úpravu HAPSu a například mezi SCC a HAPS PC vložit zařízení, čímž může efektivně dosáhnout MITM (Man in the Middle) útoku

6.2.2 Návrh bezpečnostních opatření na úrovni HAPSu

Bezpečnost komunikace uvnitř HAPSu

Spojení SCC – HAPS PC by mělo být šifrované, oboustranně autentizované (po vzoru „zero trust architecture“) a měla by být zajištěna integrita dat. Pro tyto účely je vhodné využít protokol TLS, a to konkrétně ve verzi TLS 1.3. Tato verze přináší oproti té starší několik výhod [48]:

- Rychlejší handshake
- Nižší latence
- Perfect forward secrecy
- Nepodporuje zranitelné algoritmy a šifry

TLS 1.3 podporuje pouze 5 různých „cipher suites“, viz [49]. Hlavní volba stojí zejména mezi blokovými a proudovými šifrovacími algoritmy – konkrétně AES vs ChaCha20. Lze očekávat, že SCC nebude mít CPU s AES-NI, tudíž nebude schopen použít hardwarové instrukce pro AES. Dle [50] je algoritmus ChaCha20 až 3x rychlejší na mobilních procesorech bez hardwarových instrukcí pro AES. Pro zabezpečení komunikace HAPS PC – SCC tedy bude vhodné zvolit „cipher suite“ **TLS_CHACHA20_POLY1305_SHA256**.

TLS serverem bude v tomto případě HAPS PC, protože je to právě ten, na kterém běží MQTT Broker, ve kterém bude TLS spojení terminováno.

Dále je nutné vytvořit Public Key Infrastructure (PKI). PKI bude založená na certifikátech X.509. Jako certifikační autorita (CA) bude vystupovat společnost odpovědná za provoz HAPSů. Soukromé klíče CA musí být bezpečně uložené na zemi, ideálně v „Hardware Security Module“ (HSM). SCC i HAPS PC budou mít každý svůj X.509 certifikát podepsaný CA a kořenový certifikát CA.

Opatření proti MQTT Broker SPOF

MQTT Broker je v navržené architektuře kriticky důležitým prvkem. V případě jeho nefunkčnosti dojde k výpadku celé komunikace HAPSu s pozemní stanicí, čímž by stanice ztratila nad HAPSem kontrolu. V této situaci by musel po určité době převzít řízení modul SCC, který by zahájil sekvenci nouzového přistání. Nicméně k této situaci má dojít až v případech krajní nouze. MQTT Broker může být schválně odstaven z provozu i útočníkem, který převzal kontrolu nad HAPS PC.

Řešením SPOF problému MQTT Brokeru je nasadit do sítě HAPSu druhý, záložní MQTT Broker, který bude využit pouze v případě nefunkčnosti brokeru na HAPS PC. SCC by k tomuto záložnímu brokeru mělo přístup nehledě na stav HAPS PC. V případě detekovaných problémů (nefunkčnost HAPS PC) by se pozemní kontrolní stanice i SCC připojili k záložnímu brokeru a komunikace by dále fungovala. Kontrola letu by mohla dávat přímé rozkazy HAPSu z pozemní stanice a HAPS by nemusel skončit v režimu nouzového přistání.

Pro správný přechod na záložní broker musí být implementovaný algoritmus, který spolehlivě zjistí nefunkčnost HAPS PC, či potenciální modifikaci komunikace, která by značila napadení HAPS PC. Jednou z možností, jak detekovat spojení SCC – GSPC je využití „heartbeat“ zpráv. SCC by periodicky odesílalo „heartbeat“ zprávy, které by byly ošetřeny digitálním podpisem pomocí soukromého klíče SCC. Pozemní kontrolní stanice by na tyto „heartbeat“ zprávy musela odpovědět též podepsanou „heartbeat“ zprávou. Oba koncové body znají veřejné klíče protistrany, tudíž by pravost mohly ověřit. Při reálné komunikaci by pak bylo vyžadováno

potvrzení přijetí zprávy, či potvrzení vykonání požadavku. Selhání těchto podmínek by bylo důvodem pro přechod na záložní MQTT Broker.

Pro záložní broker by také platilo zabezpečení komunikace pomocí TLS.

Zajištění autenticity a integrity reportů pozemní stanici ohledně stavu HAPSu

Z DFD na *obr. 5.2* vyplývá, že reporty obsahující stav HAPSu pro pozemní kontrolní stanici posílá HAPS PC skrze proces HSC. Zde vyvstává bezpečnostní problém v podobě modifikace, či falzifikace dat v těchto reportech. Pokud bude mít útočník přístup k HAPS PC, může tyto reporty modifikovat tak, aby se jevilo vše v pořádku a zároveň manipulovat HAPSem (v rámci možností SCC).

Data, která se budou na pozemní stanici posílat musí z tohoto důvodu být opatřena digitálním podpisem ze strany SCC. Bude tak ošetřen původ a integrita dat. Tato podepsaná data pak již mohou být uložena na HAPS PC a zasílána pozemní stanici odtud.

Jelikož digitální podpis může být výpočetně náročnou operací, je vhodné, aby SCC podepisovalo svazek dat nasbíraných za delší časový úsek, nikoliv jednotlivá měření senzorů.

Ochrana HAPS PC pomocí Firewall a IPS + DPI

Jelikož je většina provozu na HAPS PC šifrována, klasické „rule based“ IPS, jako například Suricata, či Snort by byly málo účinné. Existují ale i případy, kdy komunikace šifrovaná být nemusí. Šlo by zejména o případy, kdy selhaly jiné bezpečnostní opatření a došlo k infiltraci systémů, které jsou dále využívány pro další útoky, například právě na HAPS PC. Typicky se může jednat o napadení pozemní kontrolní stanice, odkud může útočník napadat HAPS PC. Nasazení „rule based“ IPS zde tedy má určitý smysl. Je ale vhodné doplnit tento druh IPS o „machine learning based“ IPS. Tyto IPS jsou vhodnější, protože dokážou fungovat i nad šifrovanou komunikací. Příkladem takové IPS může být „Stratosphere IPS“, která je vyvíjená na ČVUT FEL [51].

Vhodným doplňkem je také monitorování síťového provozu pomocí DPI (Deep Packet Inspection). Díky DPI mohou administrátoři získat cenné informace o povaze a původu síťového provozu na HAPS PC.

6.2.3 Diskuse k návrhu bezpečnostních opatření mezi pozemní stanicí a HAPSem

Hrozby s největším vyhodnoceným rizikem jsou podobné jako v případě podkapitoly 6.2.1, nicméně lze najít i hrozby specifické pro spojení mezi HAPSem a pozemní kontrolní stanicí. Tyto hrozby se týkají zejména zajištění následujících vlastností:

- Utajení (Confidentiality) – ochrana proti přístupu k datům neautorizovaným aktérem
- Integrita (Integrity) – ochrana proti změně obsahu dat neautorizovaným aktérem
- Autentizace (Authentication) – ověření totožnosti komunikujících stran
- Odolnost proti L1 DoS – odolnost proti rušení rádiové komunikace
- Síťová bezpečnost pozemní stanice

V návrhu této diplomové práce nejsou zahrnuty opatření pro zajištění poslední z vyjmenovaných vlastností – tedy zajištění síťové bezpečnosti pozemní stanice. Jedná se o zásadní bezpečnostní problém, který je třeba vyřešit, nicméně v tomto případě je mimo vytyčené cíle této práce.

Navržená opatření by se tedy měla v co největší míře zabývat zabezpečením rádiového linku Země – HAPS. Je také nutné zajistit ochranu proti cílenému rušení rádiové komunikace, či nabídnout alternativní prostředky, jak se může HAPS se pozemní stanicí spojit.

6.2.4 Bezpečnostní opatření mezi pozemní stanicí a HAPSem

Bezpečnost rádiové komunikace

Zabezpečení rádiového linku by mělo být zajištěno již samotným komunikačním modulem pro Point-to-Point (P2P) komunikaci. Tato rádiová komunikační rozhraní mají často ochranu přenosu již zabudovanou, často v podobě protokolu WPA2 (Wi-Fi Protected Access 2). Stále je ale nutné na tento faktor myslet při výběru tohoto komunikačního modulu.

Je nutné zabezpečit i provoz samotné aplikace, kdy je vhodné použít protokol TLS 1.3 a oboustrannou autentizaci, kdy se klient i server navzájem autentizují. Volba „cipher suite“ záleží zejména na podpoře AES-NI v CPU na HAPS PC a GSPC. V tomto případě lze očekávat, že se bude jednat o výkonná zařízení s využitím moderních CPU, kde je podpora AES-NI. Z tohoto důvodu je vhodné využít „cipher suite“ **TLS_AES_128_GCM_SHA256**.

Opatření proti úmyslnému rušení rádiové komunikace

Příklady řešení jammingu rádiové komunikace jsou k dispozici v [52]. Toto řešení již ale musí být implementováno v zařízení pro rádiovou komunikaci, tudíž i zde je nutné brát na tento fakt zřetel při výběru komunikačního modulu.

Pokud i tak dojde k rušení rádiového linku do takové míry, že přímá komunikace pozemní stanice a HAPSu nebude možná, lze využít komunikaci přes satelitní rozhraní, jakožto záložní link pro spojení s HAPSem.

6.2.5 Shrnutí navržených bezpečnostních opatření

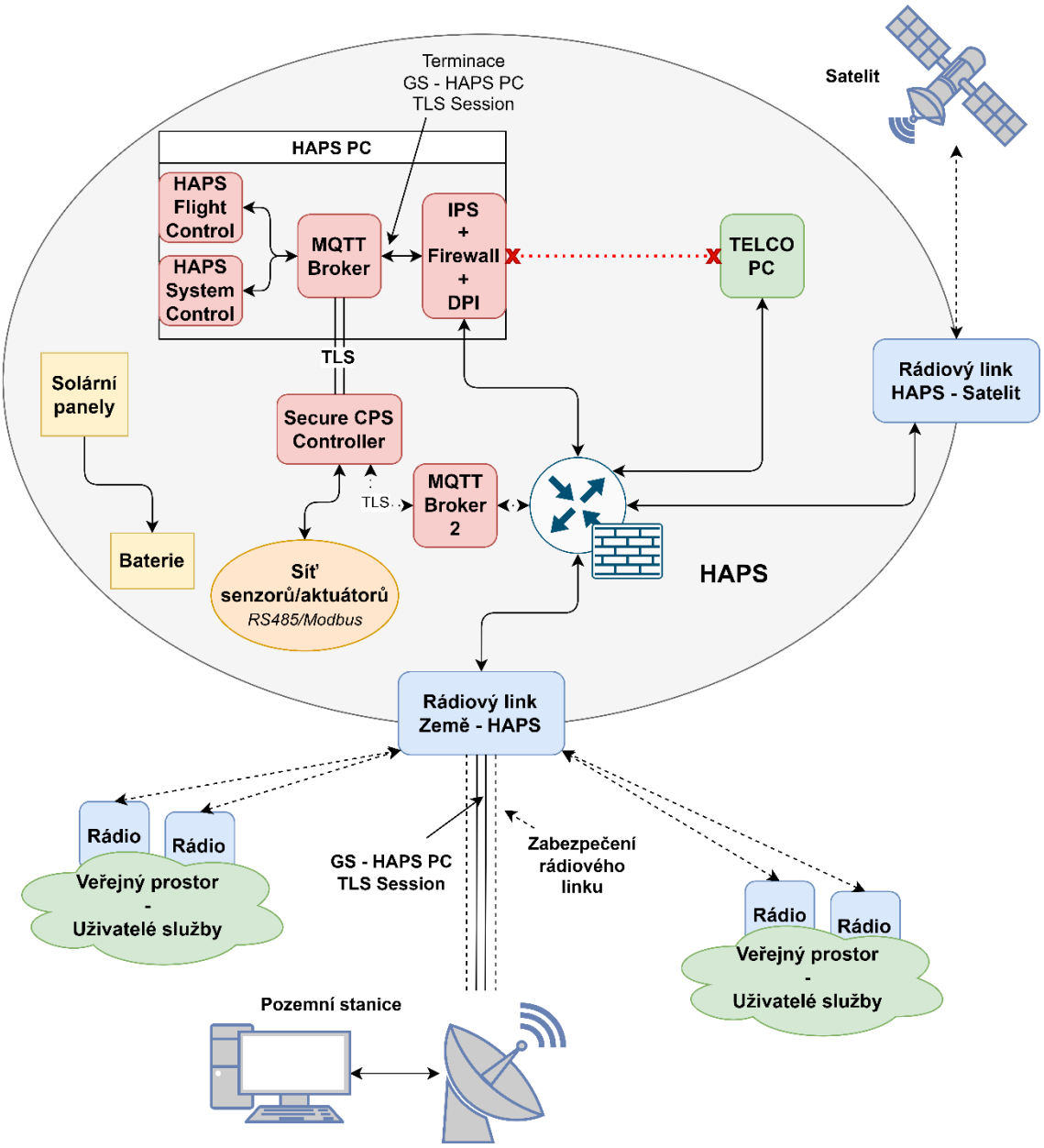
V tabulce níže je pro přehlednost uvedeno shrnutí navržených bezpečnostních opatření pro konkrétní hrozby. Navrženými bezpečnostními opatřeními bylo mitigováno 30 hrozeb z celkem 53 identifikovaných hrozeb. Všechny hrozby s nejzávažnějším stupněm rizika (high), byly mitigovány pomocí navržených opatření.

Tab. 6.1 *Souhrn navržených bezpečnostních opatření*

Hrozba	Navržený způsob mitigace hrozby
• SCC – 3, 5, 7, 9 • HFC – 1, 2, 3 • HSC – 1, 2, 3	→ TLS 1.3 uvnitř sítě HAPSu (mezi SCC a HAPS PC, resp. MQTT Broker)
• HSC – 7, 8	→ Digitální podpis (SCC) senzorických dat pro GS
• MQTT Broker SPOF	→ Záložní MQTT Broker
• GSPC – 6, 12 • Útoky na HAPS PC	→ Kombinovaný IPS na HAPS PC (ML + Rule based)
• HFC – 7, 8, 9 • HSC – 7, 8 ,9 • GSPC – 1, 2, 4, 7, 8, 10	→ Zabezpečení rádiového linku (WPA2-AES) → TLS 1.3 mezi GSPC a HAPS PC (resp. MQTT Broker)
• HFC – 10 • GSPC – 5, 11 • HSC – 10	→ Opatření proti rušení (zabudované v modulu pro rádiovou komunikaci) → Záložní link přes satelitní spojení

6.3 Finální podoba zabezpečené architektury systému HAPS

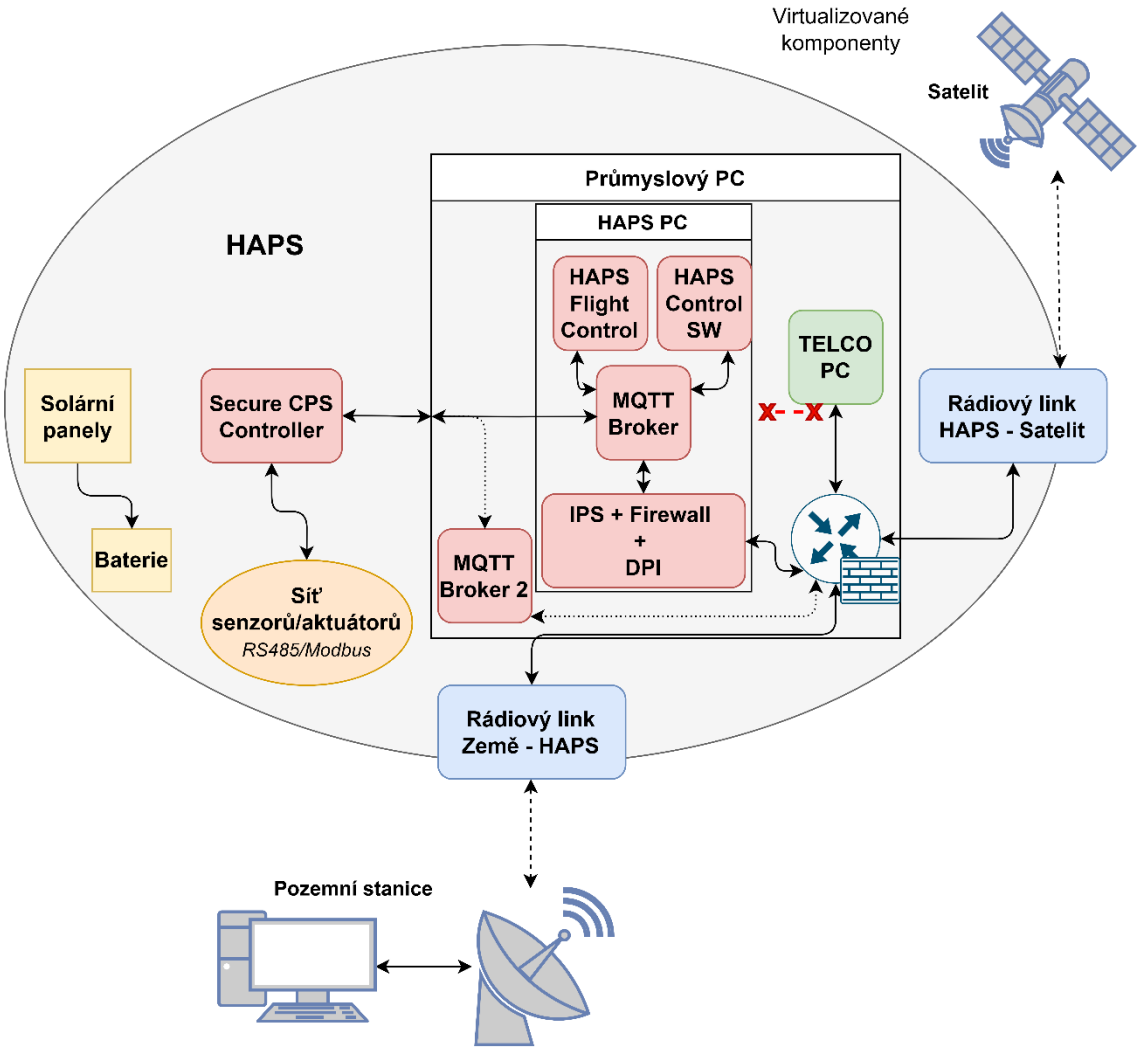
Architektura HAPS navržená v kapitole 4 (viz *obr. 4.2*) byla doplněna o bezpečnostní opatření vyplývající z analýzy hrozeb v předchozích částech. Logická architektura je vyobrazena na *obr. 6.1*.



Obr. 6.1 *Zabezpečená architektura HAPS – Logické schéma*

Na *obr. 6.1* je doplněno znázornění zabezpečení komunikace pomocí TLS, a to jak uvnitř HAPS PC, tak mezi GS a HAPS. V HAPS PC byl doplněn modul „IPS + Firewall + DPI“. Dále přibyl záložní MQTT Broker, označen jako „MQTT Broker 2“. Tento broker ale není za běžného provozu využíván, což je ve schématu znázorněno tečkovaným spojením.

Fyzické uspořádání této zabezpečené architektury je vyobrazeno na obrázku níže. I zde jsou stále všechny komponenty kromě SCC virtualizovány uvnitř průmyslového PC. Přestože je v logické architektuře SCC připojeno ke dvěma brokerům zároveň, ve fyzické architektuře je možné redukovat toto spojení na jeden fyzický kabel díky virtualizaci rozhraní a komponent.



Obr. 6.2 *Zabezpečená architektura HAPS – Fyzické schéma*

Kapitola 7
Návrh komponenty SCC

Koncept SCC byl představen v podkapitolách 4.3.3 a 4.3.4. Tato kapitola je věnována návrhu komponenty SCC. V návrhu zabezpečené architektury HAPS hraje SCC zásadní roli. Jak již bylo řečeno, hlavním úkolem SCC je omezit možnosti útočníka při kompromitaci hlavního počítače (HAPS PC), či pozemní kontrolní stanice. Díky známé množině bezpečných stavů SCC nepřipustí, aby se HAPS nacházel mimo tuto bezpečnou množinu. Vzhledem k takto omezeným možnostem útočníka budou ochráněny aktuátory HAPSu před úmyslným poškozením a HAPSem bude možné manipulovat pouze v rámci definovaných limitů. Úkolem SCC je také analyzovat senzorická data a generovat upozornění při detekci problému. V neposlední řadě je v SCC zabudován jakýsi nouzový režim, který při selhání všech ostatních systémů a komunikace s pozemní kontrolní stanicí zahájí sekvenci nouzového bezpečného přistání.

Nejdříve jsou v podkapitole 7.1 definovány požadavky na SCC. V další části jsou představeny dva navržené způsoby řízení letu HAPSu, u kterých jsou představeny výhody a nevýhody každého z nich. V podkapitole 7.3 je představena hlavní myšlenka CPS IPS stejně jako některé návrhy na její realizaci. Nakonec je představena navržená softwarová architektura komponenty SCC.

7.1 Požadované vlastnosti SCC

Požadované vlastnosti SCC lze rozdělit do dvou skupin – funkční a bezpečnostní. Funkční požadavky specifikují, jaké funkce má být SCC schopno vykonávat a jakými funkčními vlastnostmi má SCC disponovat. Požadavky bezpečnostního charakteru poté popisují požadované bezpečnostní vlastnosti SCC.

7.1.1 Funkční požadavky

Níže je uveden seznam navržených funkčních požadavků.

- Připojení senzorů/aktuátorů
  - Hardwarová podpora běžně používaných sběrnic (I2C, RS-485, CAN, 1-Wire atd.)
  - Jednotné programové rozhraní pro obsluhu senzorů/aktuátorů
- Zajištění komunikace s HAPS PC, resp. s pozemní stanicí
  - Podpora protokolu MQTT + TLS 1.3
  - Definovaný/é formát/y požadavků/odpovědí s využitím formátu JSON
- Zpracování příchozích požadavků na ovládání aktuátorů a čtení senzorů
  - Převést požadavek HAPS PC, či GSPC na vykonání konkrétní události na požadovaný úkon pomocí programového rozhraní pro obsluhu senzorů/aktuátorů
  - Vysílat odpovědi na tyto požadavky, např. potvrzení vykonání požadovaného úkonu

- Znalost o množině bezpečných stavů HAPSu
    - SCC má znalost o fyzických limitech všech aktuátorů a senzorů (max. otáčky motorů, max. frekvence čtení senzorů, atd.)
    - SCC má znalost o provozních limitech HAPSu jako celku (GPS souřadnice, výška, maximální rychlost, atd.)
    - SCC má znalost o správných a limitních hodnotách všech veličin monitorovaných pomocí senzorů
- Filtrace příchozích požadavků na SCC
    - Musí být posouzena bezpečnost každého příchozího požadavku
    - Požadavky, které by uvedly HAPS a jeho součásti mimo množinu bezpečných stavů budou zamítnuty, bude vygenerováno upozornění
    - Požadavky, které neporuší žádné z nastavených pravidel budou vykonány a žadatel bude o této skutečnosti informován
- Analýza senzorických dat
    - Získaná senzorická data budou podrobena analýze
    - Analýza senzorických dat má za úkol odhalit, zda není některá měřená veličina mimo bezpečný rozsah – pokud tomu tak je, bude vygenerováno upozornění
    - K analýze dat může být využité prahování jednotlivých veličin, je ale možné využít i složitější algoritmy k detekci anomálií, či algoritmy na bázi strojového učení – záleží na charakteru dat, požadovaném cíli analýzy i její rychlosti a výpočetní náročnosti
- Generování odpovědí a upozornění
    - SCC je schopné vygenerovat odpověď na příchozí požadavek
    - SCC je schopné vygenerovat upozornění v deklarovaných případech
- Digitální podpis vybraných dat
    - Digitální podpis vygenerovaných upozornění
    - Digitální podpis senzorických dat
- Režim nouzového řízení HAPSu
    - V případě selhání všech ostatních možností řízení HAPSu (autonomní řízení skrze HAPS PC, manuální řízení z pozemní stanice) je SCC schopno zahájit režim nouzového řízení HAPSu
    - Režim nouzového řízení umožňuje bezpečné autonomní přistání na specifikovanou lokaci

7.1.2 Bezpečnostní požadavky

Dále jsou definovány bezpečnostní požadavky na SCC, které jsou spíše obecného charakteru. Jejich realizace je otevřena interpretaci při softwarové realizaci jednotlivých částí SCC.

- Jednoduchost implementovaných aplikací
  - Díky programové jednoduchosti, zejména z hlediska vnější síťové interakce, je možné minimalizovat „attack surface“, čímž budou i minimalizovány možnosti útočníka
- Čistý, jednoduchý a přehledný kód
  - Kód, který je dobře auditovatelný bude potenciálně obsahovat méně zranitelností již ve fázi vývoje
- Penetrační testování
  - SCC musí být podrobeno důkladnému penetračnímu testování pro odhalení zranitelností
- Kontrola integrity zdrojového kódu / binárního souboru
  - Před spuštěním programu SCC musí být provedena kontrola integrity pro zjištění, zda nedošlo k jeho modifikaci
- Bezpečná konfigurace OS
  - OS, na jehož základě bude SCC postaveno, musí být bezpečně nakonfigurováno
  - Důležitý je i výběr OS – kladen důraz zejména na stabilitu systému a nízké výpočetní nároky
- Vhodný výběr HW
  - HW SCC musí být vhodně vybrán s ohledem na provozní podmínky (nízké teploty, zvýšené kosmické záření, vlhkost, atd.)
  - Vybraný HW musí odpovídat striktním spolehlivostním nárokům
  - Vybraný HW musí disponovat dostatečným výpočetním výkonem
  - HW SCC musí zohledňovat energetické možnosti HAPSu – kladen důraz na nízkou spotřebu
- Ochrana proti fyzickému poškození
  - SCC musí být chráněno proti fyzickému poškození při nečekaných událostech (náraz při dopadu, střetnutí s jiným tělesem, atd.)
  - Senzory/aktuátory musí být připojeny dostatečně robustním způsobem

7.2 Řízení letu HAPS

V této podkapitole jsou představeny dva přístupy k řízení letu HAPSu. Z těchto dvou odlišných přístupů vyplývá i odlišná struktura požadavků na SCC a následný rozdílný způsob zpracování těchto požadavků. Oba tyto přístupy přinášejí své klady i zápory. Z tohoto důvodu je předložen návrh využití vhodné kombinace obou přístupů k řízení letu HAPSu, která je schopná potlačit nevýhody využití pouze jednoho z nich.

7.2.1 Vysokoúrovňový přístup k řízení HAPSu

Prvním přístupem k řízení letu HAPSu je tzv. vysokoúrovňový přístup. Obsahem požadavku na SCC je v tomto případě vysokoúrovňový příkaz požadující vykonání komplexní činnosti skládající se ze sekvence dílčích událostí. Příkladem může být požadavek na přesun HAPSu na určité GPS souřadnice, požadavek na vzlétnutí či na přistání. Ve všech těchto případech vyžaduje takový úkon spuštění sekvence událostí, jako např. aktivace různých aktuátorů při čtení zpětné vazby ze senzorických dat. Zpracování těchto vysokoúrovňových požadavků musí být již součástí SCC. Jedná se tedy o určité předem definované sekvence.

Jelikož v případě vysokoúrovňového přístupu nedochází k přímému ovládání aktuátorů, je do určité míry zajištěno zachování bezpečných stavů HAPSu. To je dáno zejména tím, že předem naprogramované sekvence jsou již ze své podstaty bezpečně vykonány. Přesto se nabízejí možné útoky, které je potřeba ošetřit. Pokud má útočník možnost využít těchto vysokoúrovňových požadavků, může například neustále vysílat požadavky na přesun z jednoho místa na druhé a zpět. Tímto by došlo k rychlejšímu vybití baterie a ohrožení správné funkčnosti HAPSu. V SCC je nutné ošetřit tyto případy vhodným algoritmem.

Zásadní výhodou tohoto přístupu je jeho jednoduchost. Složité řízení jednotlivých aktuátorů je redukováno na jeden požadavek komplexního charakteru. Díky této jednoduchosti je zjednodušena i kontrola bezpečnosti příchozích požadavků. Některé vysokoúrovňové požadavky mohou mít pouze jednoduché argumenty – například přesun na souřadnice X, Y, Z. Kontrola důsledků vykonání takových požadavků je pak triviální, tyto případy stačí ošetřit podmínkou po vymezení povoleného okruhu souřadnic.

Zjevnou nevýhodou tohoto přístupu jsou omezené možnosti kontroly HAPSu. Pokud bude požadováno vykonání úkonu, který nebude předem naprogramován, ztrácí se možnost tento úkon vykonat. Pokud nastane specifická situace, se kterou nebylo při návrhu počítáno, může být řešení obtížné díky absenci jemné kontroly nad jednotlivými aktuátory. Pokud například nastane porucha na některém z aktuátorů, může být žádoucí přistoupit k jednotlivým aktuátorům ve snaze o kompenzaci nastalé poruchy. Tento problém je možné překonat složitějšími řídícími algoritmy na straně SCC. To by ale vyžadovalo složitější vývoj SCC a je to v rozporu s filozofií návrhu SCC.

7.2.2 Nízkoúrovňový přístup k řízení HAPSu

Druhým přístupem k řízení letu HAPSu je přístup nízkoúrovňový. Nízkoúrovňový přístup umožňuje kontrolu jednotlivých aktuátorů. Obsah požadavku by tedy musel specifikovat konkrétní aktuátor a konkrétní akci, kterou musí SCC s daným aktuátorem v daný moment vykonat. Logika řízení je v tomto případě plně v gesci procesu HFC na HAPS PC. Díky tomu není po SCC vyžadováno složitějších operací v rámci řízení HAPSu a je oproti vysokoúrovňovému přístupu redukován na pouhý kontrolér bez hlubší řídící logiky.

Hlavní výhodou tohoto přístupu je možnost plné kontroly nad aktuátory HAPSu a tedy i nad jeho letem. Celý řídící algoritmus je centralizován na HAPS PC. V případě nečekaných událostí je možné jemně kontrolovat jednotlivé aktuátory a systém není omezen pouze na předem naprogramované sekvence, které nemusí správně fungovat v každé situaci.

Jako nevýhoda se může jevit nutnost složitějšího řídícího algoritmu na HAPS PC. Nicméně v případě vysokoúrovňového přístupu je nutná složitost pouze distribuována na dvě různá zařízení. Další diskutabilní nevýhodou je složitost vyhodnocení bezpečnosti požadavků. V případě vysokoúrovňových požadavků je snadné posoudit dopady vykonání takového požadavku. U nízkoúrovňových požadavků toto vyhodnocení nemusí být takto přímočaré. Triviální případy, jako například kontrola limitních otáček motorů, jsou jednoduše ošetřeny podmínkou. Nicméně celý algoritmus vyhodnocení bezpečnosti požadavků musí být schopen vnímat všechny aktuátory jako jeden systém, jehož jednotlivé části (aktuátory) mezi sebou mají vazby. Příkladem takového problému může být zapnutí dvou různých motorů, jejichž účinky se navzájem vyruší. HAPS tedy zůstává ve stacionárním stavu, otáčky motorů jsou v bezpečných mezích, přesto zde ale dochází k útoku na baterii systému (motory baterii rychleji vyčerpají).

7.2.3 Kombinace obou přístupů

Výhody i nevýhody použití jednotlivých přístupů jsou příliš velké na to, aby bylo možné vybrat pouze jeden z nich. Vhodným řešením tohoto problému je tedy adekvátní kombinace obou představených přístupů k řízení letu HAPSu.

Adekvátní kombinací můžeme rozumět využití vhodnějšího přístupu pro danou situaci. Pokud to situace umožňuje, je značně jednodušší využít vysokoúrovňový přístup. Tento přístup by tedy měl být ve standardních situacích využíván přednostně. V situacích, kdy již možnosti vysokoúrovňového přístupu nejsou dostatečné, bude využit přístup nízkoúrovňový. Díky této kombinaci je možné zachovat jednoduchost řídícího algoritmu na SCC, protože stačí předem naprogramovat pouze základní sekvence. Je zde ale nutnost složitějšího řídícího algoritmu HFC na HAPS PC, protože ten musí být schopen řešit nečekané situace pomocí nízkoúrovňového přístupu. Řešení poruchových a různých jiných nečekaných situací je ale vnímáno jako nutnost.

Díky využití obou přístupů bude muset být složitější i algoritmus na vyhodnocování bezpečnosti požadavků, zejména díky nutnosti komplexního vnímání systému aktuátorů, jak již bylo vysvětleno v předchozí podkapitole.

7.3 Návrh CPS IPS pro systémy HAPS

Termín CPS IPS je v kontextu této práce použit pro algoritmy zodpovědné za vyhodnocení bezpečnosti příchozích požadavků a algoritmy pro analýzu měřených senzorických dat. Navržený systém CPS IPS využívá seznam definovaných pravidel pro filtrování obsahu. Z technického hlediska lze CPS IPS zařadit do tzv. „rule based“ IPS, tj. IPS systém založený na pravidlech.

7.3.1 Filtr příchozích požadavků

Navržený filtr příchozích požadavků lze rozdělit na tři dílčí části – časový filtr, filtr obsahu požadavků a kontextuální filtr. Níže je popsána funkcionalita těchto tří komponent.

Časový filtr frekvence příchozích požadavků

Úkolem tohoto filtru je zajistit dodržení limitu frekvence vysílání požadavků. Pokud by požadavky byly vykonávány v rychlosti odpovídající jejich příchodu, mohlo by dojít k poškození kontrolovaných aktuátorů. Požadavky totiž mohou přicházet daleko rychleji, než je aktuátor může vykonat, či může být pro daný aktuátor příliš rychlá změna stavu nebezpečná nebo může jinak ohrozit jeho životnost.

Pro každý ovládaný aktuátor je tedy nutné implementovat separátní časový filtr, který zajistí bezpečnou frekvenci vykonání daného požadavku tak, aby nedošlo k nadlimitní zátěži pro aktuátor. Časový filtr by měl být pro dané požadavky nastaven tak, aby při standardním provozu nijak nezasahoval do řízení.

Technická realizace časového filtru může být v podobě prostého časovače, kde každý aktuátor bude mít svůj vlastní časovač. SCC si pamatuje, kdy byl naposledy vykonán příkaz stejného typu. Uvažujme následující příklad příchozího požadavku.

Na MQTT téma „**control/actuator278**“ dorazí zpráva s obsahem:

```
{
"enable": True,
"power": 75
}
```

Tento požadavek obsahuje dva druhy příkazů. Nejprve je zde příkaz *enable*, kterým lze daný aktuátor zapnout, či vypnout. Dále je ve zprávě obsažen příkaz *power*, kterým je možné ovládat procentuální výkon daného aktuátoru. Pokud je tento požadavek vykonán, SCC začne čítat dobu *timeout*. Do uplynutí doby *timetout* budou jakékoliv požadavky na aktuátor „278“ (dle MQTT tématu) s příkazy stejných typů zamítnuty. V případě zamítnutí požadavku bude vygenerováno upozornění. Po uplynutí této doby již může být požadavek opět bezpečně vykonán.

Filtr obsahu příchozích požadavků

Tato druhá komponenta filtru příchozích požadavků se soustředí na samotný obsah požadavků. Kontroluje tedy, co chce příchozí požadavek vykonat. V tomto algoritmu je nutné specifikovat limitní stavy všech ovládaných aktuátorů, které jsou pro dané aktuátory bezpečné. Každý aktuátor s každým definovaným typem příkazu má určité limity, které musí být nastaveny programátory HAPSu. Pokud přijde požadavek k uvedení aktuátoru, či systému do stavu, který by definované meze překračoval, bude požadavek zamítnut. Tuto filtraci je možné provádět jednoduchým prahováním.

Nyní uvažujme vysokoúrovňový požadavek na změnu pozice HAPSu. Pro tento zjednodušený modelový příklad definujme, že množina povolených souřadnic HAPSu pro stacionární provoz je pro jednotlivé osy:

X: [-100, 100]; Y: [-100, 100]; Z: [19500; 20500].

Dále uvažujme, že HAPS se aktuálně nachází na souřadnicích {x: 25; y: -30; z: 20000} a uvažujme následující příchozí požadavek, který přišel na MQTT téma „**control/flight**“:

```
{
"moveX": -25,
"moveY": 175
}
```

Souřadnice na ose *x* by se po vykonání příkazu změnila na -25, což je v povolené množině pro osu *x*. Souřadnice na ose *y* by se ale změnila na 175, což je mimo povolenou množinu souřadnic. Tento požadavek tedy musí být zamítnut.

Kontextuální filtr příchozích požadavků

Jak již bylo popsáno v podkapitole 7.2.2, je nutné brát při filtraci požadavků v potaz i kontext celého systému, nikoliv hledět pouze na samostatné aktuátory. Kontextem systému je v tomto případě chápána znalost aktuálního stavu systému. V případě HAPSu existují situace, kdy aktivace některého aktuátoru může být problematická z důvodu toho, že je již jiný aktuátor v chodu. To odpovídá již zmíněnému příkladu zapnutí dvou motorů s protilehlými efekty, kdy například jeden z motorů slouží k pohybu doleva a druhý k pohybu doprava. Současné využití těchto motorů nedává smysl a může být případně útočníkem zneužito k vybití baterie. Tyto situace by měl ošetřit právě kontextuální filtr.

Realizace spočívá v definici vzájemných vazeb, které chceme zakázat. Vazeb, které chceme zakázat, bude méně, než vazeb povolených.

Příklad může být následující situace. V seznamu zakázaných vazeb je definováno, že aktuátor „123“ a aktuátor „278“ nemohou být v aktivním stavu oba zároveň. Uvažujme, že aktuátor „123“ již v aktivním stavu je. Pokud na MQTT téma „**control/actuator278**“ přijde požadavek na jeho zapnutí, bude takový požadavek zamítnut z kontextuálních důvodů.

7.3.2 Analýza senzorických dat

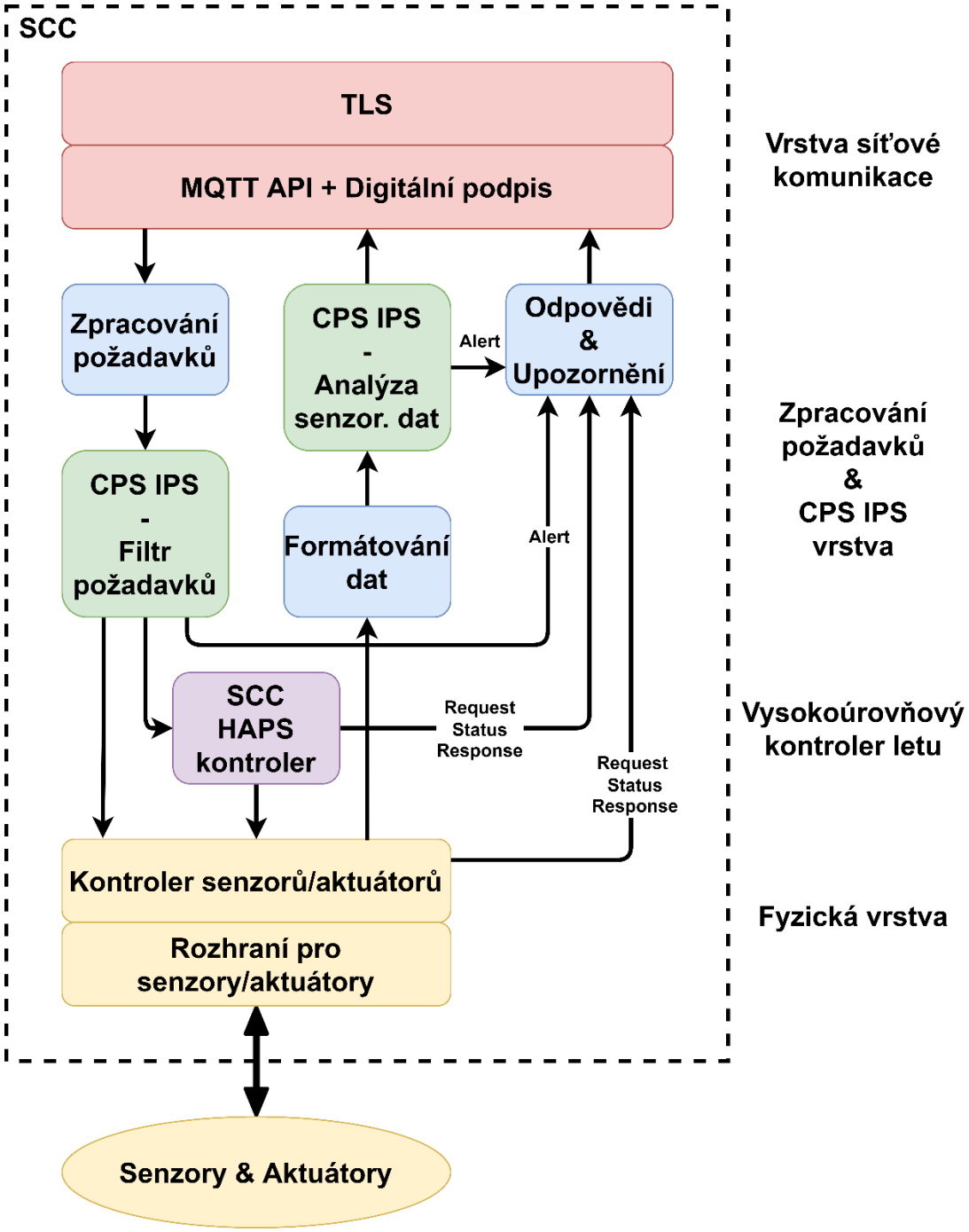
Analýza senzorických dat slouží zejména k monitorování stavu HAPSu a k detekci případných vzniklých problémů. Pokud dojde k detekci anomálie v hodnotách senzorických dat, vygeneruje se upozornění, které bude k dispozici řízení provozu na pozemní kontrolní stanici. Na základě těchto upozornění mohou být provedena opatření pro zabezpečení provozu HAPSu, či může být HAPS z bezpečnostních důvodů stažen z aktivního provozu a přivolán zpět na zem.

K analýze senzorických dat se nabízí vícero přístupů. Nejjednodušším přístupem je prahování. U prahování je nastaven pevný limit hodnoty měřené veličiny, jehož překročení je monitorováno. Výhodou tohoto přístupu je jeho jednoduchost a výpočetní nenáročnost. Zároveň lze stále odhalit poruchové stavy a hodnoty, které jsou mimo přijatelné meze. Tato metoda je tedy vhodná k nasazení v CPS IPS na SCC.

Další možností při analýze senzorických dat jsou prediktivní analýzy. Na základě prediktivních analýz je možné předpovídat poruchy a zajistit tak bezpečný a bezporuchový let HAPSu. Využití prediktivních analýz senzorických dat je tedy zásadní. U prediktivních analýz se ale nejedná o akutní analýzu, která musí být prováděna okamžitě v reálném čase. Vzhledem k výpočetní náročnosti některých metod pro prediktivní analýzu a s ohledem na fakt, že SCC je baterií napájené zařízení, je vhodnější nasadit prediktivní analýzu na pozemní stanici. Možné metody prediktivní analýzy jsou k dispozici v [53], [54] a [55].

7.4 Architektura SCC

Na *obr. 7.1* je vyobrazená navržená architektura SCC. Navržená architektura se skládá celkem ze čtyř úrovní. První úrovní je fyzická vrstva, jejímž úkolem je interakce se senzory a aktuátory. Druhou úrovní je vysokoúrovňový kontrolér letu, který slouží pro zpracování vysokoúrovňových požadavků. Třetí úroveň slouží ke zpracování požadavků, generaci odpovědí a upozornění a také zde probíhá filtrace příchozích požadavků a analýza senzorických dat v rámci CPS IPS. Poslední úrovní je vrstva síťové komunikace, která je zodpovědná za zabezpečenou komunikaci pomocí protokolu MQTT.



Obr. 7.1 *Navržená architektura komponenty SCC*

7.4.1 Fyzická vrstva

Tato úroveň se skládá ze dvou bloků, kterými jsou kontrolér senzorů/aktuátorů a rozhraní pro senzory a aktuátory.

Rozhraní pro senzory/aktuátory

- Fyzické rozhraní pro připojení senzorů, či aktuátorů
    - Potřebná rozhraní záleží na použitých typech senzorů/aktuátorů

Kontrolér senzorů/aktuátorů

- Softwarová komponenta
    - Část programu zodpovědná za přímou interakci se senzory a aktuátory
    - Komunikuje se senzory a aktuátory skrze fyzické rozhraní
- Poskytuje „high level“ programové rozhraní pro interakci se senzory a aktuátory
    - Pomocí tohoto programového rozhraní lze realizovat „low level“ příkazy

7.4.2 Vysokoúrovňový kontrolér letu

Tato komponenta je již čistě softwarovým prvkem. Níže je popsána její funkcionalita.

SCC HAPS kontrolér

- Hlavním úkolem je interpretovat vysokoúrovňové požadavky
    - Využívá předem naprogramované sekvence
    - Vysokoúrovňové požadavky jsou převedeny na předem naprogramované sekvence
    - Implementace částečně autonomního řízení
        - Například sekvenci vzlétnutí mohou provázet povětrnostní podmínky, které nelze předvídat
        - Autonomní vzlétnutí v této komponentě inteligentně řídí předem naprogramované sekvence, algoritmus je schopen reagovat na vlivy prostředí a vhodně je kompenzovat
- Zajištění nouzového řízení HAPSu
    - Využito v případě nedostupnosti HAPS PC, dlouhodobé ztráty spojení s pozemní stanicí, selhání ostatních řídících systému, nedostatku energie pro provoz hlavního PC
    - Režim nouzového řízení HAPSu zajistí bezpečné autonomní přistání na specifikovanou lokaci

7.4.3 Zpracování požadavků a CPS IPS

Tato úroveň je klíčovou úrovní pro funkcionalitu SCC. Úroveň zpracování požadavků a CPS IPS je zodpovědná za bezpečnostní logiku SCC. Níže jsou popsány bloky, ze kterých se tato úroveň skládá.

Zpracování požadavků

- Přijímá požadavky v unifikovaném JSON formátu
- Určuje jaké funkci v CPS IPS předat přijatý požadavek podle jeho obsahu
    - De facto třídí příchozí požadavky a předává je dalším částem kódu

Formátování dat

- Převádí senzorická data na unifikovaný formát v JSONu a volá vhodnou funkci z CPS IPS

Odpovědi & Upozornění

- Blok generující odpovědi pro HAPS PC a GSPC
    - Pokud vyšlou požadavek, je nutné dát žadateli zpětnou vazbu o způsobu jeho zpracování
- CPS IPS dává v určitých případech pokyn tomuto bloku, že by mělo být vygenerováno upozornění
    - V případech, kdy CPS IPS detekuje porušení bezpečnostních pravidel
    - Tento blok na základě pokynů z CPS IPS generuje upozornění a předá jej komunikační vrstvě (MQTT API)

CPS IPS – Filtr požadavků

- Obsahuje seznam definovaných pravidel pro bezpečnost požadavků
- Množina různých funkcí, které zpracovávají různé druhy požadavků
- Vysokoúrovňové požadavky bez argumentů nebudou filtrovány (např. vzlet, přistání)
    - Budou filtrovány pouze vysokoúrovňové požadavky s argumenty (např. přesun na pozici X Y Z)
- Nízkoúrovňové požadavky jsou vždy filtrovány
- Více detailů v podkapitole 7.3.1

CPS IPS – Analýza senzorických dat

- Viz podkapitola 7.3.2

7.4.4 Vrstva síťové komunikace

MQTT API + Digitální podpis

- Tento blok slouží jako programové rozhraní pro posílání zpráv pomocí protokolu MQTT
    - Jsou specifikována různá MQTT témata
        - Každé MQTT téma odpovídá jednomu aktuátoru, senzoru, či vysokoúrovňové komponentě
- Digitální podpis slouží pro kryptografický digitální podpis dat, která tento podpis vyžadují
    - Takto upravená zpráva je následně poslána na stanovené MQTT téma

TLS

- Zajišťuje zabezpečení MQTT komunikace pomocí protokolu TLS

Kapitola 8
Implementace komponenty SCC

Tato kapitola je věnována popisu implementace komponenty SCC. Je popsán cíl vývoje, kterým je vytvořit „Proof of Concept“ (PoC) komponenty SCC. Dále je popsán hardware a software, který byl pro implementaci využit. Další podkapitoly jsou věnovány popisu implementace funkcí v jednotlivých dílčích částech.

8.1 Cíl vývoje

Hlavním cílem vývoje je implementovat komponentu SCC na úrovni PoC. Tato úroveň zahrnuje implementaci základní funkcionality komponenty SCC. Z důvodu, že plná implementace komponenty SCC je díky její složitosti rozsáhlým úkolem, je SCC realizováno právě na úrovni PoC, díky čemuž lze ověřit základní myšlenku a ověřit funkčnost a proveditelnost konceptu jako takového.

Požadavky na základní funkcionalitu SCC na úrovni PoC můžeme definovat následovně:

- Připojení senzorů/aktuátorů k SCC pomocí sběrnice odolné vůči rušení (např. RS-485)
- Komunikace s připojenými senzory/aktuátory
- Komunikace s jiným zařízením pomocí protokolu MQTT
- Zpracování příchozích požadavků z jiného zařízení
- CPS IPS – filtr příchozích požadavků
- Zpracování vysokoúrovňových požadavků – předem naprogramované sekvence kontroly letu

Jelikož je v kontextu HAPSu složité a nepraktické realizovat některé tyto prvky na reálných senzorech a aktuátorech, je vhodné nasimulovat potřebné prvky HAPSu. Z toho důvodu je potřeba vytvořit simulaci letu HAPSu. Součástí simulace je virtuální GPS senzor a sada virtuálních aktuátorů (motorů), díky kterým se simulovaný HAPS může pohybovat ve virtuálním prostoru. Na takové simulaci letu je vhodné demonstrovat bezpečnostní mechanismy SCC.

Součástí implementace PoC je i webové grafické rozhraní (GUI – Graphical User Interface), které umožní s SCC intuitivně komunikovat a vysílat požadavky, na které bude následně SCC reagovat. Toto webové GUI simuluje svojí funkčností HAPS PC, případně pozemní kontrolní stanici, a to v tom smyslu, že posílá požadavky na SCC. Nejedná se o implementaci HAPS PC, která je schopná autonomního řízení letu, pouze o způsob, kterým lze vhodně demonstrovat funkčnost SCC.

8.2 Použitý hardware a software

Z důvodu, že se nejedná o finální produkt vhodný k reálnému nasazení v HAPSu, byly použité hardwarové a softwarové komponenty voleny zejména s přihlédnutím na rychlost výsledného prototypování.

V následujících podkapitolách je představen použitý hardware a software pro implementaci dílčích součástí PoC SCC.

8.2.1 SCC

Základní vývojovou platformou pro SCC byla zvolena platforma Raspberry Pi 4 Model B 4GB, viz *obr. 8.1*. Raspberry Pi 4 bylo zvoleno pro jeho dostatečně vysoký výkon, možnost připojení několika USB zařízení a pro 1 Gbps Ethernet rozhraní. Díky těmto vlastnostem tak nedochází při implementaci SCC k omezení ze strany hardwaru. Raspberry Pi 4 bylo dále použito z důvodu, že se jedná o minipočítač, který váhou, velikostí, spotřebou energie a výpočetním výkonem odpovídá hardwaru, který by byl pravděpodobně použit i při realizaci reálného SCC vhodného k nasazení na HAPS.

Obr. 8.1 *Raspberry Pi 4 Model B 4GB* [56]

Jako operační systém byl zvolen Raspberry Pi OS 64-bit ve verzi bez grafického rozhraní. Jedná se o oficiální operační systém pro platformy Raspberry Pi. Vzhledem k tomu, že Raspberry Pi OS je portem linuxového operačního systému Debian, jedná se o velice nenáročný a stabilní operační systém.

Jako primární programovací jazyk pro vývoj hlavního programu SCC byl zvolen Python 3. Díky jeho jednoduchosti je vhodný k prototypování různých druhů produktů. Ze zásadních nezákladních knihoven lze zmínit knihovnu „Paho MQTT“ [57], která slouží jako MQTT klient a knihovnu „MinimalModbus“ [58], která slouží ke komunikaci pomocí protokolu Modbus.

8.2.2 Webové GUI

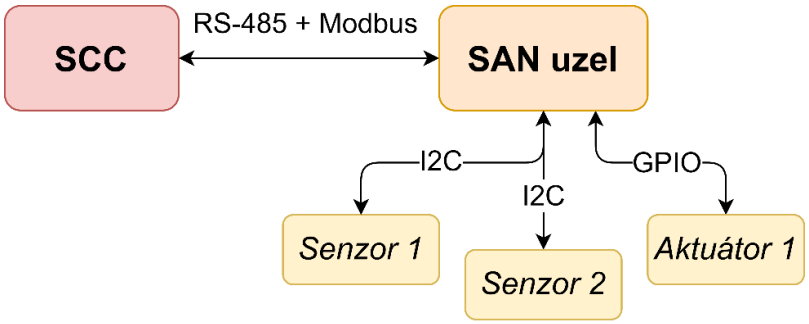
Základem pro webové GUI je virtualizovaný počítač s OS Ubuntu ve verzi bez grafického rozhraní. Na tomto virtualizovaném počítači je po vzoru teoretického návrhu HAPS PC z kapitoly 4 nainstalován MQTT Broker „Mosquitto“ [59].

K vytvoření webového GUI byl využit nástroj Node-RED [60]. Node-RED je programovací nástroj v prohlížeči vhodný k programování „event-driven“ aplikací. Node-RED byl následně doplněn o rozšíření „node-red-dasboard“ [61] pro usnadnění tvorby webového GUI. Zprávy přijaté pomocí protokolu MQTT jsou následně zpracovány pomocí Javascriptových funkcí přímo uvnitř Node-RED.

8.2.3 SAN uzel

SAN uzel zde figuruje jako jakýsi mezičlánek mezi SCC a senzory/aktuátory. Zvoleným způsobem komunikace mezi senzory/aktuátory a SCC se stala sběrnice RS-485 s využitím protokolu Modbus. Sériová sběrnice RS-485 byla využita díky vlastnostem zmíněných v podkapitole 2.2.2, zejména díky spolehlivosti a odolnosti vůči rušení. Takové vlastnosti jsou předpokládanými požadavky pro využití v HAPSu.

Jelikož jednoduché senzory a aktuátory nedisponují možnostmi pro komunikaci po sběrnici RS-485 s využitím protokolu Modbus, bylo nutné využít mikrokontroler (SAN uzel), který tyto možnosti má. K tomuto mikrokontroleru byly následně připojeny senzory/aktuátory, a to buď přes sběrnici I2C, či přímo na GPIO mikrokontroleru a data byla přeposílána na SCC s využitím sběrnice RS-485 + Modbus. Schéma tohoto zapojení je na *obr. 8.2*.



Obr. 8.2 *Model SAN uzlu*

Jako SAN uzel byl využit mikrokontroler Arduino UNO, viz *obr. 8.3*. Jedná se o 8-bit mikrokontroler s frekvencí procesoru 16 MHz. Disponuje 14 digitálními GPIO, z čehož 6 podporuje PWM a na pinech A4 a A5 se nacházejí vodiče SDA a SCL pro možnost využití sběrnice I2C. Pro platformu Arduino také existuje mnoho knihoven a je zde rozsáhlá a aktivní podpora komunity.

K naprogramování firmwaru byly využity knihovny „SimpleModbusSlave“ [62], „Adafruit_HTU21DF“ [63] a „BH1750“ [64].

Obr. 8.3 *Arduino UNO* [65]

Pro komunikaci po sběrnici RS-485 bylo zapotřebí využít na straně Arduina převodník TTL na RS-485 s čipem MAX485. Využitý převodník je na *obr. 8.4*. Jelikož ani využité Raspberry Pi 4 nedisponuje možností komunikace po sběrnici RS-485, bylo nutné využít na straně Raspberry Pi 4 převodník RS-485 na USB s čipem CH340C. Tento převodník je na *obr. 8.5*.

Obr. 8.4 *Převodník TTL na RS-485* [66]

Obr. 8.5 *Převodník USB na RS-485* [67]

Pro demonstraci schopnosti číst a posílat senzorická data a ovládat aktuátory byly vybrány dva senzory a jeden aktuátor. Prvním senzorem je HTU21D (*obr. 8.6*), který slouží jako digitální vlhkoměr a teploměr. Druhým senzorem je BH1750 (*obr. 8.7)*, což je senzor intenzity světla. Oba tyto senzory byly k SAN uzlu připojeny pomocí sběrnice I2C. Jako binární aktuátor (stavy ON/OFF) bylo využito relé, které bylo připojeno přímo na digitální GPIO výstup mikrokontroleru.

Obr. 8.6 *HTU21D* [68]

Obr. 8.7 *BH1750* [69]

8.3 Implementace aplikace SCC

Cílem vývoje bylo naprogramovat aplikaci, která by vyhovovala požadavkům definovaným v podkapitole 8.1. V tabulce níže je uvedena použitá verze jazyka Python i využité knihovny, včetně jejich verzí použitých v době vývoje aplikace. Standardní knihovny nejsou v tabulce zahrnuty.

Tab. 8.1 *Verze použitých knihoven*

Software	Verze
Python	3.9.2
Paho-MQTT	1.6.1
MinimalModbus	2.0.1
PyYAML	6.0
Munch	2.5.0

V dalších částech je popsána funkčnost jednotlivých částí aplikace.

8.3.1 Popis funkce hlavního bloku kódu

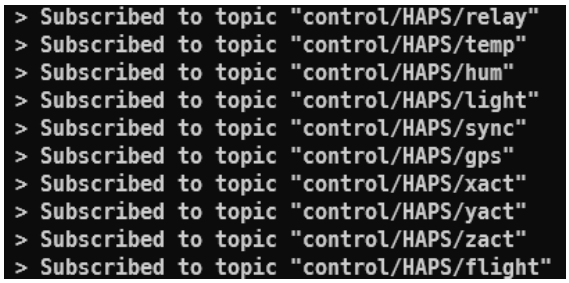
Aplikace je spouštěna z příkazového řádku na SCC. Pro spuštění je možné využít příkaz `python3 main.py` ze složky *src*, ve které se soubor *main.py* nachází anebo je možné využít Bash skript *start.sh*, který slouží ke spuštění aplikace na pozadí.

Po spuštění aplikace se nejprve inicializují globální objekty se kterými aplikace nadále pracuje. Prvním inicializovaným objektem je objekt třídy `Mqtt`. Díky jeho konstruktoru se aplikace ihned pokouší připojit k MQTT Brokeru. Na *obr. 8.8* je vyobrazeno spuštění aplikace v příkazovém řádku a následné úspěšné připojení k MQTT Brokeru.

IP adresa a port MQTT Brokeru jsou definovány v souboru *config.py*, který slouží k uložení různých konfiguračních proměnných. Kromě již zmíněné IP adresy a portu MQTT Brokeru zde lze najít názvy používaných MQTT témat, cestu v souborovém systému k souborům, do kterých jsou logována data a název fyzického portu pro převodník RS-485 na USB.

Obr. 8.8 *Spuštění aplikace a úspěšné připojení k MQTT Brokeru*

Po úspěšném připojení dojde k přihlášení k MQTT tématům, která používá web GUI k vysílání požadavků na SCC. Tato témata mají strukturu `control/HAPS/<název_zařízení>`, viz *obr. 8.9*.



Obr. 8.9 *Přihlášení ke kontrolním MQTT tématům – výpis z příkazového řádku*

Dále dojde v aplikaci k synchronizaci vnitřního stavu aplikace SCC a stavu webového GUI. To je nutné z důvodu, že ve chvíli kdy je ukončen jeden z těchto uzlů (SCC nebo web GUI) a poté opět spuštěn, dojde k desynchronizaci celého systému, kde vnitřní stav aplikace SCC neodpovídá stavu zobrazeném na webovém GUI. Tento problém je vyřešen zasláním žádosti o synchronizaci při spuštění aplikace SCC, resp. webového GUI. Tato žádost spočívá ve vyslání zprávy „SYNC“ na příslušné MQTT téma.

```
    # sync with GUI
    print("> Syncing with server GUI...")
    mqtt.publish(conf.sync_pub_topic, "SYNC")
```

Po přijetí této zprávy druhou stranou, zašle tato strana aktualizace všech svých stavů straně která o synchronizaci žádá.

Dále vstupuje kód do nekonečné smyčky *while*, ve které setrvává do ukončení programu. V této hlavní smyčce se opakovaně čtou senzorická data, pokud je jejich čtení vyžádáno z webového GUI.

```
while True:
        try:
            if SANNode.enable_temp == True:
                temp_status = SANNode.readTemperature()
                if temp_status == True:
                    sendData.temperature(SANNode, mqtt)

            if SANNode.enable_hum == True:
                hum_status = SANNode.readHumidity()
                if hum_status == True:
                    sendData.humidity(SANNode, mqtt)

            if SANNode.enable_light == True:
                light_status = SANNode.readLightLevel()
                if light_status == True:
                    sendData.light(SANNode, mqtt)
```

Objekt `SANNode` odpovídá SAN uzlu z podkapitoly 8.2.3. Jako takový reprezentuje připojené senzory a aktuátory. Tento objekt také obsahuje metody pro čtení dat z připojených senzorů, konkrétně metody `readTemperature()`, `readHumidity()` a `readLightLevel()`. Metody objektu `SANNode` využívají přímou komunikaci pomocí protokolu Modbus se SAN uzlem, ke kterému jsou tyto senzory připojené. V kontextu teoretického návrhu SCC se tedy jedná o kontrolér senzorů/aktuátorů definovaný v podkapitole 7.4.1.

Objekt `sendData` je v celé aplikaci využíván pro pohodlné zaslání potřebných dat webovému GUI. Obsahuje situačně specifické metody, které automaticky získají potřebná data, vhodně je formátují a odešlou na potřebné MQTT téma.

Dále se ve smyčce *while* vykonává několik metod. Metoda `runSimulation()` objektu `sim` je zodpovědná za běh simulace letu HAPSu. Metoda `run()` objektu `haps` je zodpovědná za řízení letu při přijetí vysokoúrovňových požadavků z webového GUI. Funkce `correctGPSposition()` je součástí CPS IPS a zajišťuje udržení simulovaného HAPSu v definovaném povoleném rozsahu GPS souřadnic.

Ve smyčce *while* je zahrnuta také metoda `sleep()`, která uspí běh programu po dobu 0,01 s. Díky tomu dojde k redukci zátěže CPU.

```
            # Run flight simulation
            sim.runSimulation(sendData, mqtt)

            # Run HAPS Controller
            haps.run(mqtt, sim)

            # CPS IPS -> protecting correct GPS coords
            correctGPSposition(sim, haps, mqtt)

            # time.sleep() in while loop to reduce CPU load
            time.sleep(0.01)
```

8.3.2 Zpracování příchozích požadavků

Součástí hlavního běhu programu je funkce `onMessage()`, která slouží jako tzv. „callback“ při přijetí MQTT zprávy z webového GUI - tedy funkce, která se vykoná při přijetí MQTT zprávy. V této funkci dochází ke zpracování příchozí zprávy a následně je zpráva předána funkci `handler()`. Tato funkce třídí příchozí zprávy dle MQTT témat a předává je příslušným částem kódu. Níže je část definice této funkce.

```
def handler(topic, msg, mqtt, SANNode, sim, haps):
    if topic == conf.relay_sub_topic:
        handleRelay(msg, mqtt, SANNode)
    elif topic == conf.temperature_sub_topic:
        handleTemperature(msg, mqtt, SANNode)
    elif topic == conf.humidity_sub_topic:
        handleHumidity(msg, mqtt, SANNode)
    .
    .
    .
```

Další „handle“ funkce poté mají za úkol vykonat požadavek na základě obsahu přijaté zprávy. Níže je jako jednoduchý příklad uvedena „handle“ funkce pro kontrolu relé, které podporuje pouze dva stavy (ON/OFF).

```
def handleRelay(msg, mqtt: mqtt, SANNode: modbus.SANNode):
    #msg
    # {"state": True/False}
    if "state" in msg:
        if msg["state"] == True:
            SANNode.relayControl(True)
            sendData.relay(SANNode, mqtt)
        elif msg["state"] == False:
            SANNode.relayControl(False)
            sendData.relay(SANNode, mqtt)
```

Funkce analyzuje obsah zprávy, na základě něhož je poté vykonán příslušný úkon s využitím metody `relayControl()` objektu `SANNode`. Po vykonání daného úkonu je odeslána odpověď webovému GUI pomocí metody objektu `sendData`.

8.3.3 Simulace letu HAPSu

Jak již bylo zmíněno v podkapitole 8.1, hlavním důvodem pro implementaci simulace letu HAPSu byla potřeba vhodné demonstrace bezpečnostních funkcí SCC. Z důvodu, že primárním cílem nebylo vytvořit plnou simulaci letu, včetně implementace fyziky letu, je implementovaná simulace pouze zjednodušeným modelem.

Základem simulace jsou virtuální senzor GPS a tři virtuální aktuátory, resp. motory. Virtuální senzor GPS uchovává pozici virtuálního HAPSu v trojrozměrné kartézské soustavě souřadnic pomocí třech různých proměnných pro pozici *x*, *y*, a *z*. Pro každou ze tří souřadnicových os je implementován virtuální aktuátor, který je schopen změnit virtuální GPS souřadnici dané osy v kladném, či záporném směru.

Každý virtuální aktuátor je definovaný jako třída. Chování jednotlivých aktuátorů a následně i jejich vliv na pohyb virtuálního HAPSu specifikují následující atributy třídy.

```
    # Status
    __status = False    # ON/OFF
    __power = 0         # 0 - 100 %
    __mode = ""   # "UP"/"DOWN", "LEFT"/"RIGHT", "FORWARD"/"BACKWARDS"
```

Atribut `status` popisuje, zda je aktuátor zapnutý, či nikoliv. Atribut `power` specifikuje procentuální výkon daného aktuátoru. Atribut `mode` slouží ke specifikaci směru působení aktuátoru a následně pohybu virtuálního HAPSu na dané ose.

Samotná simulace je implementována v souboru *sim.py*, který obsahuje třídu `FlightSimulation`. V této třídě jsou využity zmíněné virtuální aktuátory a virtuální senzor GPS. Hlavní metodou je metoda `runSimulation()`, která již byla uvedena při popisu hlavního bloku kódu. V bloku kódu níže je uvedena ukázka naprogramované simulace pohybu HAPSu ve směru osy *x*.

```
# Handle X axis
speed = (float(self.x_actuator.getPower())/100)*float(self.__x_max_speed)
distance = self.__simTick*speed     # distance covered in one tick

if self.x_actuator.getMode() == "LEFT":
    self.gps.x = self.gps.x - distance
elif self.x_actuator.getMode() == "RIGHT":
    self.gps.x = self.gps.x + distance
```

Celková rychlost, kterou se virtuální HAPS pohybuje, odpovídá procentuální hodnotě výkonu z definované maximální rychlosti, kterou je aktuátor schopen v daném směru vyvinout. Vzdálenost je poté vypočítána jako součin této rychlosti a doby jedné periody běhu simulace, která je stanovena na 0,1 s. Následně je potřeba zjistit, kterým směrem je aktuátor nastaven, zda má působit v kladném směru osy, či záporném. Na základě toho je příslušná GPS souřadnice buď zvětšena o uraženou vzdálenost nebo naopak zmenšená.

8.3.4 Vysokoúrovňový kontrolér letu

Vysokoúrovňový kontrolér letu je v aplikaci zodpovědný za autonomní vykonání vysokoúrovňových požadavků. Je naprogramován v souladu s návrhem z podkapitoly 7.4.2, s výjimkou implementace nouzového režimu.

Kód vysokoúrovňového kontroléru je obsažen v souboru *HAPSController.py*. Jsou podporovány tři typy vysokoúrovňových požadavků a funkce pro zastavení všech aktuátorů a zrušení probíhajících sekvencí.

Prvním vysokoúrovňovým požadavkem je požadavek pro vzlétnutí. Za kontrolu virtuálních aktuátorů a vykonání tohoto požadavku je zodpovědná metoda `controlLiftOff()`. Dále je implementována podpora pro vysokoúrovňové požadavky pro přistání a přesun na konkrétní pozici, které jsou vykonány pomocí metod `controlLanding()`, resp. `controlMovement()`. Tyto kontrolní metody představují autonomní řízení určitého úkonu.

Pro aktivaci těchto metod jsou v příslušné „handle“ funkci `handleFlightCommands()` použity metody `liftOff()`, `landing()` a `move()`. Tyto metody dají signál kontrolním metodám popsaným v minulém odstavci, že mají začít vykonávat daný úkon. V hlavní smyčce kódu obstarává obsloužení kontrolních funkcí metoda `run()`, která již byla uvedena v podkapitole 8.3.1. Tato metoda pouze volá kontrolní metody popsané v předchozím odstavci a zajišťuje jejich vykonání.

8.3.5 CPS IPS

Implementovaný PoC CPS IPS je typu „rule based“ – opírá se tedy o pevně definovaná pravidla. Tato pravidla specifikují množinu povolených stavů různých částí HAPSu. Úkolem CPS IPS je zajistit setrvání stavů hlídaných částí v této povolené množině.

Jelikož je implementovaný CPS IPS založený na pravidlech, je nejdříve nutné tato pravidla definovat. V aplikaci jsou tato pravidla definována v konfiguračním souboru *CPSIPSrules.yaml*. Pro ukázku funkčnosti byly definovány pravidla limitující procentuální výkon aktuátorů a množina povolených GPS souřadnic, která tvoří povolenou 3D oblast, kde se může virtuální HAPS volně pohybovat bez omezení ze strany CPS IPS. Tento soubor pravidel se dá v kontextu návrhu SCC označit jako filtr obsahu příchozích požadavků. Dále byl implementovaný i časový filtr frekvence příchozích požadavků, který zajišťuje maximální frekvenci příchozích požadavků, které budou zpracovány. Níže je ukázka pravidel pro jeden aktuátor a GPS souřadnic z konfiguračního souboru *CPSIPSrules.yaml*.

```
x_flight_actuator:
  power:
    min: 0
    max: 100
  freq: 1
```

```
GPS:
  x:
    min: -500
    max: 500
  y:
    min: -500
    max: 500
  z:
    min: 19500
    max: 20500
```

Samotné povolení, či blokování požadavku na základě těchto pravidel je realizováno pomocí podmíněného vykonání příslušné „handle“ funkce. Příkladem může být aktuátor pro pohyb ve směru osy *x*, jehož „handle“ funkcí je funkce `handleXflight()`. Funkcí z rodiny CPS IPS pro tento aktuátor je funkce `filterXflight()`. Tato funkce má návratovou hodnotu typu *boolean*. Pokud příchozí požadavek vyhovuje definovaným pravidlům, a to jak obsahově, tak frekvencí příchodu, funkce navrátí hodnotu *True*, což zajistí vykonání funkce `handleXflight()`, která se nachází v bloku kódu podmíněném fukci `filterXflight()`. V opačném případě, kdy filtrační funkce navrátí hodnotu *False*, funkce vykonána nebude a na příkazový řádek bude vypsána informace o této skutečnosti spolu s obsahem požadavku, který blokaci způsobil.

Speciálním případem je funkce `correctGPSposition()`. Tato funkce je volána každý průběh hlavní smyčkou. Jejím úkolem je kontinuálně monitorovat pozici virtuálního HAPSu a v případě detekce porušení definované množiny GPS souřadnic autonomně kompenzovat řízení HAPSu takovým způsobem, aby došlo k návratu do definované množiny GPS souřadnic.

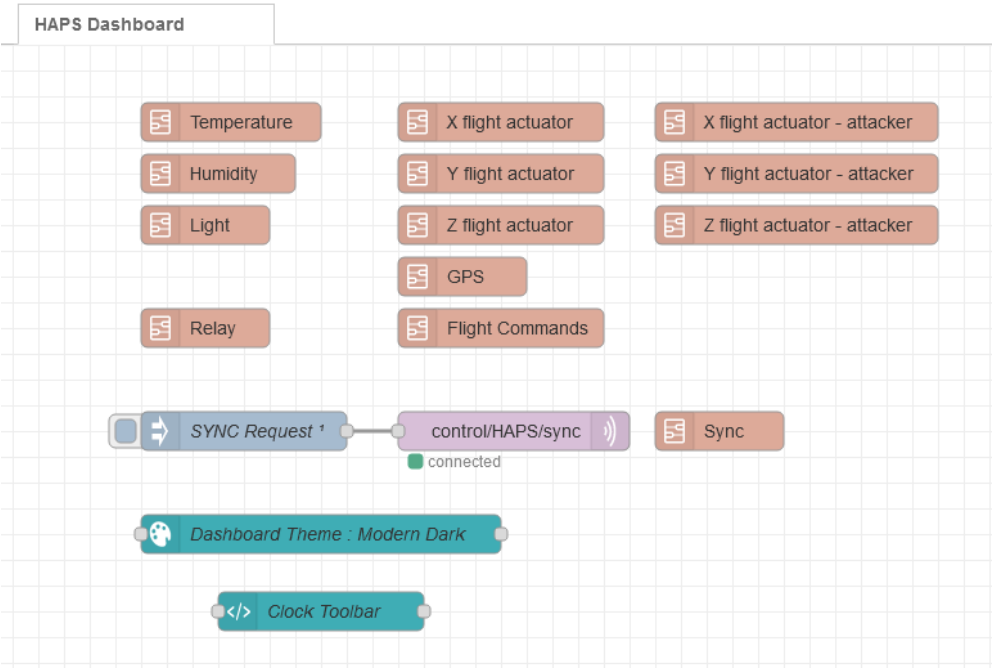
8.4 Implementace webového GUI

K implementaci webového GUI byl využit nástroj Node-RED, jak již bylo zmíněno v podkapitole 8.2.2. Node-RED je nástrojem pro vizuální programování pomocí spojování funkčních bloků. Samotný Node-RED má již značnou část funkčních bloků předdefinovanou, je ale možné definovat vlastní funkční blok s využitím Javascriptu. Obě možnosti jsou pro implementaci grafického GUI využity.

Běžící program můžeme v prostředí Node-RED nazývat jako tzv. „flow“. „Flow“ se může skládat z dalších subprogramů – tzv. „subflow“. V následujících podkapitolách jsou popsány jednotlivé části webového GUI implementovaného v Node-RED.

8.4.1 Hlavní flow

Hlavní „flow“ je vyobrazen na *obr. 8.10*. Hlavní „flow“ je složen zejména z dalších „subflows“. Tento přístup byl zvolen pro přehlednější orientaci v celém programu díky jeho rozsáhlosti.

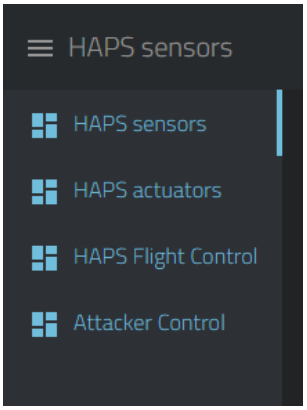


Obr. 8.10 *Hlavní „flow“ webového GUI v prostředí Node-RED*

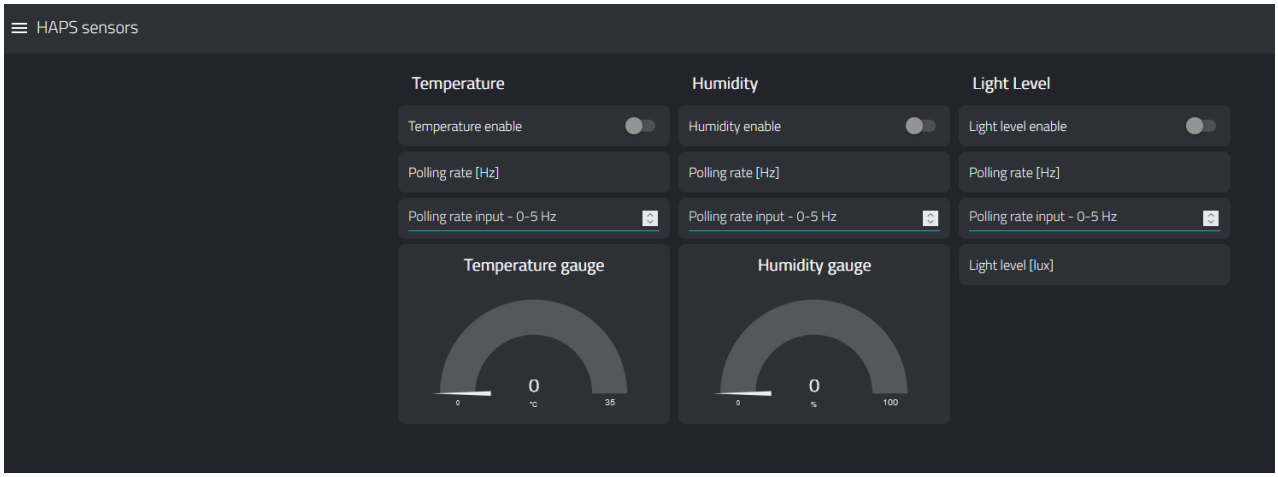
„Subflow“, který je zodpovědný za vytvoření samotného „frontendu“ webového GUI, je „subflow“ „*Dashboard Theme: Modern Dark*“. „Subflow“ „*Clock Toolbar*“ slouží pro zobrazení aktuálního času ve webovém GUI.

Další „subflows“ jsou čistě funkčního typu a slouží ke zpracování dat z SCC a k posílání požadavků na SCC. Tyto „subflows“ je možné rozdělit na tři typy podle toho, kterou část aplikace zpracovávají – senzorická část, část akčních členů a „subflows“ pro ovládací panel simulace letu HAPSu. Tomuto rozdělení odpovídá i struktura webového GUI, které je rozděleno do třech

hlavních částí. První část slouží pro správu senzorů, druhá pro správu aktuátorů, třetí část je ovládacím panelem pro simulaci letu HAPSu. Speciální čtvrtou částí je pak ovládací panel pro simulovaného útočníka, který je pouze modifikací standardního ovládacího panelu. Mezi zmíněnými částmi je ve webovém GUI možné přepínat pomocí nabídky v levém horním rohu, viz *obr. 8.11*. Na *obr. 8.12* je ukázka senzorické části webového GUI. Z důvodu přehlednosti budou v dalších částech textu zobrazeny pouze menší výřezy GUI.



Obr. 8.11 *Volba aktivní karty webového GUI*



Obr. 8.12 *Ukázka webového GUI*

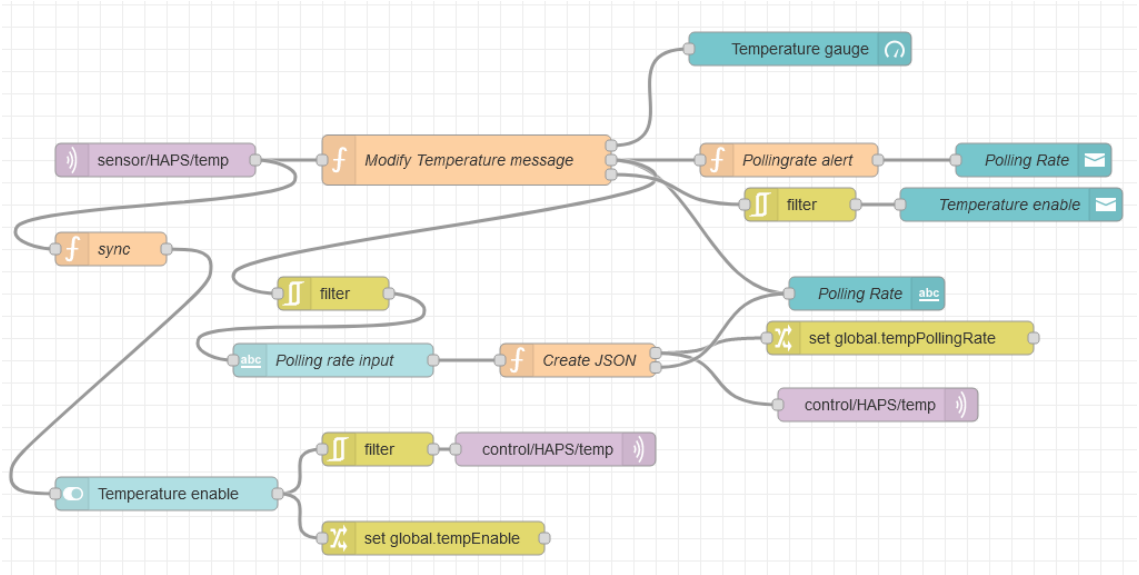
8.4.2 Senzorická část

Do senzorické části patří z *obr. 8.10* následující „subflows“ – *Temperature, Humidity, Light*. Jedná se o „subflows“ zodpovědné za zpracování dat ze senzorů teploty, vlhkosti a intenzity světla na SCC. Dále jsou zodpovědné za vizualizaci dat ve webovém GUI, za aktivaci daného senzoru a za nastavení jeho frekvence měření.

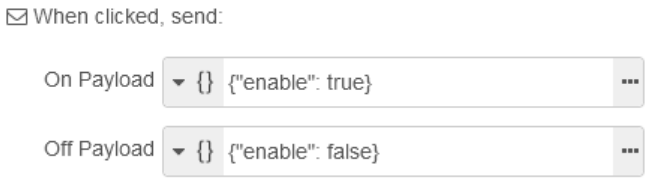
Jako příklad lze uvést „subflow“ *Temperature*, který je vyobrazen na *obr. 8.13*. V levé horní straně je umístěn funkční blok *MQTT subscribe*, přijímací data z daného MQTT témata. Funkční blok *Modify Temperature message* je zodpovědný za zpracování příchozí zprávy z SCC ve formátu JSON. Přijímání dat z SCC primárně slouží k vizualizaci naměřené hodnoty ve webovém GUI. SCC ale také prostřednictvím MQTT zpráv informuje webové GUI o provedení požadovaných úkonů – v tomto případě o aktivaci/deaktivaci senzoru a změně frekvence měření. Díky této zpětné vazbě se požadované změny projeví i ve stavu GUI. Pokud se tomu tak nestane, značí to chybu ve spojení mezi webovým GUI a SCC.

Další částí „subflow“ *Temperature* je tlačítko pro aktivaci/deaktivaci teplotního senzoru *Temperature enable*. Nastavení jeho chování je vyobrazena na *obr. 8.14*, kde lze vidět nastavení zasílané zprávy pro SCC v případě stavu ON i v případě stavu OFF.

Nastavování frekvence měření probíhá v textovém vstupním poli *Polling rate input*. Zde uživatel manuálně zadá hodnotu frekvence měření z definovaného rozsahu 0-5 Hz. Vstupní hodnota je dále zpracována funkčním blokem *Create JSON*, který zprávu formátuje do příslušného formátu JSON. Ta je poté odeslána pomocí protokolu MQTT na téma odebírané SCC.

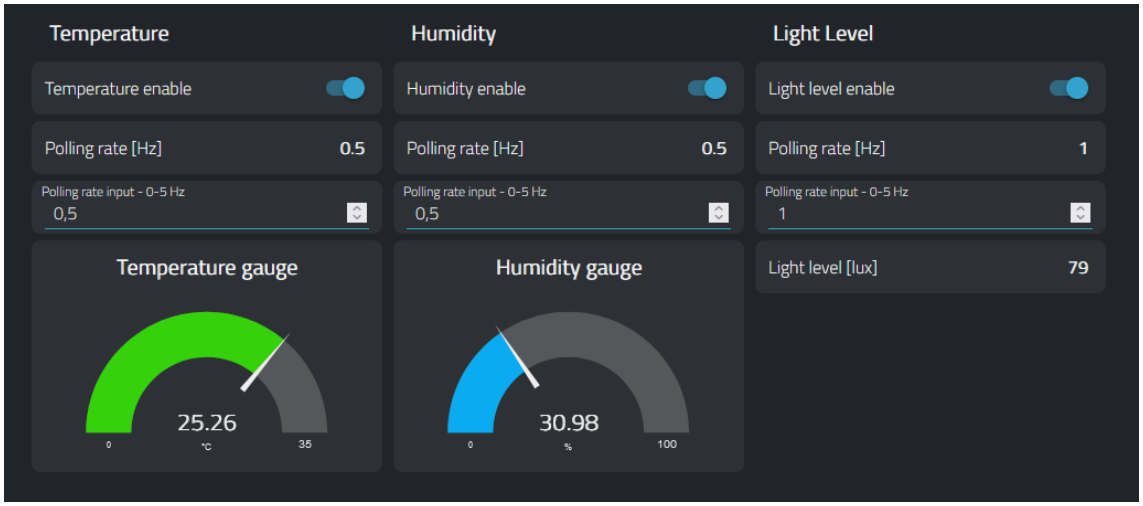


Obr. 8.13 *„Subflow“ „Temperature“*



Obr. 8.14 *Nastavení tlačítka „Temperature enable“*

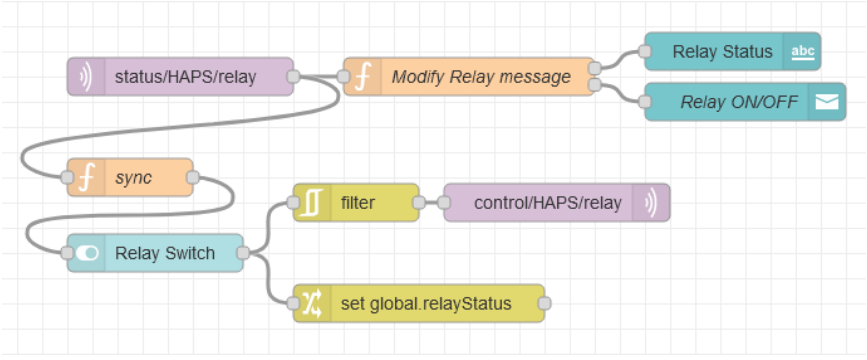
Část webového GUI pro senzory je vyobrazena na *obr. 8.15*. Například blok *Temperature* ve webovém GUI přímo koresponduje se „subflow“ *Temperature* z *obr. 8.12*. Na *obr. 8.15* lze vidět již deklarované vlastnosti, jako je tlačítko pro aktivaci/deaktivaci senzoru, změna a zobrazení frekvence měření, či vizualizace naměřených dat.



Obr. 8.15 *Webové GUI – HAPS senzory*

8.4.3 Část akčních členů

Jako zástupce aktuátorů bylo pro použití v aplikaci zvolené jednoduché relé. Webové GUI je tak pro tento prvek poměrně jednoduché – tlačítko na aktivaci/deaktivaci relé a zobrazení jeho aktuálního stavu. „Subflow“ *Relay* je zodpovědný za tuto funkcionalitu. Jeho struktura v nástroji Node-RED je vyobrazena na *obr. 8.16*.



Obr. 8.16 *„Subflow“ „Relay“*

Část webového GUI použitá pro ovládání relé je vyobrazena na *obr. 8.17*.

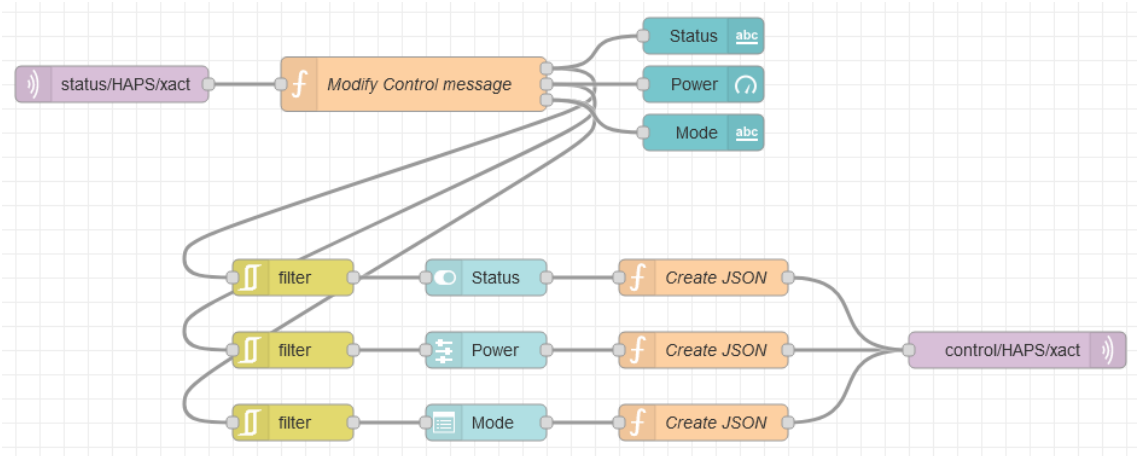
Obr. 8.17 *Webové GUI – HAPS aktuátory (vlevo – stav relé ON, vpravo – stav relé OFF)*

8.4.4 Ovládací panel simulace letu HAPSu

Ovládací panel simulace letu HAPSu se skládá z celkem pěti bloků. Tři bloku jsou dedikovány kontrole virtuálních aktuátorů – *X flight actuator*, *Y flight actuator* a *Z flight actuator*. Dále je zde blok *GPS*, který slouží pro zobrazení aktuální GPS pozice. Blok *GPS* ale slouží i pro funkci „teleport“, která umožňuje manuálně změnit aktuální GPS souřadnice HAPSu. Tato funkce usnadňuje kontrolu funkcionality systému – často není praktické čekat, než virtuální HAPS sám dosáhne požadované pozice. Poslední z bloků v ovládacím panelu simulace je blok *Flight Commands* sloužící pro vysílání vysokoúrovňových požadavků.

Jako příklad je na *obr. 8.18* uvedena struktura „subflow“ *X flight actuator*. V horní části obrázku se nacházejí funkční bloky *Status*, *Power* a *Mode*. Tyto funkční bloky slouží pro zobrazení stavu těchto proměnných tak, jak je reportuje SCC.

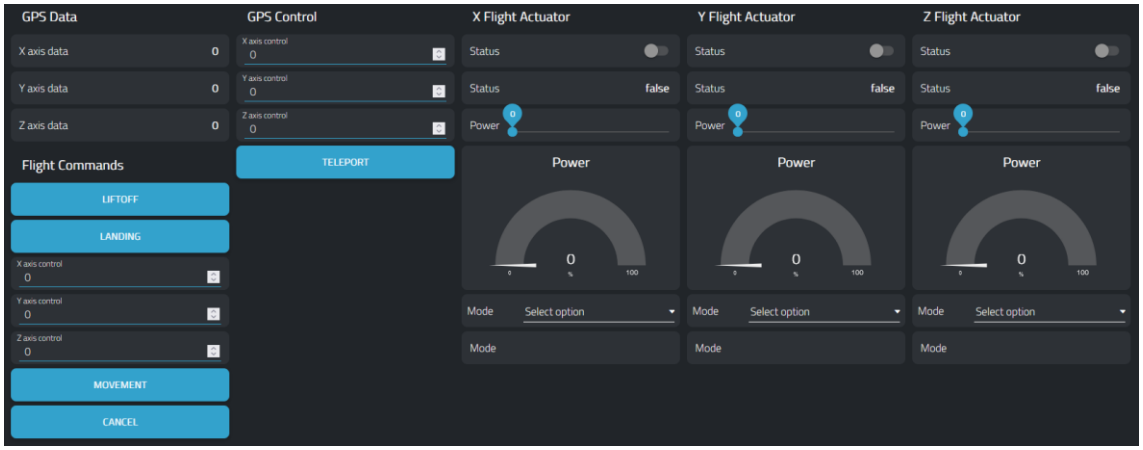
Ve spodní části obrázku jsou identicky pojmenované funkční bloky, nicméně nejedná se o bloky totožné, jak napovídá jejich grafická reprezentace. Funkční blok *Status* reprezentuje tlačítko, kterým je možné daný virtuální aktuátor aktivovat, či deaktivovat. Funkční blok *Power* představuje posuvník, pomocí kterého lze nastavovat výkon motoru v rozmezí 0–100 %. Funkční blok *Mode* umožňuje uživateli volit směr působení virtuálního aktuátoru.



Obr. 8.18 *„Subflow“ „X flight actuator“*

Na *obr. 8.19* je vyobrazen implementovaný ovládací panel simulace letu HAPSu. V pravé části jsou vidět bloky pro kontrolu virtuálních aktuátorů. V horní levé části je pak zobrazena vizualizace aktuálních GPS dat spolu s funkcí „teleport“, kde může uživatel manuálně specifikovat GPS souřadnice. V levé dolní části je ukázán blok *Flight Commands*, díky kterému může uživatel vyslat na SCC vysokoúrovňové příkazy. Konkrétně se jedná o příkazy:

- LIFTOFF – HAPS autonomně vyletí do výšky 20 km, kde se zastaví
- LANDING – HAPS autonomně přistane na zem (výška 0 m)
- MOVEMENT – HAPS se přesune na uživatelem specifikované souřadnice
- CANCEL – HAPS zruší aktuálně prováděné vysokoúrovňové sekvence a vypne všechny motory



Obr. 8.19 *Webové GUI – Ovládací panel simulace letu HAPSu (v klidu)*

Na *obr. 8.20* jsou aktivovány všechny tři virtuální aktuátory na různé výkony. Je možné si všimnout, že každý aktuátor má také zvolen směr letu, a GPS data jsou aktualizována.

Obr. 8.20 *Webové GUI – Ovládací panel simulace letu HAPSu (v provozu)*

8.5 Implementace SAN uzlu

Implementace SAN uzlu spočívala v naprogramování vhodného firmware pro mikrokontroler Arduino UNO. Firmware byl programován v C/C++ s využitím knihoven zmíněných v podkapitole 8.2.3.

Firmware mikrokontroleru musel být naprogramován tak, aby vyhovoval potřebám SCC. Tyto potřeby jsou zejména čtení senzorických dat ze senzorů připojených ke sběrnici I2C, změna frekvence vyčítání senzorických dat a podpora protokolu Modbus přes sběrnici RS-485.

Jak již bylo zmíněno v podkapitole 8.2.3, pro demonstraci funkcionality SAN uzlu byly zvoleny dva senzory komunikující po sběrnici I2C. V reálném provozu bude HAPS vyžadovat připojení podstatně většího množství senzorů různých druhů. Použitý koncept SAN uzlu je ale velmi dobře škálovatelný a je jednoduché jej doplnit o další senzory či aktuátory. K SCC bude připojeno větší množství SAN uzlů pomocí sběrnice RS-485 s využitím protokolu Modbus. Větší délky kabeláže náchylnější na rušení pokryje spojení „SCC – SAN uzel“ pomocí sběrnice RS-485. K SAN uzlům pak může být pomocí sběrnice I2C připojeno několik senzorů a aktuátorů s již významně kratší kabeláží. To je velice důležité pro sběrnici I2C, jelikož sběrnice I2C nebyla navržena pro robustnost a odolnost vůči rušení.

Kód pro platformu Arduino se tradičně skládá ze dvou hlavních funkcí – `setup()` a `loop()`. Funkce `setup()` proběhne na začátku programu pouze jednou. Standardně tedy slouží k nastavení různých proměnných a inicializaci senzorů. Funkce `loop()` pak představuje nekonečnou smyčku firmwaru, ve které mikrokontroler pracuje.

8.5.1 Inicializace

Ve funkci `setup()` dochází k inicializaci protokolu Modbus a k inicializaci použitých senzorů. Kód pro inicializaci protokolu Modbus je uveden níže.

```
/* MODBUS INIT
baudrate = 19200
address = 1
transmit enable pin = 2
holding registers size = TOTAL_REGS_SIZE
low latency = 0
*/
modbus_configure(19200, 1, 2, TOTAL_REGS_SIZE, 0);
```

Dále se inicializují senzory pomocí funkcí uvedených níže.

```
//init HTU21D
  if(!htu.begin()){
    while(1);
  }

  //init BH1750
  Wire.begin();
  lightSensor.begin(BH1750::CONTINUOUS_HIGH_RES_MODE);
```

8.5.2 Nekonečná smyčka

Ve funkci `loop()` jsou vždy čtená senzorická data po uplynutí periody vyčítání hodnot. Tato perioda odpovídá převrácené hodnotě frekvence měření, zvolené ve webovém GUI. Senzorická data jsou uložena do definovaných registrů protokolu Modbus, díky kterým je možné tyto hodnoty číst z SCC.

Jako příklad je uvedeno měření teploty a vlhkosti ze senzoru HTU21D. Po uplynutí periody vyčítání hodnot (kontrolováno v podmínce) dojde k přečtení hodnot ze senzoru a k uložení do registru. Ukládané hodnoty jsou násobené 100x z důvodu, že tyto registry drží pouze celočíselné hodnoty. Aby bylo možné získat přesnost na dvě desetinná čísla, jsou měřená data násobena 100x, čímž dosáhneme při přenosu požadované přesnosti. Kód SCC s tímto faktem počítá a následně jsou hodnoty opět poděleny.

```
  //read sensors after sampleperiod passed
  if(currentTime - prevTime_htu21d >= sampleperiod_htu21d){
    //store temp value in MODBUS holding register
    holdingRegs[TEMP_HTU21D] = htu.readTemperature()*100;
    // -> need to set 2 decimal points in master
    holdingRegs[HUM_HTU21D] = htu.readHumidity()*100;

    prevTime_htu21d = currentTime;
  }
```

Kapitola 9
Testování navržené architektury HAPS a implementace SCC

Tato kapitola je zaměřena na teoretický návrh testů navržené zabezpečené architektury HAPS, která byla představena v podkapitole 6.3. Dále se kapitola věnuje testování implementace komponenty SCC, která byla představena v kapitole 8.

9.1 Návrh testů navržené architektury HAPS

Návrh testů pro představenou architekturu HAPS (viz podkapitola 6.3 – *obr. 6.1* a *obr. 6.2*) vychází z definované funkcionality, zejména té bezpečnostní. Tato bezpečnostní funkcionalita je shrnutá v *tab. 9.1*.

Tab. 9.1 *Shrnutí bezpečnostní funkcionality navržené architektury HAPS*

Sekce	Bezpečnostní prvek	Bezpečnostní funkce
Rádio link Země - HAPS	Zabezpečení rádiového linku	Šifrování veškeré komunikace po rádiovém spojení, autentizace
	TLS Session mezi GS a HAPS PC	Šifrování MQTT provozu, oboustranná autentizace
HAPS Router	Firewall, ACL	Síťová segmentace, oddělení provozu HAPS a TELCO
SCC	CPS IPS	Filtrace požadavků na SCC, zajištění setrvání HAPSu v bezpečném stavu, analýza senzorických dat
	TLS mezi HAPS PC a SCC	Šifrování MQTT provozu, oboustranná autentizace
HAPS PC	Firewall	Povolení vhodného provozu
	IPS	Filtrace útoků při selhání jiných ochranných mechanismů
	TLS mezi HAPS PC a SCC	Šifrování MQTT provozu, oboustranná autentizace
	TLS mezi GS a HAPS PC	Šifrování MQTT provozu, oboustranná autentizace
Záložní MQTT Broker	Záložní MQTT Broker	Odstranění problému SPOF na straně MQTT Brokeru

Navržené testy musí vhodně testovat bezpečnostní funckionalitu prvků z *tab. 9.1*. Dále je vhodné provést penetrační testy použitých zařízení a aplikací. Níže jsou definovány testy ověřující funkčnost navržených bezpečnostních opatření.

Rádiový link Země – HAPS

Tab. 9.2 *Návrh testů pro sekci „Rádiový link Země – HAPS“*

Funkce	Test
Zabezpečení rádiového linku	• Odposlouchávání rádiového provozu • Ověření, že odposlouchávaný provoz je šifrovaný • Ověření bezpečnosti použitého bezpečnostního protokolu • Zachycení „4-way handshake“ a pokus o zjištění hesla pomocí slovníkového útoku (*pozn.: pouze pokud je použit protokol WPA2-PSK, pokud je použita autentizace pomocí 802.1x nelze tento útok provést)*
TLS Session mezi GS a HAPS PC	• Ověření funkčnosti TLS – ověření oboustranné autentizace a šifrování provozu • Skenování TLS zranitelností pomocí nástrojů (např. *testssl.sh* [70]

Dále se doporučuje provést testování odolnosti vůči různým druhům DoS útoků na rádiový link Země – HAPS. Mezi tyto útoky patří např.:

- L1 DoS
  - Typicky se jedná o rušení komunikace rádiovým signálem
- L2 DoS
  - Při použití protokolu 802.11 se může jednat např. útoky pomocí deautentizačních paketů
- L3 DoS
  - Jedná se např. o ping flood, TCP SYN flood, atd.

HAPS Router

Tab. 9.3 *Návrh testů pro sekci „HAPS Router“*

Funkce	Test
Síťová segmentace, oddělení provozu HAPS a TELCO	• Ověření síťové segmentace – pokusit se obejít firewall a umožnit komunikaci z TELCO PC na HAPS PC (využití metod pro obcházení firewallu) • Ověřit, že pouze pozemní stanice může komunikovat s HAPS PC

SCC

Tab. 9.4 *Návrh testů pro sekci „SCC“*

Funkce	Test
CPS IPS	• Ověřit správnou funkci filtru obsahu příchozích požadavků • Zadávání požadavků, které by dostaly HAPS do nebezpečného stavu (záleží na konkrétních použitých aktuátorech) • Ověřit správnou funkci časového filtru příchozích požadavků • Zasílat požadavky s větší frekvencí, jejichž vykonání v plné rychlosti by poškodilo aktuátory • Ověřit reakci CPS IPS na „Request Flooding“ – zahltit CPS IPS velkým množstvím požadavků ve velmi krátkém čase po dostatečně dlouhou dobu – sledovat stabilitu systému • Ověřit správnou funkci kontextuálního filtru příchozích požadavků • Zadávat kombinace požadavků, které by v reálném provozu neměly nastat – např. zapnutí dvou proti sobě působících motorů (záleží na konkrétní fyzické realizaci) • Otestovat odolnost proti GPS spoofingu • Otestovat odolnost proti vzdálenému ovlivňování senzorů
TLS mezi HAPS PC a SCC	• Ověření funkčnosti TLS – ověření oboustranné autentizace a šifrování provozu • Skenování TLS zranitelností pomocí nástrojů (např. *testssl.sh* [70]

Vzhledem ke kritickému významu bezpečnosti SCC je nutné provést i celkovou bezpečnostní analýzu systému a penetrační testování této komponenty. Doporučuje se provést standardní bezpečnostní analýzu pomocí automatizovaných nástrojů z vnějšího pohledu (Nmap, Nessus, atd.), ale také analýzu pomocí automatizovaných nástrojů testující vnitřní zranitelnosti systému umožňující eskalaci privilegií (např. skript *Linpeas* [71]). Zásadní je také zaměřit se zejména na porty 1883 a 8883, na kterých probíhá komunikace protokolem MQTT v nezabezpečené (1883) a zabezpečené (8883) podobě. Bezpečnostní analýzu pomocí automatizovaných nástrojů je vhodné doplnit i manuální bezpečnostní analýzou.

Dále je nutné provést bezpečnostní analýzu samotné aplikace, či aplikací, které byly speciálně vyvinuté pro SCC. Zranitelnosti v těchto proprietárních aplikacích mohou být kritické a mohou vést ke kompromitaci systému. Lze očekávat, že z důvodu efektivity bude aplikace programována v nízkoúrovňovém programovacím jazyce, jako např. C/C++. Takové jazyky jsou často náchylné k chybám programátorů, které vedou ke zranitelnostem souvisejícím se správou paměti, např. „buffer overflow“. Tyto zranitelnosti bývají často kritické a je nutné se na ně zaměřit při bezpečnostní analýze.

HAPS PC

Tab. 9.5 *Návrh testů pro sekci „HAPS PC“*

Funkce	Test
Firewall	• Ověřit, že s HAPS PC mohou komunikovat pouze povolené IP adresy po povolených protokolech
IPS	• Otestovat detekci a blokování útoků, které projdou firewallem • Otestovat útoky typu skenování, bruteforce (např. SSH login), DoS • Otestovat reakci systému na L3 DoS útok – vytížení CPU, stabilita systému, atd. • Otestovat reakci na uměle vytvořenou zranitelnost na HAPS PC – sledovat, zda IPS detekuje a zablokuje sofistikovanější útoky
TLS mezi HAPS PC a SCC	• Ověření funkčnosti TLS – ověření oboustranné autentizace a šifrování provozu • Skenování TLS zranitelností pomocí nástrojů (např. *testssl.sh* [70]
TLS mezi GS a HAPS PC	• Ověření funkčnosti TLS – ověření oboustranné autentizace a šifrování provozu • Skenování TLS zranitelností pomocí nástrojů (např. *testssl.sh* [70]

Stejně jako v případě SCC je nutné, aby HAPS PC prošel bezpečnostní analýzou a penetračním testováním. Doporučení pro bezpečnostní analýzu SCC jsou analogická i pro HAPS PC.

Jelikož HAPS PC bude zodpovědný za logické řízení HAPSu pomocí algoritmu pro autonomní let, je z hlediska fyzické bezpečnosti nezbytně nutné zajistit také dostatečné testování správné funkce tohoto algoritmu.

Záložní MQTT Broker

Tab. 9.6 *Návrh testů pro sekci „Záložní MQTT Broker“*

Funkce	Test
Záložní MQTT Broker	• Ověřit správné přepnutí komunikace na záložní MQTT Broker v případě simulovaného výpadku HAPS PC

9.2 Testování implementovaného PoC komponenty SCC

Tato podkapitola je věnována návrhu testů a následnému testování implementovaného PoC komponenty SCC z kapitoly 8. Cílem testů je ověřit funkčnost implementovaných funkcí.

9.2.1 Návrh testů PoC komponenty SCC

Úkony specifikované v jednotlivých testech se provádějí ve webovém GUI, pokud není uvedeno jinak. V tabulce je vždy uvedeno kódové označení testu pro pozdější referenci ve sloupci označeném symbolem „#“.

Senzory a aktuátory

Tab. 9.7 *Návrh testů pro PoC SCC – senzory a aktuátory*

Funkce	Test	#
Čtení teplotního senzoru	• Aktivovat senzor • Ověřit aktualizování senzorických dat	S1
Čtení senzoru vlhkosti	• Aktivovat senzor • Ověřit aktualizování senzorických dat	S2
Čtení senzoru intenzity světla	• Aktivovat senzor • Ověřit aktualizování senzorických dat	S3
Změna frekvence měření teplotního senzoru	• Změnit frekvenci měření ve specifikovaném rozsahu • Ověřit, zda se senzorická data aktualizují s frekvencí odpovídající nastavené frekvenci měření • Změnit frekvenci měření mimo specifikovaný rozsah – sledovat reakci systému	S4
Změna frekvence měření senzoru vlhkosti	• Změnit frekvenci měření ve specifikovaném rozsahu • Ověřit, zda se senzorická data aktualizují s frekvencí odpovídající nastavené frekvenci měření • Změnit frekvenci měření mimo specifikovaný rozsah – sledovat reakci systému	S5
Změna frekvence měření senzoru intenzity světla	• Změnit frekvenci měření ve specifikovaném rozsahu • Ověřit, zda se senzorická data aktualizují s frekvencí odpovídající nastavené frekvenci měření • Změnit frekvenci měření mimo specifikovaný rozsah – sledovat reakci systému	S6
Zapnutí/Vypnutí aktuátoru	• Změnit stav aktuátoru – sledovat reakci systému, včetně fyzické změny stavu aktuátoru	A1

Simulace letu HAPSu a její ovládání

Tab. 9.8 *Návrh testů pro PoC SCC – simulace letu HAPSu a její ovládání*

Funkce	Test	#
Nízkoúrovňové požadavky - ovládání jednotlivých virtuálních aktuátorů	• Ověřit funkci zapnutí/vypnutí jednotlivých aktuátorů • Ověřit vliv nastavení výkonu jednotlivých aktuátorů na rychlost letu • Ověřit vliv nastavení směru působení jednotlivých aktuátorů na směr letu virtuálního HAPSu (*pozn.: aby se virtuální HAPS začal pohybovat, je nutné aby měl aktuátor nenulový výkon a měl zvolený směr letu)* • Sledovat, zda změny údajů virtuálního GPS korespondují s nastavením jednotlivých aktuátorů	H1
Vysokoúrovňové požadavky	• Ověřit funkčnost implementovaných vysokoúrovňových požadavků • Požadavek LIFTOFF – ověřit, že virtuální HAPS vzlétne do výšky 20 km, kde se sám zastaví • Požadavek LANDING – ověřit, že virtuální HAPS přistane zpět na zem (výška HAPSu je nulová) • Požadavek MOVEMENT – ověřit, že virtuální HAPS se přesune na specifikované souřadnice • Požadavek CANCEL – ověřit, že virtuální HAPS vypne všechny motory a deaktivují se vysokoúrovňové sekvence	H2
Funkce "teleport"	• Zadat GPS souřadnice, do bloku „GPS Control“ ve webovém GUI • Ověřit, že GPS souřadnice v bloku „GPS Data“ ve webovém GUI souhlasí se zadanými souřadnicemi z bloku „GPS Control“	H3

CPS IPS

Tab. 9.9 *Návrh testů pro PoC SCC – CPS IPS*

Funkce	Test	#
Filtr obsahu požadavků - dodržení rozsahu požadovaného výkonu virtuálních aktuátorů	• Vyslat požadavek na změnu výkonu virtuálního aktuátoru mimo povolený rozsah (0-100%) • Vyslat požadavek, ve kterém bude místo očekávané číselné hodnoty text (různě dlouhý) – testování ošetření vstupu	C1
Filtr obsahu požadavků - dodržení rozsahu GPS souřadnic při použití požadavku MOVEMENT	• Vyslat vysokoúrovňový požadavek na přesun virtuálního HAPSu na virtuální GPS souřadnice mimo povolený rozsah souřadnic • Otestovat zpracování hraničních hodnot – zadat souřadnice na hranici povoleného rozsahu souřadnic • Vyslat požadavek, ve kterém bude místo očekávané číselné hodnoty text (různě dlouhý) – testování ošetření vstupu	C2

Časový filtr - dodržení maximální frekvence příchozích požadavků na změnu stavu virtuálních aktuátorů	• Vysílat požadavky s větší frekvencí, než je maximální povolená frekvence • Provést útok, ve kterém zaplavím SCC velkým množstvím požadavků v krátkém čase („Request flooding attack“)	C3
Autonomní udržení virtuálního HAPSu v rozmezí povolených GPS souřadnic	• Zapnout libovolný aktuátor tak, aby se virtuální HAPS přesunul mimo povolený rozsah GPS souřadnic • Otestovat tuto funkcionalitu pomocí funkce „teleport“ – přesunout HAPS mimo povolený rozsah (může simulovat GPS spoofing)	C4

Další funkce

Tab. 9.10 *Návrh testů pro PoC SCC – další funkce*

Funkce	Test	#
Synchronizace GUI a SCC po startu jedné z částí	• Po zapnutí obou částí a uvedení webového GUI do libovolného stavu se vypne libovolná ze dvou částí • Vypnutá část se poté opět zapne a sleduje se, zda byla vypnutá část uvedena do stavu před vypnutím • Test se poté analogicky opakuje pro druhou část	X1
Logování	• Ověřit funkci logování zaslaných zpráv a přijatých požadavků	X2
Zapnutí/Vypnutí pomocí přiložených skriptů	• Ověřit funkčnost skriptů pro zapnutí/vypnutí aplikace (*start.sh* a *stop.sh*) • Pro ověření stačí spustit daný skript a zhodnotit, zda došlo k požadované reakci	X3
Ošetření vstupů na různých místech aplikace	• Zadat na libovolné vstupy aplikace neočekávanou hodnotu, či typ vstupu (např. text místo čísel)	X4

V dalších podkapitolách jsou testovány jednotlivé části PoC SCC. Shrnutí výsledků obsahuje podkapitola 9.2.6.

9.2.2 Testování – Senzory a aktuátory

Testy S1, S2 a S3

Testy S1, S2 a S3 proběhly v pořádku. Po aktivaci příslušného senzoru ve webovém GUI pomocí tlačítka „enable“ se v GUI okamžitě objevily naměřené hodnoty z aktivovaného senzoru. Na *obr. 9.1* je vyobrazen výpis z terminálu po aktivaci teplotního senzoru. Je možné si všimnout přijatého požadavku z webového GUI, reakce SCC na tento požadavek i první odeslanou zprávu obsahující naměřenou teplotu. Analogicky se projevilo i chování zbylých senzorů.

```
> Received message: "{"enable":true}" on topic "control/HAPS/temp"
> Sent message: "{"enable": true}" to topic "sensor/HAPS/temp"
> HTU21D Temperature enable -> True
> Sent message: "{"temperature": 26.54}" to topic "sensor/HAPS/temp"
```

Obr. 9.1 *Reakce SCC na přijatý požadavek na aktivaci teplotního senzoru*

Testy S4, S5 a S6

Pro ověření testů S4, S5 a S6 byla využita funkce logování odeslaných zpráv. Jako příklad je uvedeno měření pomocí senzoru vlhkosti. V části logu, která je zachycena na *obr. 9.2* byla nastavena frekvence měření na 1 Hz. Dle přiložené časové známky je možné si všimnout, že hodnoty naměřené vlhkosti se odesílají jednou za vteřinu. V logu na *obr. 9.3* lze naopak vidět odesílání zpráv při nastavené frekvenci měření na 4 Hz. Z časových známek je patrné, že naměřené hodnoty vlhkosti se odesílají zhruba v intervalu 250 ms. Oba tyto intervaly odpovídají nastavené frekvenci měření v daný moment.

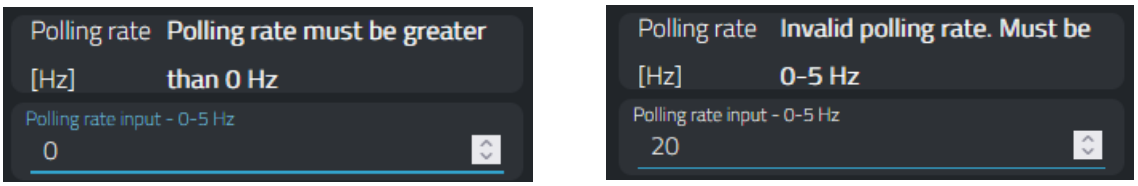

```
{"timestamp": "1652283407.7060647", "pollingrate": 1}
{"timestamp": "1652283407.7073271", "humidity": 30.3}
{"timestamp": "1652283408.670082", "humidity": 30.29}
{"timestamp": "1652283409.6785152", "humidity": 30.29}
{"timestamp": "1652283410.6872208", "humidity": 30.28}
{"timestamp": "1652283411.6956508", "humidity": 30.27}
```


Obr. 9.2 *Měření vlhkosti s frekvencí měření 1 Hz – záznam z logu*


```
{"timestamp": "1652283414.8223617", "pollingrate": 4}
{"timestamp": "1652283414.8233", "humidity": 30.27}
{"timestamp": "1652283414.973207", "humidity": 30.27}
{"timestamp": "1652283415.2234933", "humidity": 30.27}
{"timestamp": "1652283415.473823", "humidity": 30.27}
{"timestamp": "1652283415.7241416", "humidity": 30.27}
{"timestamp": "1652283415.98443", "humidity": 30.27}
{"timestamp": "1652283416.2449398", "humidity": 30.27}
```


Obr. 9.3 *Měření vlhkosti s frekvencí měření 4 Hz – záznam z logu*

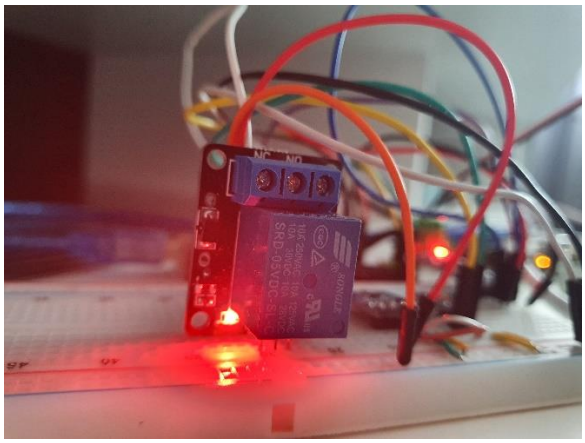
Na *obr. 9.4* vlevo je vyobrazena reakce webového GUI po pokusu o nastavení frekvence měření na 0 Hz. Na *obr. 9.4* vpravo je pak vyobrazena reakce webového GUI po zadání hodnoty frekvence měření mimo povolený interval 0-5 Hz. V obou případech byl tento požadavek zablokován již ve „frontendu“ webového GUI. Kontrola nastavené hodnoty frekvence měření ale probíhá i v kódu aplikace SCC. Filtrace ve „frontendu“ webového GUI jde obejít buď díky injekci dat, či zasíláním zpráv z jiného zdroje. Kontrola přímo v aplikaci SCC ale funguje vždy a senzory tedy nemohou být nastaveny na frekvenci měření mimo rozsah 0-5 Hz ani při různých útocích.



Obr. 9.4 *Zadání neplatných hodnot frekvence měření (vlevo - nulová frekvence, vpravo - mimo povolený rozsah)*

Test A1

Provedení testu A1, při kterém se měnil stav připojeného relé, neprokázal žádné vady. SCC úspěšně získalo požadavek na aktivaci relé a skrze protokol Modbus a sběrnici RS-485 byla předána instrukce SAN uzlu na aktivaci relé. Na *obr. 9.5* je fotografie relé po jeho aktivaci – je možné si všimnout svítící červené diody značící, že relé je v zapnutém stavu. Po vypnutí relé tato dioda již nesvítí.



Obr. 9.5 *Fotografie aktivovaného relé*

9.2.3 Testování – Simulace letu HAPSu a její ovládání

Test H1

K provedení testu H1 byl nejprve aktivován virtuální aktuátor pomocí tlačítka „Status“ ve webovém GUI. Jeho status se tímto přepnul na „true“. Dále byl změněn výkon na hodnotu 50 %. Reakce výpisu v terminálu na tyto události je vyobrazena na *obr. 9.6*.

```
> Received message: "{"enable":true}" on topic "control/HAPS/xact"
> X flight actuator -> ON
> Sent message: "{"x_status": true, "x_power": 0, "x_mode": ""}" to topic "status/HAPS/xact"
> Received message: "{"power":50}" on topic "control/HAPS/xact"
> X flight actuator power set to -> 50%
> Sent message: "{"x_status": true, "x_power": 50.0, "x_mode": ""}" to topic "status/HAPS/xact"
```

Obr. 9.6 *Reakce na přijaté požadavky na změnu stavu virtuálního aktuátoru (nahoře – zapnutí aktuátoru, dole – nastavení výkonu aktuátoru na 50 %)*

V tuto chvíli byl virtuální HAPS stále nehybný. Jak lze vidět na *obr. 9.6*, stále chybí nastavit parametr „mode“. Poté byl ve webovém GUI zvolen směr působení aktuátoru „RIGHT“, což by mělo způsobit pohyb virtuálního HAPSu v kladném směru osy *x*. Jelikož je maximální rychlost, kterou je virtuální aktuátor schopný v daném směru vyvinout, stanovena na 10 m/s, je očekávaná rychlost pohybu virtuálního HAPSu při výkonu aktuátoru 50 % rovna 5 m/s. Na *obr. 9.7* je vyobrazen výpis z terminálu, kde lze toto očekávané chování pozorovat. Je nutno dodat, že aktualizace virtuálních GPS souřadnic je na webové GUI odesílána jednou za vteřinu.

```
> Sent message: "{"x": 0, "y": 0, "z": 0}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": 0, "y": 0, "z": 0}" to topic "sensor/HAPS/gps"
> Received message: "{"mode":"RIGHT"}" on topic "control/HAPS/xact"
> X flight actuator going RIGHT
> Sent message: "{"x_status": true, "x_power": 50.0, "x_mode": "RIGHT"}" to topic "status/HAPS/xact"
> Sent message: "{"x": 0.5, "y": 0, "z": 0}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": 5.5, "y": 0, "z": 0}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": 10.5, "y": 0, "z": 0}" to topic "sensor/HAPS/gps"
```

Obr. 9.7 *Pohyb virtuálního HAPSu po změně parametru „mode“ virtuálního aktuátoru*

Test H2

Test H2 byl zaměřen na ověření funkčnosti implementovaných vysokoúrovňových požadavků. Vysokoúrovňový požadavek „LIFTOFF“ realizuje vzlétnutí virtuálního HAPSu do výšky 20 km. Po stisknutí příslušného tlačítka ve webovém GUI se zahájí vzletová sekvence. SCC v tento moment vypne aktuátory v jiných směrech než ve směru osy *z* a nastaví aktuátor pro pohyb ve směru osy *z* na maximální výkon ve směru působení „UP“. Po dosažení výšky 20 km se tento aktuátor vypne. Výpis terminálu celé vzletové sekvence je vyobrazen na *obr. 9.8*.

```
> Received message: "{"command":"liftoff"}" on topic "control/HAPS/flight"
> Started LIFTOFF sequence
> Sent message: "{"x_status": false, "x_power": 0, "x_mode": "LEFT"}" to topic "status/HAPS/xact"
> Sent message: "{"y_status": false, "y_power": 0, "y_mode": ""}" to topic "status/HAPS/yact"
> Sent message: "{"z_status": true, "z_power": 100, "z_mode": "UP"}" to topic "status/HAPS/zact"
> Sent message: "{"x": 0.0, "y": 0, "z": 7.0}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": 0.0, "y": 0, "z": 17.0}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": 0.0, "y": 0, "z": 27.0}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": 0.0, "y": 0, "z": 19983.0}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": 0.0, "y": 0, "z": 19993.0}" to topic "sensor/HAPS/gps"
> Sent message: "{"z_status": false, "z_power": 0, "z_mode": "UP"}" to topic "status/HAPS/zact"
> Sent message: "{"x": 0.0, "y": 0, "z": 20000.0}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": 0.0, "y": 0, "z": 20000.0}" to topic "sensor/HAPS/gps"
```

Obr. 9.8 *Vysokoúrovňový požadavek „LIFTOFF“*

Funkčnost zbylých vysokoúrovňových požadavků v testu H2 byla analogicky ověřena a nebyly shledány žádné problémy.

Test H3

Test H3 prokázal správné chování pomocné funkce „teleport“. Po zadání určitých GPS souřadnic byly GPS souřadnice virtuálního HAPSu změněny na zadané hodnoty.

9.2.4 Testování – CPS IPS

Test C1

Test C1 se zaměřuje na filtraci obsahu příchozích požadavků na modifikaci výkonu virtuálních aktuátorů. V části „Attacker Control“ webového GUI byl manuálně nastaven výkon na hodnotu přesahující 100 %, konkrétně na hodnotu 150 %. Bylo také testováno zadání záporné hodnoty výkonu. Reakce CPS IPS na tyto požadavky je vyobrazena na *obr. 9.9*. Povolený rozsah výkonu virtuálního aktuátoru je 0 – 100 %.

```
> Received message: "{"power":150}" on topic "control/HAPS/xact"
CPS IPS -> BLOCKED REQUEST: {'power': 150}
```

```
> Received message: "{"power":-100}" on topic "control/HAPS/xact"
CPS IPS -> BLOCKED REQUEST: {'power': -100}
```

***Obr. 9.9** Blokování požadavků mimo povolený rozsah*

Jak je vidět na *obr. 9.9*, oba škodlivé požadavky byly zablokovány pomocí CPS IPS.

Pro otestování ošetření vstupu nebylo možné použít webové GUI, protože při zadání jiného vstupu, nežli číselné hodnoty, byl tento vstup ignorován. K překonání této limitace byl v příkazovém řádku na virtuálním počítači na straně webového GUI použit nástroj *mosquitto_pub*, který umožňuje manuálně publikovat zprávy na specifikovaná MQTT témata. Použitý příkaz, včetně reakce SCC na tuto zprávu, je uveden na obrázku níže. Jak je vidět byl odeslán „string“ sestávající z písmen „A“. Díky použití bloku „try – except“ ve funkci zpracování požadavku byla pouze vypsána chybová hláška a nedošlo k pádu aplikace.

```
❯ mosquitto_pub -h localhost -t control/HAPS/xact -m "{\"power\":\"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\"}" -d
Client mosq-eJ4yTWEClymS4SG7te sending CONNECT
Client mosq-eJ4yTWEClymS4SG7te received CONNACK (0)
Client mosq-eJ4yTWEClymS4SG7te sending PUBLISH (d0, q0, r0, m1, 'control/HAPS/xact', ... (53 bytes))
Client mosq-eJ4yTWEClymS4SG7te sending DISCONNECT
```

```
> Received message: "{"power":"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"}" on topic "control/HAPS/xact"
onMessage Error!!!
> Sent message: "{"x": 379.0, "y": 300.0, "z": 19800.0}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": 379.0, "y": 300.0, "z": 19800.0}" to topic "sensor/HAPS/gps"
```

***Obr. 9.10** Testování ošetření vstupů při zpracování požadavků na změnu výkonu*
(nahoře – použitý příkaz pro poslání zprávy, dole – reakce SCC na přijatou zprávu)

Test C2

K provedení testu C2 byly nastavené GPS souřadnice u požadavky „MOVEMENT“ přesahující povolený rozsah (na osách *x* i *y* je tento rozsah -500 až 500). Reakce CPS IPS na tento požadavek je na *obr. 9.11*. Jak je možné vidět, byl tento požadavek úspěšně blokován.

```
> Received message: "{"command":"move","x":-600,"y":600,"z":20000}" on topic "control/HAPS/flight"
CPS IPS -> BLOCKED REQUEST: {'command': 'move', 'x': -600, 'y': 600, 'z': 20000}
```

Obr. 9.11 *Blokování požadavku „MOVEMENT“ se souřadnicemi mimo povolený rozsah*

Testování požadavku „MOVEMENT“ s hodnotami GPS souřadnic (osa *x* na -500, osa *y* na 500), které jsou na hranici povolených souřadnic je vyobrazeno na *obr. 9.12*. Hraniční hodnoty byly přijaty a virtuální HAPS se na specifikované souřadnice přesunul.

```
> Sent message: "{"x": 0, "y": 0, "z": 20000}" to topic "sensor/HAPS/gps"
> Received message: "{"command":"move","x":-500,"y":500,"z":20000}" on topic "control/HAPS/flight"
> Sent message: "{"x_status": true, "x_power": 100, "x_mode": "LEFT"}" to topic "status/HAPS/xact"
> Sent message: "{"y_status": true, "y_power": 100, "y_mode": "FORWARD"}" to topic "status/HAPS/yact"
> Sent message: "{"z_status": false, "z_power": 0, "z_mode": ""}" to topic "status/HAPS/zact"
> Sent message: "{"x": -1.0, "y": 1.0, "z": 20000}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": -11.0, "y": 11.0, "z": 20000}" to topic "sensor/HAPS/gps"
> Sent message: "{"x": -21.0, "y": 21.0, "z": 20000}" to topic "sensor/HAPS/gps"
```

Obr. 9.12 *Testování hraničních hodnot souřadnic u požadavku „MOVEMENT“*

V případě testování ošetření vstupu u požadavku „MOVEMENT“ byl opět použit nástroj *mosquitto_pub* pro zaslání zfalšované zprávy. Příklad výsledku testu v případě zaslání neplatného požadavku je na *obr. 9.13*. V prvním případě se nejednalo o validní JSON formát, což reflektuje i chybová hláška. V druhém případě byl vstupním parametrem místo číselné souřadnice „string“. V obou případech byla chyba zachycena díky použití bloku „try - except“.

```
> Received message: "{"command":"move","x":AAAAA,"y":10,"z":20000}" on topic "control/HAPS/flight"
> Received message is not JSON
onMessage Error!!!
> Received message: "{"command":"move","x":"AAAAA","y":10,"z":20000}" on topic "control/HAPS/flight"
> ERROR! Coords not integers!
```

Obr. 9.13 *Testování ošetření vstupů u požadavku „MOVEMENT“*

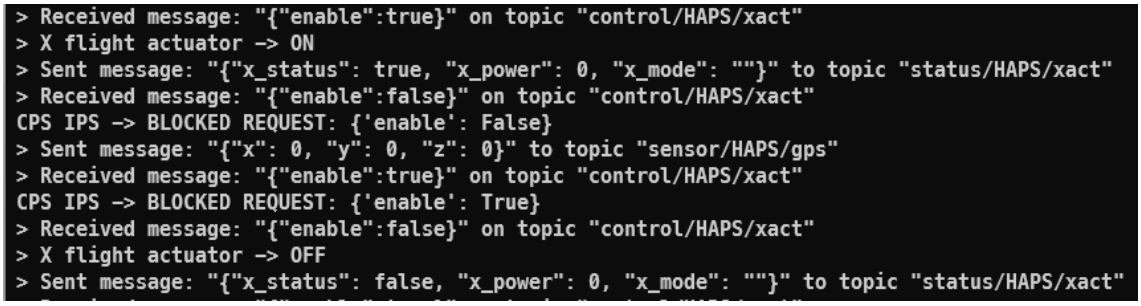
Test C3

Pro test C3 hodnotící funkčnost časového filtru byl napsán krátký skript v jazyce Bash. Kód tohoto skriptu je uveden níže. Argumentem skriptu je frekvence přepínání stavu virtuálního aktuátoru pro osu *x* ze stavu „ON“ do stavu „OFF“ a naopak.

```
#!/bin/bash
period=$(bc -l <<< "1/$1")
while true
do
    mosquitto_pub -h localhost -t control/HAPS/xact -m "{\"enable\":true}"
    sleep $period
    mosquitto_pub -h localhost -t control/HAPS/xact -m "{\"enable\":false}"
    sleep $period
done
```

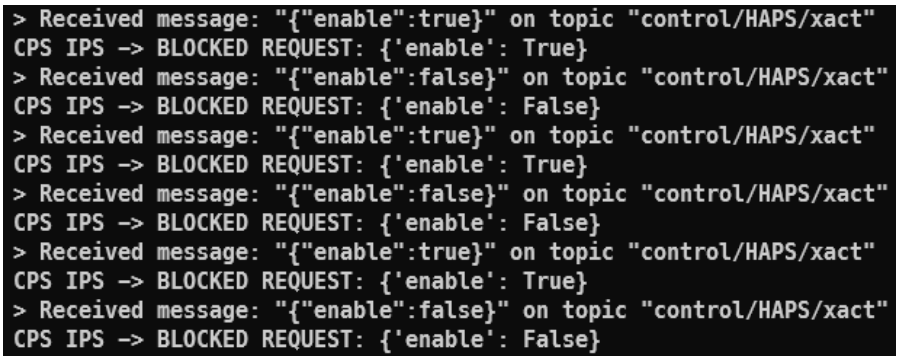
Limitní frekvence byla v CPS IPS nastavena na 1 požadavek za vteřinu. Příchozí požadavky, které přicházejí rychleji, než touto frekvencí, by měly být blokovány.

Při spuštění skriptu s argumentem menším, než 1 požadavek za vteřinu je provoz povolen. V případě, že byl skript spuštěn s vyšší frekvencí vysílání požadavků, jsou již některé požadavky blokovány, viz *obr. 9.14*. V tomto konkrétním případě byla frekvence vysílání požadavků nastavena na 3 požadavky za vteřinu.



Obr. 9.14 *Testování časového filtru příchozích požadavků*

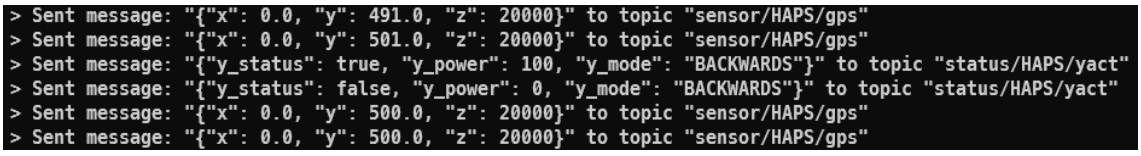
Jako zátěžový test byl dle návrhu testu C3 proveden útok „Request flooding“ pomocí zmíněného skriptu s nastavenou frekvencí vysílaných požadavků na 1000 požadavků za vteřinu. Během tohoto testu byla zablokována majoritní část příchozích požadavků, viz *obr. 9.15*, nicméně nedošlo k pádu aplikace ani ke znatelnému zvýšení vytížení procesoru SCC. Větší vytížení procesoru bylo znát u virtualizovaného počítače s webovým GUI, na kterém ale běží i MQTT Broker a zmíněný testovací skript.



Obr. 9.15 *Blokování příchozích požadavků při útoku „Request flooding“*

Test C4

Cílem testu C4 bylo ověřit funkčnost autonomního udržení virtuálního HAPSu v rozmezí povolených GPS souřadnic. Dle navrženého testu byl aktivován libovolný virtuální aktuátor, konkrétně aktuátor pro osu *y* ve směru „FORWARD“. Díky tomu se virtuální HAPS začal pohybovat v kladném směru osy *y*. Po překročení meze povolených GPS souřadnic se změní směr působení virtuálního aktuátoru, díky čemuž je virtuální HAPS navrácen do povoleného rozmezí GPS souřadnic. Poté je daný virtuální aktuátor deaktivován a HAPS je zastaven na hraniční hodnotě povoleného rozmezí GPS souřadnic. Toto chování je zachyceno na výpisu z terminálu na *obr. 9.16*.



Obr. 9.16 *Udržení virtuálního HAPSu v rozmezí povolených GPS souřadnic*

Tato funkcionalita autonomního udržení virtuálního HAPSu v rozmezí povolených GPS souřadnic byla otestována i za pomocí funkce „teleport“. Zde byly skokově změněny výškové GPS souřadnice na 21 km, přičemž horní hranice je 20,5 km. Ve stejný moment začal HAPS řídit příslušný virtuální aktuátor osy *z*, aby se začal pohybovat směrem dolů do povoleného rozmezí GPS souřadnic. Po dosažení hodnoty 20500 m se příslušný aktuátor vypnul.

Přestože byla ověřena správná funkce implementované korekce GPS souřadnic, v reálném nasazení není tento přístup možný. Díky útoku typu „GPS spoofing“ může díky takto implementované korekci získat útočník kontrolu nad letem HAPSu, a to i přes nasazení CPS IPS. Je tedy nutné implementovat ochrany proti útokům typu „GPS spoofing“ a využívat senzorickou fúzi, např. s využitím akcelerometrů, které mohou konfirmovat pohyb zaznamenaný pomocí senzorů GPS.

9.2.5 Testování – Další funkce

Test X1

Pro otestování funkce synchronizace webového GUI a SCC byl systém (GUI + SCC) uveden do následujícího stavu:

- Zapnuté senzory:
  - Teplota – frekvence měření = 0,5 Hz
  - Vlhkost – frekvence měření = 0,5 Hz
  - Intenzita světla – frekvence měření = 0,5 Hz
- Aktuátor:
  - Stav relé = OFF
- Simulace letu HAPSu
  - Aktuátor osa *x* – OFF
  - Aktuátor osa *y* – OFF
  - Aktuátor osa *z* – ON
    - Výkon = 50 %
    - Režim působení = „DOWN“

Po uvedení systému do tohoto stavu byla aplikace SCC vypnuta. Po opětovném zapnutí došlo k synchronizaci webového GUI a SCC. K synchronizaci ale došlo pouze u senzorů a aktuátorů, u simulace letu HAPSu simulace neproběhla. Nejedná se o softwarovou chybu, ale o fakt, že v době testování nebyla tato funkcionalita implementovaná. Vzhledem k tomu, že tato funkce není důležitá pro správné fungování SCC, byla této chybě přiřazena malá priorita. V této verzi aplikace tedy tato funkcionalita chybí.

Po uvedení systému do dříve popsaného stavu, bylo naopak vypnuté webové GUI. Po opětovném zapnutí webového GUI se analogicky projevila chybějící implementace synchronizace stavu simulace letu HAPSu při ukončení jednoho uzlu z páru SCC, webové GUI.

Test X2

Funkce logování byla ověřena sbíráním dat během testování jiných funkcí a následnou kontrolou nasbíraných logů. Na *obr. 9.17* je vyobrazen obsah složky, kam se logy zapisují. Jak je možné vidět, většina logů má nenulovou velikost a jsou v nich uložené vyslané zprávy doplněné o časové razítko. Nulovou velikost mají pouze soubory *gps.json* a *main.log*. Soubor *gps.json* by měl obsahovat historii simulovaných GPS souřadnic virtuálního HAPSu, nicméně logování těchto dat bylo v kódu úmyslně zakázáno kvůli přílišné velikosti logu. Logování těchto dat je nicméně v případě potřeby funkční. Soubor *main.log* je využíván v případech zapnutí aplikace na pozadí pomocí startovacího skriptu *start.sh*. Jelikož aplikace byla startována manuálně pomocí spuštění hlavního souboru *main.py* v terminálu, nebyl tento logovací soubor využit.

Příklad obsahu logů je možné vidět v *obr. 9.2* a *obr. 9.3*.

```
❯ ls -la
total 92
drwxr-xr-x  3 pi pi  4096 May 12 00:00 .
drwxr-xr-x 13 pi pi  4096 May 12 00:32 ..
-rw-r--r--  1 pi pi   527 May 12 00:13 flight_actuators.json
-rw-r--r--  1 pi pi     0 May 12 00:00 gps.json
-rw-r--r--  1 pi pi 18641 May 12 00:32 humidity.json
-rw-r--r--  1 pi pi 17305 May 12 00:32 light.json
-rw-r--r--  1 pi pi     0 May 12 00:00 main.log
drwxr-xr-x  2 pi pi  4096 May 12 00:00 old_rotate
-rw-r--r--  1 pi pi   208 May 12 00:23 relay.json
-rw-r--r--  1 pi pi 18872 May 12 00:32 temperature.json
```

Obr. 9.17 *Obsah složky sloužící k ukládání logů*

Test X3

V hlavní složce aplikace SCC lze najít dva skripty – *start.sh* a *stop.sh*. Cílem skriptu *start.sh* je spustit aplikaci na pozadí pomocí příkazu *nohup*. Skript *stop.sh* poté spuštěnou aplikaci na pozadí dokáže zastavit. Testování proběhlo spuštěním těchto skriptů.

Při testování skriptu *start.sh* nastal při spuštění aplikace problém s nalezením určitých souborů v kódu aplikace. To bylo díky tomu, že aplikace nebyla spuštěna ze stejné složky, jako když byl manuálně spouštěný soubor *main.py* ze složky *src*. Díky použití relativních cest k určitým souborům pak došlo k této chybě.

Tato chyba byla vyřešena modifikací skriptu *start.sh* takovým způsobem, aby se před spuštěním aplikace nejdříve skript přesunul do složky *src* a spouštěl aplikaci až z této složky. Díky tomu byla zachována správnost relativních cest a aplikace se v pozadí spustila.

Skript *stop.sh* vyžadoval malou změnu korespondující s popsanou opravou chyby skriptu *start.sh*. Poté byla ověřena funkčnost skriptu *stop.sh*, při čemž nebyly nalezeny žádné nedostatky.

Test X4

Při testování ošetření vstupů nebyla shledáno žádné nečekané chování ani chyby. Opět byl k testování použit nástroj *mosquitto_pub*. Reakce aplikace byly buď analogické již popsaným případům, kdy byla chyba zachycena díky bloku „try – except“ nebo aplikace neplatný vstup zcela ignorovala.

9.2.6 Výsledky testů

Výsledky testování jsou shrnuty v *tab. 9.11* níže.

Tab. 9.11 *Výsledky testování*

Test	Výsledek	Poznámka
S1	**OK**	*Plně funkční*
S2	**OK**	*Plně funkční*
S3	**OK**	*Plně funkční*
S4	**OK**	*Plně funkční*
S5	**OK**	*Plně funkční*
S6	**OK**	*Plně funkční*
A1	**OK**	*Plně funkční*
H1	**OK**	*Plně funkční*
H2	**OK**	*Plně funkční*
H3	**OK**	*Plně funkční*
C1	**OK**	*Plně funkční*
C2	**OK**	*Plně funkční*
C3	**OK**	*Plně funkční*
C4	**OK**	*V reálném HAPSu není možné realizovat korekci pozice HAPSu pouze na základě dat z GPS (díky útoku "GPS spoofing")*
X1	**X / OK**	*Chybějící implementace pro synchronizaci stavu simulace letu HAPSu, zbytek funkční*
X2	**OK**	*Plně funkční*
X3	**X → OK**	*Plně funkční, chyba opravena*
X4	**OK**	*Nebyly nalezeny žádné problémy s ošetřením vstupů*

Kapitola 10
Možnosti dalšího vývoje ve zkoumané oblasti

Hlavní možností pro další vývoj je vylepšení navržené zabezpečené architektury HAPS. Nelze vyloučit, že při realizaci navržené architektury se projeví nedostatky návrhu, které nejsou v úrovni teoretického návrhu zřejmé. Dalším krokem vývoje je tedy realizace prezentovaného návrhu zabezpečené architektury HAPS, na kterou navazuje testování dle navržené metodiky v podkapitole 9.1. Na základě výsledků testování budou odstraněny identifikované problémy a navržena další vylepšení.

Vzhledem k rozsáhlosti problému byl vynechán detailní návrh určitých zařízení, například HAPS PC. Pro realizaci takového zařízení bude nutné nejprve vypracovat detailnější návrh, a to zejména z hlediska konfigurace jednotlivých bezpečnostních prvků, jako jsou Firewall, či IPS. Díky přílišné rozsáhlosti byl také zcela opominut návrh pozemní kontrolní stanice. Tato část je pro bezpečnost HAPSu kritická. Ve chvíli kdy dojde k napadení sítě pozemní kontrolní stanice, může útočník získat přímý přístup k HAPSu, resp. HAPS PC a snažit se jej kompromitovat, či jej jiným způsobem ovlivnit. Z pozice pozemní kontrolní stanice má totiž útočník daleko větší možnosti.

Další velkou příležitostí je vývoj a zdokonalení SCC, zejména části CPS IPS. Přestože testování primárně ověřilo dobrou funkčnost navrženého systému CPS IPS, který je založen na pravidlech, tak nelze tvrdit, že navržené řešení je zcela ideální ve všech směrech. S „rule based“ CPS IPS hrozí, že nebudou pokryty všechny hrozby. Vývoj by se tedy mohl zaměřit na jiné možnosti CPS IPS, například na CPS IPS založené na strojovém učení. Ideálním scénářem by pak mohla být vhodná kombinace různých technik pro detekci škodlivých požadavků.

Testování implementovaného PoC SCC prokázalo téměř plně funkční implementaci. Nejzávažnějším (ale stále minoritním) identifikovaným problémem byla chybějící implementace synchronizace stavu simulace letu HAPSu po výpadku jedné části z páru „SCC - webové GUI“ a po jejím opětovném zapnutí. Implementované PoC je možné rozšířit o další funkcionalitu, která nebyla součástí návrhu PoC SCC. Vhodné by bylo například doplnění o algoritmus pro analýzu senzorických dat. Dále se může jednat o implementaci logování vygenerovaných upozornění do souboru. Z výkonnostního hlediska by bylo zajímavé testování využití paralelizace aplikace. Využitá platforma Raspberry Pi 4 nabízí v procesoru 4 fyzická jádra. Paralelizace na tomto procesoru tedy nabízí prostor pro zvýšení výpočetní efektivity aplikace.

Kapitola 11
Závěr

Tato diplomová práce popisuje kyberfyzikální bezpečnost systému HAPS. Bezpečnost těchto systémů je vnímána jako závažný problém pro jejich nasazení do reálného provozu. Případný úspěšný kybernetický útok proti HAPSům může znamenat až jejich kompromitaci a převzetí kontroly nad tímto zařízením. V takových situacích by útočník teoreticky mohl poškodit některé části HAPSu, například motory pro kontrolu letu. Útočník by také mohl zneužít kontrolu nad HAPSem ke způsobení škod na majetku či k ohrožení zdraví osob. Díky závažnosti těchto rizik bylo nutné navrhnout taková opatření, která by znesnadnila útočníkův průnik do počítačových systémů HAPSu. Vzhledem k faktu, že kybernetickou bezpečnost nelze vnímat jako absolutní, bylo nutné navrhnout i taková opatření, která by minimalizovala možnosti útočníka v případě úspěšné kompromitace počítačových systému HAPSu.

Jako řešení výše zmíněných problémů byla ve čtvrté kapitole navržena základní architektura HAPS s posílenou kyberfyzikální bezpečností. V této kapitole byly nejdříve definovány bezpečnostní nároky na HAPS, na kterých se návrh architektury zakládal. Byl zde také představen koncept komponenty „Secure CPS Controller (SCC)“, která hraje klíčovou roli v zajištění kyberfyzikální bezpečnosti systému. SCC fyzicky odděluje senzory a aktuátory od řídicího počítače (HAPS PC). Funguje tedy jako jakýsi mezičlánek mezi senzory/aktuátory a HAPS PC. SCC přijímá řídicí požadavky od HAPS PC, přičemž jsou tyto požadavky filtrovány dle jejich nebezpečnosti pro HAPS částí SCC, která byla nazvaná CPS IPS. Úkolem SCC je primárně udržet HAPS a jeho součásti v bezpečné povolené množině stavů.

Navržená základní architektura HAPS s posílenou kyberfyzikální bezpečností byla podrobena analýze hrozeb. Nejprve byly namodelovány interakce systému do diagramu datových toků. Tento diagram byl dále použit jako základ pro analýzu hrozeb pomocí modelu STRIDE. Identifikované hrozby byly následně ohodnoceny pomocí modelu DREAD a rozděleny do kategorií „Low, Medium, High“.

V šesté kapitole byly navrženy bezpečnostní vylepšení základní architektury HAPS s posílenou kyberfyzikální bezpečností na základě identifikovaných hrozeb, přičemž při návrhu byla upřednostňována mitigace hrozeb s vysokou závažností. V této kapitole byl představen finální návrh zabezpečené architektury HAPS.

Sedmá kapitola se věnovala detailnímu teoretickému návrhu komponenty SCC. Byly definovány funkční a bezpečnostní požadavky. Dále byly popsány možné přístupy k řízení letu HAPSu a byl předložen návrh CPS IPS pro systémy HAPS. Poté byla navržena architektura komponenty SCC.

V osmé kapitole byla popsána implementace aplikace komponenty SCC na úrovni „proof of concept (PoC)“. Součástí PoC SCC bylo i webové GUI pro ovládání naprogramované aplikace, které sloužilo zejména pro demonstrační účely.

Devátá kapitola obsahovala teoretický návrh testů pro otestování navržené zabezpečené architektury HAPS. Dále se devátá kapitola zabývala návrhem testů pro otestování implementovaného PoC SCC. Implementované PoC SCC bylo poté na základě návrhu těchto testů otestováno, přičemž byla ověřena dobrá funkčnost konceptu SCC.

Desátá kapitola nabídla přehled možností dalšího vývoje a výzkumu v oblasti kyberfyzikální bezpečnosti HAPS.

Literatura

[1] GUAN, Xinping, Bo YANG, Cailian CHEN, Wenbin DAI a Yiyin WANG. A comprehensive overview of cyber-physical systems: from perspective of feedback system. *IEEE/CAA Journal of Automatica Sinica* [online]. 2016, **3**(1), 1-14 [cit. 2022-05-12]. ISSN 2329-9266. Dostupné z: doi:10.1109/JAS.2016.7373757

[2] BHRUGUBANDA, Meenakshi. A Review on Applications of Cyber Physical Systems. *IJISET - International Journal of Innovative Science, Engineering & Technology* [online]. 2015, **2**(6), 728-730 [cit. 2022-05-12]. ISSN 2348 – 7968. Dostupné z: https://ijiset.com/vol2/v2s6/IJISET_V2_I6_103.pdf

[3] *ITU Radio Regulations - Article 1: Terms and definitions*. 2020 edition. Geneva: ITU, 2020. Dostupné také z: https://life.itu.int/radioclub/rr/art1.pdf

[4] Stratosyst: Applications. In: *Stratosyst* [online]. [cit. 2022-05-12]. Dostupné z: http://www.stratosyst.com/#applications

[5] High-altitude pseudo-satellites. In: *The European Space Agency* [online]. [cit. 2022-05-12]. Dostupné z: https://www.esa.int/ESA_Multimedia/Images/2017/11/High-altitude_pseudo-satellites

[6] SKOROBOGATJKO, Alyona, Andrejs ROMANOVS a Nadezhda KUNICINA. State of the Art in the Healthcare Cyber-physical Systems/ Veselības aizsardzības kiberfizikālo sistēmu apskats Современное состояние медицинских кибер-физических систем. *Information Technology and Management Science* [online]. 2014, **17**(1), 126-131 [cit. 2022-05-12]. ISSN 2255-9094. Dostupné z: doi:10.1515/itms-2014-0019

[7] LEE, Edward A. Cyber-Physical Systems - a Concept Map. In: *Ptolemy.berkeley.edu* [online]. Berkeley, 2012 [cit. 2022-05-12]. Dostupné z: https://ptolemy.berkeley.edu/projects/cps/

[8] HORVÁTH, Imre a Bart GERRITSEN. CYBER-PHYSICAL SYSTEMS: CONCEPTS, TECHNOLOGIES AND IMPLEMENTATION PRINCIPLES. *TMCE* [online]. 2012, **1**, 19-36 [cit. 2022-05-12]. ISBN 978-90-5155-082-5. Dostupné z: https://www.researchgate.net/publication/229441298_CYBER-PHYSICAL_SYSTEMS_CONCEPTS_TECHNOLOGIES_AND_IMPLEMENTATION_PRINCIPLES

[9] AVILA, Risto. Embedded Software Programming Languages: Pros, Cons, and Comparisons of Popular Languages. In: *Embedded development talk* [online]. 2021 [cit. 2022-05-12]. Dostupné z: https://www.qt.io/embedded-development-talk/embedded-software-programming-languages-pros-cons-and-comparisons-of-popular-languages

[10] RAUTMARE, Sharvari a D. BHALERAO. MySQL and NoSQL database comparison for IoT application. *2016 IEEE International Conference on Advances in Computer Applications (ICACA)* [online]. IEEE, 2016, 235-238 [cit. 2022-05-12]. ISBN 978-1-5090-3769-8. Dostupné z: doi:10.1109/ICACA.2016.7887957

[11] Grafana Dashboard. In: *Grafana* [online]. [cit. 2022-05-12]. Dostupné z: https://grafana.com/

[12] *APPLICATION NOTE 3884: HOW FAR AND HOW FAST CAN YOU GO WITH RS-485?* [online]. In: . Maxim Integrated, 2014, s. 1-12 [cit. 2022-05-12]. Dostupné z: https://pdfserv.maximintegrated.com/en/an/AN3884.pdf

[13] BURATTI, Chiara, Andrea CONTI, Davide DARDARI a Roberto VERDONE. An Overview on Wireless Sensor Networks Technology and Evolution. *Sensors* [online]. 2009, **9**(9), 6869-6896 [cit. 2022-05-12]. ISSN 1424-8220. Dostupné z: doi:10.3390/s90906869

[14] MOHAMAD NOOR, Mardiana a Wan HASSAN. Current research on Internet of Things (IoT) security: A survey. *Computer Networks* [online]. 2019, **148**, 283-294 [cit. 2022-05-12]. ISSN 13891286. Dostupné z: doi:10.1016/j.comnet.2018.11.025

[15] The Human Factor: The Hidden Problem of Cybersecurity. In: *CYDEF* [online]. 2021 [cit. 2022-05-12]. Dostupné z: https://cydef.ca/blog/the-human-factor-the-hidden-problem-of-cybersecurity/

[16] BITCHKEI, Silvia. Common Exploits Found in a Penetration Test. In: *Hitachi Systems Security Inc.* [online]. 2019 [cit. 2022-05-12]. Dostupné z: https://hitachi-systems-security.com/common-exploits-found-in-a-penetration-test/

[17] CHASE, Richard a Douglas STEWART. Make Your Service Fail-Safe. *Sloan Management Review* [online]. 1994, **35**(3), 35-44 [cit. 2022-05-12]. Dostupné z: https://www.proquest.com/scholarly-journals/make-your-service-fail-safe/docview/224959576/se-2?accountid=44866

[18] HUANG, Min, Zhen LIU a Yang TAO. Mechanical fault diagnosis and prediction in IoT based on multi-source sensing data fusion. *Simulation Modelling Practice and Theory* [online]. 2020, **102** [cit. 2022-05-12]. ISSN 1569190X. Dostupné z: doi:10.1016/j.simpat.2019.101981

[19] CARDENAS, Alvaro. *Cyber-Physical Systems Security Knowledge Area* [online]. Issue 1.0. Santa Cruz: University of California, 2019 [cit. 2022-05-12]. Dostupné z: https://www.cybok.org/media/downloads/Cyber-Physical_Systems_Security_issue_1.0.pdf

[20] KROTOFIL, Marina, Alvaro CÁRDENAS, Jason LARSEN a Dieter GOLLMANN. Vulnerabilities of cyber-physical systems to stale data—Determining the optimal time to launch attacks. *International Journal of Critical Infrastructure Protection* [online]. 2014, **7**(4), 213-232 [cit. 2022-05-12]. ISSN 18745482. Dostupné z: doi:10.1016/j.ijcip.2014.10.003

[21] MALEH, Yassine, Mohammad SHOJAFAR, Ashraf DARWISH a Abdelkrim HAQIQ. *Cybersecurity and Privacy in Cyber Physical Systems* [online]. CRC Press, 2019 [cit. 2022-05-12]. ISBN 9781138346673. Dostupné z: https://www.researchgate.net/figure/tree-diagram-of-attacks-and-threats-on-cyber-physical-systems-technologies_fig1_329519489

[22] SUGAWARA, Takeshi, Benjamir CYR, Sara RAMPAZZI, Daniel GENKIN a Kevin FU. *Light Commands: Laser-Based Audio Injection Attacks on Voice-Controllable Systems* [online]. 2019 [cit. 2022-05-12]. Dostupné z: https://lightcommands.com/

[23] Tesla Model S and Model 3 vulnerable to GNSS spoofing attacks. In: *GPS World* [online]. [cit. 2022-05-12]. Dostupné z: https://www.gpsworld.com/tesla-model-s-and-model-3-vulnerable-to-gnss-spoofing-attacks/

[24] SELVARAJ, Jayaprakash, Gökçen DAYANıKLı, Neelam GAUNKAR, David WARE, Ryan GERDES a Mani MINA. Electromagnetic Induction Attacks Against Embedded Systems. *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* [online]. New York, NY, USA: ACM, 2018, 499-510 [cit. 2022-05-12]. ISBN 9781450355766. Dostupné z: doi:10.1145/3196494.3196556

[25] MITCHELL, Robert a Ing-Ray CHEN. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys* [online]. 2014, **46**(4), 1-29 [cit. 2022-05-12]. ISSN 0360-0300. Dostupné z: doi:10.1145/2542049

[26] ERBEN, Lukáš. Příchod hackerů: příběh Stuxnetu. In: *Root.cz* [online]. 2014 [cit. 2022-05-12]. Dostupné z: https://www.root.cz/clanky/prichod-hackeru-pribeh-stuxnetu/

[27] What Is Stuxnet?. In: *Trellix* [online]. [cit. 2022-05-13]. Dostupné z: https://www.trellix.com/en-us/security-awareness/ransomware/what-is-stuxnet.html

[28] KARNOUSKOS, Stamatis. Stuxnet worm impact on industrial cyber-physical system security. *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society* [online]. IEEE, 2011, 4490-4494 [cit. 2022-05-13]. ISBN 978-1-61284-972-0. Dostupné z: doi:10.1109/IECON.2011.6120048

[29] ZETTER, Kim. Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid. In: *WIRED* [online]. 2016 [cit. 2022-05-13]. Dostupné z: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/

[30] SZPORER, Ryan. HAPS Take Off in Connectivity Game Despite Challenges. In: *6G World* [online]. 2021 [cit. 2022-05-13]. Dostupné z: https://www.6gworld.com/exclusives/haps-take-off-in-connectivity-game-despite-challenges/

[31] MANN, Adam, Tereza PULTAROVA a Elizabeth HOWELL. Starlink: SpaceX's satellite internet project. In: *SPACE.com* [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://www.space.com/spacex-starlink-satellites.html

[32] KARABULUT KURT, Gunes, Mohammad KHOSHKHOLGH, Safwan ALFATTANI, Ahmed IBRAHIM, Tasneem DARWISH, Md ALAM, Halim YANIKOMEROGLU a Abbas YONGACOGLU. A Vision and Framework for the High Altitude Platform Station (HAPS) Networks of the Future. *IEEE Communications Surveys & Tutorials* [online]. 2021, **23**(2), 729-779 [cit. 2022-05-13]. ISSN 1553-877X. Dostupné z: doi:10.1109/COMST.2021.3066905

[33] What is multi-access edge computing?. In: *JUNIPER Networks* [online]. [cit. 2022-05-13]. Dostupné z: https://www.juniper.net/us/en/research-topics/what-is-multi-access-edge-computing.html

[34] YOKOTANI, Tetsuya a Yuya SASAKI. Comparison with HTTP and MQTT on required network resources for IoT. *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)* [online]. IEEE, 2016, 1-6 [cit. 2022-05-13]. ISBN 978-1-5090-0744-8. Dostupné z: doi:10.1109/ICCEREC.2016.7814989

[35] AZAD, Usama. VLAN Hopping Attack and Mitigation. In: *Linux Hint* [online]. 2021 [cit. 2022-05-13]. Dostupné z: https://linuxhint.com/vlan-hopping-attack-mitigation/

[36] ABUSAIMEH, Hesham. Virtual Machine Escape in Cloud Computing Services. *International Journal of Advanced Computer Science and Applications* [online]. 2020, **11**(7) [cit. 2022-05-13]. ISSN 21565570. Dostupné z: doi:10.14569/IJACSA.2020.0110743

[37] DRAKE, Victoria. Threat Modeling. In: *OWASP* [online]. [cit. 2022-05-13]. Dostupné z: https://owasp.org/www-community/Threat_Modeling

[38] CONKLIN, Larry a Victoria DRAKE. Threat Modeling Process. In: *OWASP* [online]. [cit. 2022-05-13]. Dostupné z: https://owasp.org/www-community/Threat_Modeling_Process

[39] HEWKO, Alex. STRIDE Threat Modeling: What You Need to Know. In: *Software Secured* [online]. [cit. 2022-05-13]. Dostupné z: https://www.softwaresecured.com/stride-threat-modeling/

[40] KHAN, Salman A. A STRIDE Model based Threat Modelling using Unified and-Or Fuzzy Operator for Computer Network Securit. *International Journal of Computing and Network Technology* [online]. 2017, **5**(1), 13-20 [cit. 2022-05-13]. ISSN 2210-1519. Dostupné z: doi:10.12785/ijcnt/050103

[41] What Is the DREAD Cybersecurity Model?. In: *Logix* [online]. 2019 [cit. 2022-05-13]. Dostupné z: https://logixconsulting.com/2019/12/18/what-is-the-dread-cybersecurity-model/

[42] CZAGAN, Dawid. Qualitative risk analysis with the DREAD model. In: *Infosec Institute* [online]. 2014 [cit. 2022-05-13]. Dostupné z: https://resources.infosecinstitute.com/topic/qualitative-risk-analysis-dread-model/

[43] Microsoft Threat Modeling Tool. In: *Microsoft* [online]. 2020 [cit. 2022-05-13]. Dostupné z: https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

[44] KHAN, Rafiullah, Kieran MCLAUGHLIN, David LAVERTY a Sakir SEZER. STRIDE-based threat modeling for cyber-physical systems. *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)* [online]. IEEE, 2017, 1-6 [cit. 2022-05-13]. ISBN 978-1-5386-1953-7. Dostupné z: doi:10.1109/ISGTEurope.2017.8260283

[45] PARK, Youngseok, Yunmok SON, Hocheol SHIN, Dohyun KIM a Yongdae KIM. This ain't your dose: Sensor Spoofing Attack on Medical Infusion Pump. *Usenix* [online]. [cit. 2022-05-13]. Dostupné z: https://www.usenix.org/system/files/conference/woot16/woot16-paper-park_0.pdf

[46] ZANGVIL, Yoav. Research on GPS Resiliency & Spoofing Mitigation Techniques Across Applications. In: *GPS.gov* [online]. 2019 [cit. 2022-05-13]. Dostupné z: https://www.gps.gov/governance/advisory/meetings/2019-06/zangvil.pdf

[47] HAIDER, Zeeshan a Shehzad KHALID. Survey on effective GPS spoofing countermeasures. *2016 Sixth International Conference on Innovative Computing Technology (INTECH)* [online]. IEEE, 2016, 573-577 [cit. 2022-05-13]. ISBN 978-1-5090-2000-3. Dostupné z: doi:10.1109/INTECH.2016.7845038

[48] JACKSON, Ambler. Compare TLS 1.3 and TLS 1.2 Certificates: Which is Stronger?. In: *Venafi* [online]. [cit. 2022-05-13]. Dostupné z: https://www.venafi.com/blog/why-tls-13-radically-different-tls-12

[49] *The Transport Layer Security (TLS) Protocol Version 1.3: RFC 8446*. Internet Engineering Task Force (IETF), 2018. ISSN: 2070-1721. Dostupné také z: https://datatracker.ietf.org/doc/html/rfc8446

[50] TLS Symmetric Crypto. In: *Imperial Violet* [online]. 2014 [cit. 2022-05-13]. Dostupné z: https://www.imperialviolet.org/2014/02/27/tlssymmetriccrypto.html

[51] *Stratosphere IPS* [online]. [cit. 2022-05-13]. Dostupné z: https://www.stratosphereips.org/stratosphere-ips-suite

[52] GROVER, Kanika, Alvin LIM a Qing YANG. Jamming and anti-jamming techniques in wireless networks: a survey. *International Journal of Ad Hoc and Ubiquitous Computing* [online]. 2014, **17**(4), 197-215 [cit. 2022-05-13]. ISSN 1743-8225. Dostupné z: doi:10.1504/IJAHUC.2014.066419

[53] RAZI, Muhammad A. a Kuriakose ATHAPPILLY. A comparative predictive analysis of neural networks (NNs), nonlinear regression and classification and regression tree (CART) models. *Expert Systems with Applications* [online]. 2005, **29**(1), 65-74 [cit. 2022-05-13]. ISSN 09574174. Dostupné z: doi:10.1016/j.eswa.2005.01.006

[54] BOKONDA, Patrick Loola, Khadija OUAZZANI-TOUHAMI a Nissrine SOUISSI. Predictive analysis using machine learning: Review of trends and methods. *2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)* [online]. IEEE, 2020, 1-6 [cit. 2022-05-13]. ISBN 978-1-6654-2222-2. Dostupné z: doi:10.1109/ISAECT50560.2020.9523703

[55] NAMUDURI, Srikanth, Barath Narayanan NARAYANAN, Venkata Salini Priyamvada DAVULURU, Lamar BURTON a Shekhar BHANSALI. Review—Deep Learning Methods for Sensor Based Predictive Maintenance and Future Perspectives for Electrochemical Sensors. *Journal of The Electrochemical Society* [online]. 2020, **167**(3) [cit. 2022-05-13]. ISSN 0013-4651. Dostupné z: doi:10.1149/1945-7111/ab67a8

[56] RPI4-MODBP-4GB. In: *Farnell* [online]. [cit. 2022-05-13]. Dostupné z: https://cz.farnell.com/raspberry-pi/rpi4-modbp-4gb/raspberry-pi-4-model-b-4gb/dp/3051887

[57] Paho-MQTT. In: *Eclipse Foundation* [online]. [cit. 2022-05-13]. Dostupné z: https://www.eclipse.org/paho/index.php?page=clients/python/index.php

[58] MinimalModbus. In: *Github* [online]. [cit. 2022-05-13]. Dostupné z: https://github.com/pyhys/minimalmodbus

[59] Eclipse Mosquitto: An open source MQTT broker. In: *Mosquitto* [online]. [cit. 2022-05-13]. Dostupné z: https://mosquitto.org/

[60] Node-RED: Low-code programming for event-driven applications. In: *Node-RED* [online]. [cit. 2022-05-13]. Dostupné z: https://nodered.org/

[61] Node-red-dashboard: A set of dashboard nodes for Node-RED. In: *Node-RED* [online]. [cit. 2022-05-13]. Dostupné z: https://flows.nodered.org/node/node-red-dashboard

[62] SimpleModbus NG. In: *Github* [online]. [cit. 2022-05-13]. Dostupné z: https://github.com/angeloc/simplemodbusng

[63] Adafruit HTU21D-F Humidity/Temp Sensor for Arduino. In: *Github* [online]. [cit. 2022-05-13]. Dostupné z: https://github.com/adafruit/Adafruit_HTU21DF_Library

[64] BH1750. In: *Github* [online]. [cit. 2022-05-13]. Dostupné z: https://github.com/claws/BH1750

[65] Vývojový kit Arduino Uno R3 - 100% klon. In: *GM Electronic* [online]. [cit. 2022-05-13]. Dostupné z: https://www.gme.cz/100-kompatibilni-klon-arduino-uno-r3-usb-b-atmega16u2

[66] Převodník TTL na RS 485. In: *Drátek.cz* [online]. [cit. 2022-05-13]. Dostupné z: https://dratek.cz/arduino/985-prevodnik-ttl-na-rs-485.html

[67] Převodník USB na RS485 chip CH340C. In: *Drátek.cz* [online]. [cit. 2022-05-13]. Dostupné z: https://dratek.cz/arduino/1171-prevodnik-usb-na-rs485-chip-ch340c.html

[68] IIC I2C Senzor Teploty a Vlhkosti HTU21D. In: *Drátek.cz* [online]. [cit. 2022-05-13]. Dostupné z: https://dratek.cz/arduino/1487-iic-i2c-senzor-teploty-a-vlhkosti-htu21d.html

[69] Měření intenzity světla BH1750. In: *Drátek.cz* [online]. [cit. 2022-05-13]. Dostupné z: https://dratek.cz/arduino/902-mereni-intenzity-svetla-bh1750.html

[70] Testssl.sh. In: *Github* [online]. [cit. 2022-05-13]. Dostupné z: https://github.com/drwetter/testssl.sh

[71] LinPEAS: Linux Privilege Escalation Awesome Script. In: *Github* [online]. [cit. 2022-05-13]. Dostupné z: https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS