SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta elektrotechniky a informatiky

Ing. Ján Šefčík

Autoreferát dizertačnej práce

Otvorené technológie pre vzdialený experiment

na získanie akademického titulu
„doktor“ („philosophiae doctor“, v skratke „PhD.“)

v doktorandskom študijnom programe:	mechatronické systémy
v študijnom odbore:	kybernetika
forma štúdia:	denná

Miesto a dátum: Bratislava, 9.7.2025

Dizertačná práca bola vypracovaná na:

Ústav automobilovej mechatroniky
Fakulta elektrotechniky a informatiky STU v Bratislave
Ilkovičova 3, 84104 Bratislava

Predkladateľ:

Ing. Ján Šefčík
Ústav automobilovej mechatroniky
Fakulta elektrotechniky a informatiky STU v Bratislave
Ilkovičova 3, 841 04 Bratislava

Školiteľ:

prof. Ing. Katarína Žáková, PhD.
Ústav automobilovej mechatroniky
Fakulta elektrotechniky a informatiky STU v Bratislave
Ilkovičova 3, 841 04 Bratislava

Oponenti:

Ing. Ivana Budinská, PhD.
Ústav informatiky
Slovenská akadémia vied
Dúbravská cesta 9, 845 07 Bratislava

doc. Ing. Ján Vachálek, PhD.
Ústav automatizácie, informatizácie a merania
Strojnícka fakulta STU v Bratislave
Nám. slobody 17, 812 31 Bratislava

Autoreferát bol rozoslaný dňa:

Obhajoba dizertačnej práce sa bude konať dňa: 27.8.2025 **o** 10:30 hod., **na** Fakulte elektrotechniky a informatiky STU v Bratislave, Ilkovičova 3, 841 04 Bratislava – Karlova Ves, v miestnosti D101.

.....

prof. Ing. Vladimír Kutiš, PhD.

Abstrakt

Dizertačná práca sa zaoberá návrhom a implementáciou otvorených technológií pre vzdialené experimentovanie v online laboratóriách zameraných na riadenie mechatronických systémov v reálnom čase. Na základe analýzy súčasného stavu vývoja online laboratórií sú identifikované kľúčové technické a metodické výzvy spojené s integráciou experimentov a riadením hardvérovo-softvérových komponentov prostredníctvom webových technológií. V práci je predstavená unifikovaná architektúra vzdialeného laboratória využívajúca moderné softvérové a hardvérové prostriedky, ktorá je navrhnutá s dôrazom na využitie otvorených štandardov, modulárnosť a škálovateľnosť.

Súčasťou riešenia je zovšeobecnenie procesov integrácie experimentálnych zariadení do online laboratória, ktoré umožňuje efektívnu správu a opätovnú použiteľnosť komponentov pri tvorbe nových experimentov. Pre overenie navrhovanej architektúry bol realizovaný a experimentálne overený nový vzdialený experiment riadenia mechatronického systému využívajúceho senzorovú redundanciu. Na základe skúseností z implementácie bol zovšeobecnený metodický postup návrhu a integrácie experimentov s ohľadom na ich čo najväčšiu otvorenosť voči rôznym softvérovým platformám a podporu vzdialenej spolupráce.

Kľúčové slová

vzdialené riadenie, mechatronický systém, open-source, otvorený softvér, slobodný softvér

Abstract

This dissertation explores the design and implementation of open technologies for remote experimentation in online laboratories, with a focus on real-time control of mechatronic systems. The work begins with an analysis of the current state of development of online laboratory platforms, emphasizing challenges and solutions in integrating real-time control of mechatronic devices.

A unified architecture is proposed, leveraging modern software and hardware tools to enable the control of mechatronic systems using open-source technologies. The dissertation further generalizes the integration process of experiments into an online laboratory framework, providing guidelines for modular and scalable implementations.

As part of the research, a novel experiment involving the control of a mechatronic system with sensor redundancy was designed, implemented, and validated in the proposed architecture. The outcomes demonstrate the feasibility and advantages of using open technologies for building flexible and accessible remote laboratory environments.

Keywords

remote control, mechatronic system, open-source, free software

Obsah

Úvod

Rozvoj moderných technológií, najmä v oblasti internetu, mikropočítačov a otvorených softvérových platforiem, umožnil vznik konceptu vzdialených a online laboratórií, ktoré sa stávajú neoddeliteľnou súčasťou výučby a výskumu v technických odboroch. Tieto laboratóriá umožňujú používateľom realizovať experimenty so skutočnými fyzikálnymi zariadeniami na diaľku, čím znižujú bariéry prístupu k drahému a zložitému vybaveniu a poskytujú nové možnosti pre interaktívne vzdelávanie aj výskumné aplikácie.

V oblasti riadenia mechatronických systémov, ktoré spájajú mechanické, elektronické a softvérové komponenty, je vzdialené experimentovanie obzvlášť prínosné. Umožňuje overovanie regulačných algoritmov v reálnych podmienkach a porovnávanie výsledkov so simulačnými modelmi, čím podporuje proces návrhu a ladenia systémov v reálnom čase. Súčasne však prináša výzvy v oblasti architektúry systémov, bezpečnosti komunikácie, kompatibility hardvéru a softvéru, ako aj škálovateľnosti a udržateľnosti celého riešenia.

Existuje niekoľko architektonických prístupov k návrhu vzdialených laboratórií, ako sú peer-to-peer riešenia, ktoré ponúkajú priamu komunikáciu medzi zariadeniami, architektúry založené na IoT, ktoré využívajú internet vecí pre prepájanie experimentov alebo SOA (Service-Oriented Architecture) a cloudové systémy, ktoré poskytujú vysokú mieru modularity a rozšíriteľnosti. V kontexte práce je kľúčová aj analýza výhod a nevýhod týchto prístupov vzhľadom na potreby riadenia mechatronických systémov.

Navrhované riešenie a postupy kladú dôraz na modulárnosť, škálovateľnosť a otvorenosť, ktoré umožňujú jednoduchú integráciu rôznych experimentov a znižujú závislosť od proprietárnych technológií.

Táto práca predstavuje metodológiu návrhu a implementácie vzdialených experimentov, ktoré reagujú na aktuálne potreby vzdelávacích a výskumných inštitúcií a podporujú rozvoj otvorených a udržateľných platforiem pre riadenie mechatronických systémov.

1 Súčasný stav problematiky

1.1 Typy experimentov

Existuje viacero perspektív, z ktorých je možné analyzovať a vnímať laboratórne experimenty. Pre lepšiu predstavu je vhodné vopred definovať a klasifikovať oblasť experimentov, ich vzťah voči laboratóriám a ich používateľom. Konkrétny experiment môžeme definovať podľa troch základných kritérií [1]:

1. typ interakcie používateľa s experimentom,
2. typ experimentu z hľadiska jeho charakteru, t.j. či ide o skutočné zariadenie alebo virtuálny, resp. matematický model,
3. typ umiestnenia experimentu a jeho používateľa.

Vzhľadom na prvé kritérium môžeme definovať dva spôsoby interakcie používateľa s experimentom:

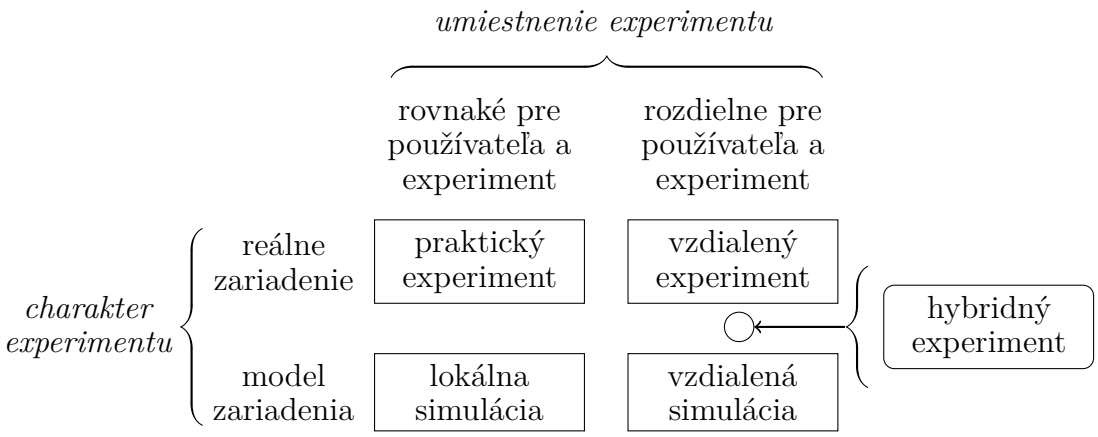
1. priame ovládanie a interakcia s reálnymi zariadeniami a vybavením v štandardnom fyzickom laboratóriu,
2. ovládanie reálnych zariadení a vybavenia laboratória prostredníctvom sprostredkovaného počítačového rozhrania.

Ak sa skombinujú zostávajúce dve kritériá, výsledkom sú štyri druhy možných experimentov (viď. obr. 1). Vzhľadom na povahu experimentu je možné rozlišovať experimenty podľa toho, či zahŕňajú skutočné alebo simulované/emulované modely zariadení a vybavenia. Z iného pohľadu je možné rozoznať dve možné umiestnenia, resp. lokality, kde sa pri experimentovaní nachádza používateľ a zariadenie, a to na rovnakom mieste alebo na rôznych miestach [1].

V poslednej dobe je možné zaznamenať aj úsilie o výskum pokročilých tzv. hybridných experimentov. V kontexte vzdialených laboratórií ide väčšinou o kombináciu vzdialených a virtuálnych experimentov, ktoré nielen napodobňujú možnosti reálneho laboratória, ale poskytujú aj nové funkcie a výhody, ktoré nemusia byť v skutočnom laboratóriu dostupné [2].

1.2 Typy laboratórií vzdialených experimentov

Z charakteristiky experimentov v ľavom stĺpci na obr. 1 [1, 2]. vyplývajú tri typy laboratórií, ktoré je možné definovať pri experimentovaní na diaľku, resp. v prostredí internetu:



Obr. 1: Charakteristika experimentov v závislosti od ich povahy a lokality

1. Virtuálne laboratóriá predstavujú simulované prostredie na realizáciu experimentov, ktoré umožňuje používateľom bezpečne skúmať rôzne podmienky bez rizika poškodenia zariadení alebo ohrozenia zdravia. Umožňujú opakované vykonávanie experimentov bez opotrebovania vybavenia či vyčerpania zdrojov. Kľúčovým prvkom je simulačný alebo fyzikálny softvér, ktorý modeluje reálne systémy a umožňuje interakciu prostredníctvom natívnych aplikácií alebo webového rozhrania s podporou 2D/3D vizualizácie, virtuálnej či zmiešanej reality. Ich výhodou je vysoká škálovateľnosť, jednoduchá implementácia a nenáročnosť na fyzické priestory. Virtuálne laboratóriá sú aj naďalej predmetom intenzívneho výskumu a rozvoja, čo dokazujú napr. publikácie [3–6].

2. Vzdialené laboratóriá umožňujú prístup k reálnym fyzikálnym experimentom a zariadeniam prostredníctvom počítačovej siete alebo internetu. Na rozdiel od virtuálnych laboratórií poskytujú autentické experimentálne dáta a vystavujú používateľov reálnym podmienkam, ako sú oneskorenia či meracie nepresnosti. Ich hlavnou výhodou je verná reprezentácia reálneho prostredia, čím vytvárajú realistickú atmosféru a môžu dosahovať výsledky porovnateľné, či dokonca lepšie než tradičné laboratóriá. Ponúkajú flexibilitu v plánovaní meraní, prekonávajú geografické bariéry a zvyšujú efektivitu výučby, čo potvrdzujú aj viaceré výskumy [7–10].

3. Hybridné laboratóriá predstavujú kombináciu vzdialených laboratórií s reálnymi zariadeniami a virtuálnych laboratórií so simuláciami alebo prvkami zmiešanej reality [1]. Súčasný výskum sa zameriava na vývoj takýchto inovatívnych modelov, ktoré spájajú výhody oboch prístupov a prinášajú nové funkcie presahujúce možnosti tradičných laboratórií. Medzi ich hlavné prínosy patrí rozšírenie dostupnosti experimentov a možnosť prípravy používateľov vo virtuálnom prostredí, čím sa

znižujú riziká pri práci s reálnym vybavením. Hybridné laboratóriá umožňujú aj dynamické prepínanie medzi simuláciou a reálnym zariadením, napríklad pri nedostupnosti fyzických zdrojov, alebo súbežné využitie oboch prístupov na porovnanie správania modelu so skutočným systémom [1, 2].

1.3 Architektúry vzdialených laboratórií

Odborná literatúra v súčasnosti neposkytuje štandardy alebo jednoznačné popisy architektúr, na základe ktorých je možné rozdeliť konkrétne riešenia vzdialených laboratórií alebo ich implementovať. Z vlastných pozorovaní tiež usudzujeme, že tvorcovia vzdialených laboratórií striktne nedodržiavajú jednotný koncept, ale snažia sa o čo najširší prienik a spojenie viacerých kategórií, čím vzniká čoraz viac hybridných riešení.

V nasledujúcich kapitolách sa zameriame predovšetkým na opis architektúr vzdialených laboratórií, ktoré sú priamo relevantné v kontexte navrhovaného riešenia hybridnej architektúry. Ostatné existujúce architektúry vzdialených laboratórií budú uvedené len stručne v rámci prehľadu s cieľom poskytnúť základný kontext dostupných možností.

Peer-to-peer architektúra

Architektúra *Peer-to-peer* predstavuje jednoduché riešenie pre vzdialené experimenty, kde každý uzol (peer) je počítač alebo pracovná stanica pripojená priamo k experimentálnemu zariadeniu cez káblové rozhranie. Na tomto počítači beží simulačný softvér, ktorý zabezpečuje zadávanie vstupov, čítanie výstupov a komunikáciu s hardvérom, zároveň slúži ako centrálne úložisko dát a nastavení. Výpočet riadiaceho zásahu môže prebiehať buď priamo na pracovnej stanici, alebo vo vnútri mikrokontroléra zariadenia. Uzly komunikujú navzájom cez sieť pomocou protokolov vzdialeného prístupu (napr. SSH, VNC, RDP), čo umožňuje používateľom spúšťať vlastné experimenty, zdieľať ich a vykonávať experimenty poskytované ostatnými v rámci siete [11–14].

Architektúra založená na IoT

Architektúra založená na IoT umožňuje komunikáciu medzi používateľom a viacerými rôznymi experimentálnymi zariadeniami cez internet alebo lokálnu sieť. Prepojenie je realizované káblovým alebo bezdrôtovým pripojením a komunikácia medzi uzlami využíva štandardné sieťové protokoly TCP/IP alebo UDP. Klientske zariadenie slúži ako platforma pre simulačný softvér a ukladanie experimentálnych dát. Výpočet riadiaceho zásahu môže prebiehať buď v simulačnom softvéri, samostatnom procese na klientovi, alebo priamo v mikrokontroléri zariadenia [15–20].

Architektúra klient-server

Architektúra klient-server je bežne používaná pri webových aplikáciách a vhodná aj pre vzdialené experimenty. Používateľ zadáva parametre experimentu cez webové rozhranie (klient), ktoré ich odosiela cez API na server. Server so simulačným softvérom vykonáva experimenty buď simuláciou modelov, alebo priamo na reálnom zariadení. Výpočty riadenia môžu prebiehať na serveri, ako samostatný program alebo priamo v zariadení (napr. mikrokontroléri).

Komunikácia klient-server prebieha cez štandardné webové protokoly (HTTP, WebSocket) a reálne zariadenie komunikuje so serverom cez sériové rozhrania ako USB, UART/USART alebo ADC/DAC prevodníky.

Klasická architektúra klient-server je v kontexte vzdialených laboratórií základom pre viacero riešení vďaka svojej jednoduchosti implementácie a možnostiam horizontálneho aj vertikálneho škálovania, čo dokazujú aj publikácie [21–24]. Jednoduchá škálovateľnosť architektúry klient-server tiež uľahčuje možnosti výskumu a vývoja v oblasti vzdialenej spolupráce, kooperácie a skupinovej práce na vzdialených experimentoch, čomu sa venujú publikácie [25–29].

SOA - Service-Oriented architektúra

Service-oriented architektúra rozdeľuje komplexnú funkcionalitu vzdialeného laboratória na samostatné, nezávislé služby (moduly), ako napríklad simulačný softvér, návrh diagramov, spracovanie dát, vizualizáciu či samotné experimentálne zariadenia. Výpočet riadiaceho zásahu je tiež implementovaný ako samostatná služba. Všetky služby zdieľajú spoločné úložisko údajov, ktoré zabezpečuje integritu a jednotný prístup k dátam. Kľúčovým prvkom je poskytovateľ služieb (service provider/container), ktorý spravuje registráciu a komunikáciu so službami cez ich rozhrania.

Tento prístup je inovatívny a menej preskúmaný v oblasti vzdialených laboratórií, pričom existujú implementácie založené na mikroslužbách, FaaS (Functions as a Service) a LaaS (Laboratory as a Service), ktoré podporujú kooperáciu používateľov a efektívnu interakciu s meracími prístrojmi [30–33].

Architektúra založená na cloude

Koncept cloudovej architektúry pre vzdialené laboratóriá spočíva vo virtualizácii podstatnej časti softvérového vybavenia laboratórnej infraštruktúry do cloudového prostredia.

Koncový používateľ má k dispozícii grafické používateľské rozhranie cloudovej služby v podobe webovej aplikácie, s ktorým interaguje prostredníctvom počítača, laptopu, smartfónu alebo tabletu s pripojením na internet. Cloudová platforma na základe požia-

daviek používateľov na experiment dynamicky vytvára špecifické prostredia, ktoré môžu byť implementované ako virtuálne stroje alebo kontajnery služieb. Virtuálne prostredia implementujú simulačný softvér a funkcie pre komunikáciu s reálnym experimentálnym zariadením.

Funkcia výpočtu akčného zásahu pre riadenie reálneho zariadenia počas experimentu môže byť súčasťou simulačného softvéru vo virtuálnom prostredí, môže byť implementovaná ako samostatná služba, resp. program v rámci virtuálneho prostredia alebo sa výpočet akčného zásahu regulátora môže realizovať mimo cloudovej platformy priamo na mikrokontroléri experimentálneho zariadenia [34, 35].

Hybridná architektúra

Implementácia konceptu hybridnej architektúry vzdialeného laboratória spočíva v kombinácii dvoch alebo viacerých spomínaných konceptov. Príklad takého riešenia je architektúra, ktorá kombinuje koncepty cloudovej architektúry a architektúry klient-server.

Používateľ komunikuje s centrálnym manažment serverom cez webovú alebo natívnu aplikáciu. Tento server môže byť založený na komerčných cloudových platformách (napr. AWS, GCP) alebo na vlastnom cloudovom riešení a zabezpečuje správu používateľov, ukladanie experimentálnych dát, plánovanie a rezervácie experimentov, ako aj spracovanie a vizualizáciu výsledkov. Manažment server zároveň koordinuje viaceré experimentálne servery.

Experimentálny server obsahuje simulačný softvér, lokálne úložisko dát a API alebo webovú službu na komunikáciu s centrálnym serverom. Reálne experimentálne zariadenia sú s ním prepojené káblovými rozhraniami ako USB, UART/USART alebo ADC/DAC prevodníkmi. Výpočet akčného zásahu regulátora môže prebiehať buď v simulačnom softvéri, ako samostatná funkcia na experimente serveri, alebo priamo na mikrokontroléri zariadenia.

V praxi sa koncept hybridnej architektúry vzdialeného laboratória objavuje v systémoch opísaných v publikáciách [36–39]

1.4 Simulačné prostredia

Simulačné prostredia možno kategorizovať do dvoch základných skupín:

- komerčné softvérové balíky (napr. LabView, Dymola, MATLAB/Simulink),
- open-source nástroje, ktoré umožňujú modifikáciu zdrojového kódu v rámci licencie.

V rámci našej práce sa zameriavame na prostredia podporujúce grafický návrh riadiacich algoritmov. Ako komerčné simulačné prostredie pre porovnanie bol zvolený softvér MATLAB/Simulink. Program pysimCoder bol vybraný ako jednoduchá, open-source alternatíva k aplikácii Simulink. V porovnaní s alternatívnymi open-source riešeniami (napr. Scilab/Xcos) poskytuje viacero výhod z pohľadu požiadaviek práce:

- Integrácia s Python ekosystémom – MATLAB poskytuje rozhranie *Engine API for Python*, čo umožňuje spúšťať a riadiť simulácie priamo z prostredia Pythonu. Keďže pysimCoder je implementovaný v Pythone a podporuje komunikáciu cez Python API, jeho použitie zabezpečuje jednotný spôsob interakcie so simulačnými prostrediami aj experimentálnym hardvérom.
- Program pysimCoder podporuje jednoduché doplnenie vlastných blokov alebo algoritmov, ktoré možno jednoducho definovať pomocou jazykov Python a C alebo vygenerovať ako spustiteľný program pre *embedded* zariadenia. Tento prístup umožňuje užšiu interakciu s cieľovou platformou, ktorá je potrebná pri experimentovaní s mechatronickými systémami v reálnom čase.
- Kľúčovou výhodou pysimCoderu je podpora *code generation* pre mikrokontroléry (napr. platformy STM32), čo umožňuje zostavenie blokovej schémy a jej kompiláciu priamo do spustiteľného kódu pre cieľové zariadenie. Tento prístup je obzvlášť výhodný pri návrhu vzdialených experimentov, kde časť riadiacich algoritmov beží priamo na experimentálnom zariadení.

2 Návrh unifikovanej architektúry pre vzdialené experimentovanie

Pri návrhu unifikovanej architektúry pre vzdialené experimentovanie vychádzame z poznatkov prác [40–42]. Aktuálne riešenie, ktoré budeme ďalej v práci nazývať ako „architektúra IOLab“, je navrhnuté ako viacvrstvová – hybridná architektúra. Prezentovaný koncept v základe spája prvky cloudovej architektúry a klasickej architektúry klient-server.

2.1 Opis architektúry

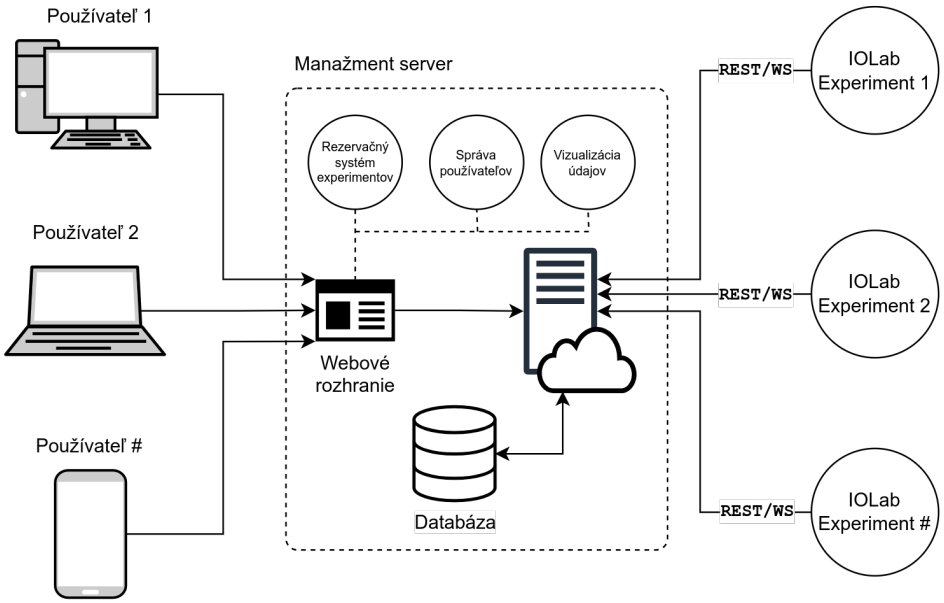
Hybridný koncept unifikovanej architektúry pozostáva z dvoch samostatných častí: server pre správu prístupu k experimentom (klientsky server) a server pre laboratórne experimenty (experimentový server) (obr. 4). Architektúra je navrhnutá s dôrazom na modularitu a otvorenosť použitých technológií s cieľom uľahčiť potenciálne budúce rozšírenia a rýchlu adaptáciu pre rôzne scenáre. Základom komunikácie medzi servermi sú štandardné webové protokoly HTTPS a WebSocket (WS), ktoré sú implementované v rámci rozhrania rešpektujúceho postupy a charakteristiky architektonického štýlu REST. Jeden centrálny klientsky server môže komunikovať a spravovať prístup k viacerým experimentovým serverom súčasne, pričom sa zachováva jednotné komunikačné rozhranie.

Klientsky server

Server pre správu prístupu k experimentom je postavený na princípoch cloudovej architektúry a funguje najmä ako centrálny bod pre správu používateľov a prístup k viacerým experimentálnym prostrediam. Úlohou klientskeho serveru je poskytnutie unifikovaného rozhrania pre prístup používateľov k širokej škále experimentálnych zariadení, bez potreby špecializovaného hardvéru.

Diagram architektúry klientskeho serveru ilustruje obr. 2. Hlavným komponentom klientskeho serveru je interaktívna webová aplikácia, ktorá je prístupná pre ľubovoľné zariadenie s webovým prehliadačom a pripojením na Internet. V prostredí webovej aplikácie má používateľ k dispozícii vytvorenie a spustenie experimentu na konkrétnom zariadení, vrátane nastavenia parametrov a vstupov reálneho zariadenia. Výstupné dáta experimentu sú v reálnom čase vizualizované graficky priamo v aplikácii a prístupné aj po skončení experimentu s možnosťou stiahnutia na lokálne úložisko. Zároveň má používateľ počas priebehu experimentu možnosť zobrazenia živého vysielania prebiehajúceho experimentu na konkrétnom zariadení.

Dáta získané z jednotlivých experimentov sú trvalo archivované na klientskom serveri a možno k nim pristupovať v budúcnosti podľa potreby. V prípade obsadenosti konkrétneho experimentu má používateľ možnosť vyhradiť si konkrétny čas pre experimentovanie v rezervačnom systéme, prípadne využiť vykonanie experimentov v dávkovom (*batch*) móde, kedy sú experimenty vykonané v čase, keď sa zariadenie nepoužíva. Dáta takto spustených experimentov sú používateľovi prístupné ihneď po ich vykonaní.



Obr. 2: Diagram architektúry serveru pre správu prístupu k experimentom

Experimentový server

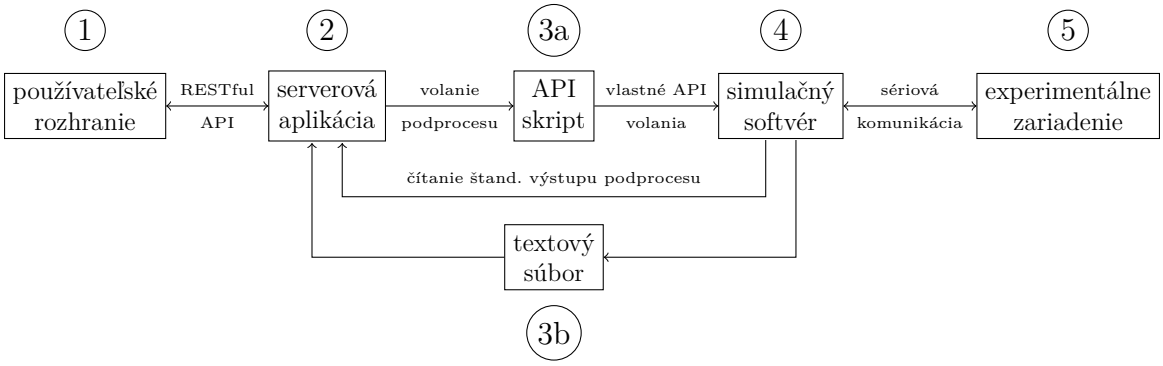
Servery laboratórnych experimentov, resp. experimentové servery, sú navrhnuté a implementované na základe kombinácie architektúry klient-server a *service-oriented* architektúry. Ich úlohou je zabezpečiť efektívne spracovanie používateľských vstupov experimentov a komunikáciu s reálnymi laboratórnymi zariadeniami. Jeden server umožňuje pripojenie viacerých reálnych zariadení aj implementáciu viacerých simulačných prostredí. K serveru môže byť pripojená webkamera, ktorá umožňuje v reálnom čase sledovať živý prenos obrazu reálneho zariadenia v laboratóriu. Súčasná podoba architektúry tiež zabezpečuje integritu komunikácie aj v prípade rozšírenia o nové simulačné prostredia alebo experimentálne zariadenia.

Jednou z výhod implementácie laboratória IOLab spočíva v tom, že poskytuje možnosť nezávislého prevádzkovania serveru pre laboratórne experimenty. Tento systém umožňuje maximalizovať flexibilitu, čo je obzvlášť užitočné pre lokálne experimentovanie

alebo ďalší vývoj laboratória. Experimentový server môže v tomto režime fungovať autonómne a nie je nutné prepojenie s klientskym serverom. Obmedzenie experimentového serveru spočíva v sekvenčnom spracovaní požiadaviek na experimenty, a teda nie je možné vykonávať experimenty paralelne na viacerých zariadeniach.

2.1.1 Komponenty experimentového serveru

Komplexnú architektúru serveru pre laboratórne experimenty vo všeobecnosti tvorí šesť komponentov. Diagram všeobecnej architektúry a prepojenia jednotlivých komponentov je znázornený na obr. 3.



Obr. 3: Všeobecný diagram architektúry serveru laboratórnych experimentov

1. Používateľské rozhranie
Používateľským rozhraním sa rozumie grafické prostredie, ktoré umožňuje interakciu medzi používateľom a prostredím vzdialeného laboratória. Toto grafické prostredie môže byť realizované v rôznych podobách, či už prostredníctvom webovej, natívnej alebo mobilnej aplikácie. V kontexte komplexnej architektúry IOLab je možné používateľským rozhraním rozumieť webovú aplikáciu, ktorú poskytuje server pre správu prístupu k experimentom. Komunikačné rozhranie pre prijímanie a odosielanie dát v rámci grafického prostredia je implementované prostredníctvom štandardných webových technológií HTTPS a WS, ktoré musia rešpektovať postupy a charakteristiky štýlu REST. Pre korektné fungovanie používateľského rozhrania je nevyhnutné, aby mala aplikácia grafického prostredia prístup na internet a aby bolo jej aplikačné rozhranie implementované podľa špecifikácií.

Podobne, ako pri riešení klientskeho serveru, disponuje aj experimentový server v základe jednoduchým webovým rozhraním, ktoré primárne slúži administrátorovi laboratória a poskytuje mu funkcie pre správu existujúcich a implementáciu nových reálnych experimentálnych zariadení. Okrem toho je možné v tomto rozhraní testovať rôzne návrhy potenciálnych experimentov, napr. nastavovať vstupné parametre a spúšťať

experimenty, sledovať v reálnom čase grafickú vizualizáciu údajov experimentu aj priebeh experimentu na konkrétnom zariadení prostredníctvom živého vysielania z webkamery pripojenej k serveru. Vďaka tomu je možné využívať experimentový server ako nezávislé prostredie pre lokálne experimentovanie.

2. Serverová aplikácia
Jedným z kľúčových komponentov experimentového serveru a celého konceptu unifikovanej architektúry je serverová aplikácia, postavená na frameworku Laravel. Jej úlohou je spracovanie požiadaviek používateľov, ktoré sú zadávané v používateľskom rozhraní a následné spustenie experimentov.

Každé experimentálne zariadenie je v aplikácii definované konfiguračným súborom, v ktorom je definovaný počet a formát vstupov a výstupov zariadenia a podporované príkazy alebo inštrukcie sériového komunikačného protokolu experimentálneho zariadenia. Každý experiment má vytvorený vlastný logovací súbor s jedinečným identifikátorom, do ktorého sa v reálnom čase zapisujú výstupné údaje experimentu.

Experimenty sú spúšťané na serveri ako asynchrónne podprocesy operačného systému, bežiace mimo hlavného procesu serverovej aplikácie. Spustenie, priebeh, ukončenie procesu experimentu a zachytenie chybového stavu procesu experimentu zabezpečuje modul Laravel Worker prostredníctvom vstavenej triedy Process a systémového programu `supervisor`. Ak je na server zaslaných viacero požiadaviek na realizáciu experimentov súčasne, logika aplikácie zabezpečí organizované zaradenie každého experimentu do fronty (queue). Táto fronta zabezpečuje sekvenčné vykonanie jednotlivých experimentov v chronologickom poradí.

3a. API skript
Integráciu serverovej aplikácie a simulačného softvéru zabezpečuje vrstva architektúry, ktorá pozostáva z API skriptu a textového súboru. API skript je implementovaný pomocou jazyka Python a zodpovedá za prenos spracovaných vstupných parametrov zo serverovej aplikácie do simulačného prostredia a ovládanie spustenia resp. ukončenia simulácie s reálnym zariadením. Serverová aplikácia v súčasnosti podporuje definovanie troch API skriptov:

1. `start.py` – nastaví vstupné parametre experimentu, čas a periódu vzorkovania simulácie v simulačnom prostredí a spustí simuláciu,
2. `change.py` – umožňuje zmeniť vybrané parametre experimentu počas jeho priebehu,
3. `stop.py` – okamžite ukončí aktuálne bežiacu simuláciu a vráti zariadenie do základného stavu.

Každý definovaný API skript je spustiteľný a obsahuje algoritmus spracovania vstupných argumentov (*argument parser*). Parser argumentov zabezpečuje spracovanie vstupných parametrov experimentu zo serverovej aplikácie. Jednotlivé API skripty sú následne spustené prostredníctvom triedy Laravel Process ako asynchrónne podprocesy systému.

Pri implementácii komunikácie s prostrediami MATLAB a Simulink sa využíva balík *MATLAB Engine API for Python*, ktorý sa štandardne nachádza v rámci základnej inštalácie softvéru. Tento balík umožňuje API skriptom architektúry IOLab volanie príkazov a interakciu s prostredím MATLAB a Simulink z Python skriptov pomocou preddefinovaných tried, metód a funkcií balíka.

Aplikácia pysimCoder v súčasnosti neposkytuje štandardizované alebo unifikované aplikačné rozhranie pre externú interakciu. Pre dosiahnutie želanej funkcionality je potrebné využiť otvorenosť aplikácie a analyzovať proces spustenia simulácie. Pri simulácii blokovej schémy programom pysimCoder je vytvorený súbor `tmp.py` ktorý kompiluje blokovú schému do spustiteľného binárneho súboru. Pri štandardnom spustení simulácie v grafickom režime sú tieto súbory po skončení simulácie automaticky vymazané. V prípade, že v grafickom režime zvolíme možnosť generovania kódu, ostanú tieto súbory v adresári s blokovou schémou.

Pri implementácii vzdialeného experimentu prostredníctvom aplikácie pysimCoder sa využije logika API skriptov architektúry IOLab, ktoré majú definovaný algoritmus spracovania vstupných argumentov. Do API skriptu skopíruje kód funkcie `tmp.py`, v ktorom sa explicitné hodnoty vstupných parametrov blokov, času simulácie a periódy vzorkovania nahradia premennými. Na koniec API skriptu sa pridá volanie kompilácie vygenerovaného kódu a spustenia súboru skompilovanej simulácie.

3b. Zápis výstupu

Druhou časťou architektúry, ktorá integruje simulačný softvér a serverovú aplikáciu, je reprezentovaná textovým súborom a štandardným výstupom spusteného procesu. Táto časť vrstvy zabezpečuje prenos aktuálnych výsledkov experimentu v reálnom čase zo zariadenia a simulačného prostredia do serverovej aplikácie. Serverová aplikácia spracuje výstupné údaje do administrátorom definovaného formátu pre korektnú vizualizáciu údajov v klientskom grafickom rozhraní. Textový súbor obsahuje po skončení experimentu kompletné výstupné dáta experimentu, vrátane času a periódy vzorkovania.

4. Simulačný softvér

Modulárny koncept architektúry IOLab umožňuje podporu viacerých simulačných prostredí. Pri vzdialenom experimentovaní zabezpečuje simulačný softvér vo všeobecnosti tri funkcionality:

1. slúži ako prostredie pre navrhovanie a modelovanie experimentov, napr. v podobe blokových schém a ich kompiláciu do spustiteľných simulácií,

2. zabezpečuje zápis spracovaných vstupných parametrov experimentu do simulačného modelu, spustenie simulácie a výpočet definovaných algoritmov počas experimentu,

3. zabezpečuje komunikáciu s reálnym experimentálnym zariadením, zápis výsledkov výpočtov algoritmov experimentu a čítanie výstupov zariadenia počas prebiehajúceho experimentu.

V súčasnosti pracujeme s experimentálnymi zariadeniami, ktoré disponujú sériovým rozhraním USB a komunikačným protokolom, ktorý je založený na kódovaní ASCII. Tento protokol umožňuje zariadeniu prijímať príkazy a odosielať údaje v štandardnom textovom formáte. Z hľadiska experimentu môžu mať takéto zariadenia aj viacero vstupov a výstupov. Výber simulačného prostredia je preto podmienený jeho podporou funkcií pre komunikáciu prostredníctvom sériového rozhrania.

Implementáciu vlastných blokov v rámci prostredia Simulink je možné riešiť viacerými spôsobmi [43]. Pre systém s viacerými vstupmi a výstupmi, ktorý je reprezentovaný fyzickým zariadením komunikujúcim cez sériové rozhranie, je možné takýto blok štandardne definovať pomocou tzv. *Level-2 MATLAB S-Function*, ktorá je schopná prijať a spracovať akýkoľvek signál vytvorený v Simulinku [44].

Na rozdiel od Simulinku je implementácia vlastného bloku pre aplikáciu pysimCoder možná pomocou dvoch súborov v jazykoch Python a C. Python súbor zabezpečuje konfiguráciu vizuálnej podoby a definuje vstupno-výstupné parametre bloku nového bloku pre grafický editor a súbor jazyka C zaisťuje implementáciu vnútornej logiky a funkcionality bloku.

Druhým hlavným rozdielom v simulácii blokovej schémy je, že program pysimCoder vždy pred simuláciou skompiluje všetky súbory definujúce blokovú schému do jedného spustiteľného binárneho súboru a následne spustí simuláciu. Totožný postup je pri generovaní spustiteľného kódu, ktorý, na rozdiel od simulácie v programe, nie je po kompilácii spustený a vymazaný.

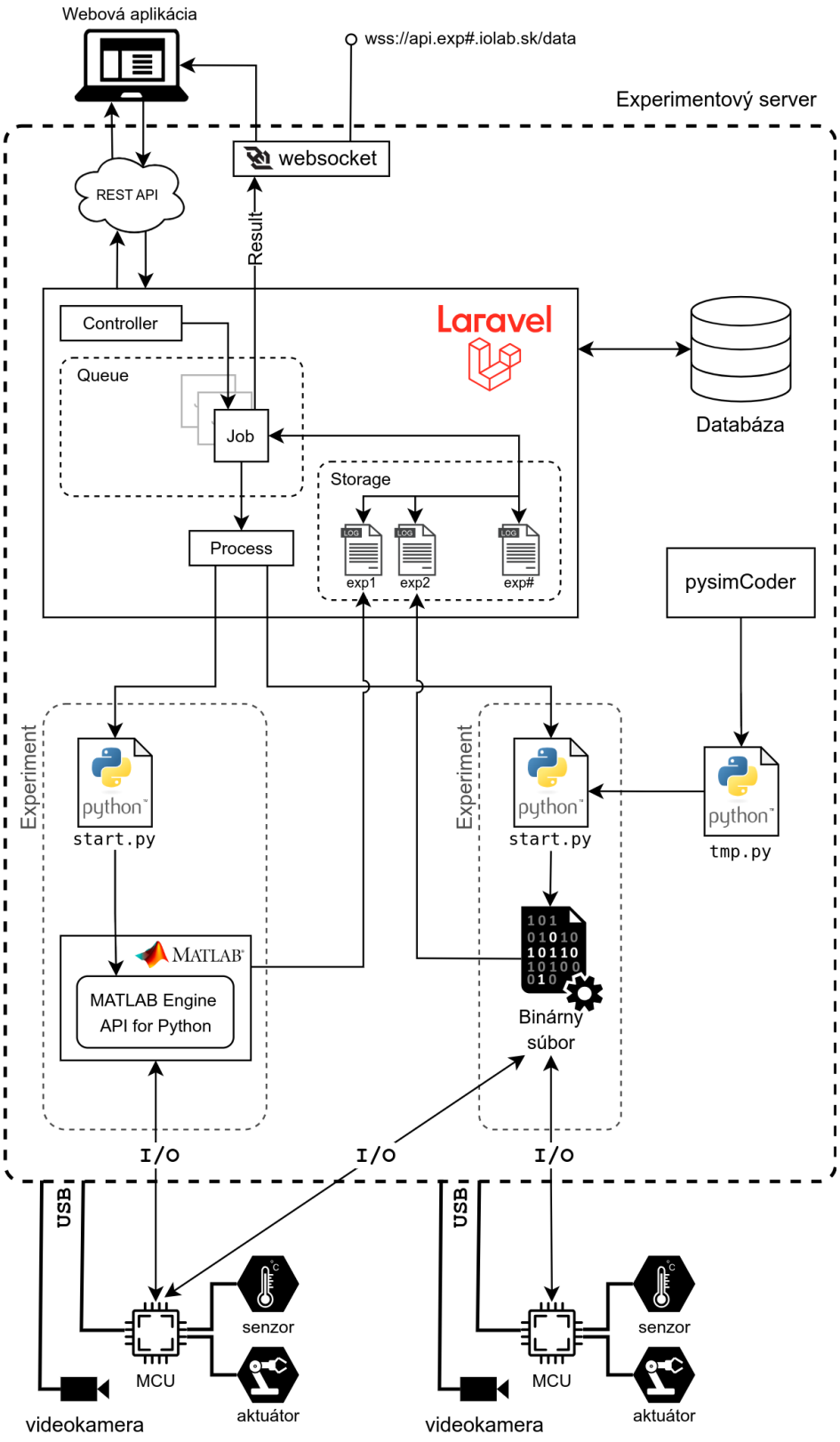
5. Experimentálne zariadenie

V kontexte vzdialeného laboratória pod pojmom experimentálne zariadenie rozumieme laboratórny aparát, pozostávajúci z mikrokontroléra, senzorov a aktuátorov. Zariadenie musí disponovať komunikačným rozhraním a známym komunikačným protokolom, prostredníctvom ktorých je možná plnohodnotná interakcia s perifériami zariadenia.

Architektúra laboratória IOLab podporuje jednosmernú aj obojsmernú komunikáciu, takže je možné simultánne získavať údaje zo senzorov a konfigurovať parametre aktuátorov. V našom riešení sa zameriavame na zariadenia, ktoré disponujú sériovým komunikačným rozhraním USB.

Jednoduchosť komunikácie so zariadením prostredníctvom USB rozhrania môžeme demonštrovať na operačnom systéme na báze Linux. Každému zariadeniu pripojenému cez USB je v Linuxovom systéme priradený jednoznačný identifikátor – súbor v adresári `/dev`. Tento súbor slúži ako rozhranie pre sériový port, ktorý umožňuje vzájomnú interakciu so zariadením. Komunikácia so zariadením zahŕňa operácie čítania a zápisu dát v kontexte tohto špecifického súboru.

Navrhnutá modulárna architektúra je postavená na open-source technológiách, ale umožňuje integráciu s obidvomi kategóriami simulačných prostredí – podporované sú otvorené aj proprietárne softvérové riešenia. Modularita riešenia je založená na koncepte, že každé zariadenie a každý simulačný softvér je možné chápať ako samostatné moduly, ktoré je možné vzájomne kombinovať pri vytváraní a spúšťaní experimentov. Diagram implementácie architektúry serveru pre laboratórne experimenty ilustruje obr. 4.



Obr. 4: Diagram implementácie serveru pre laboratórne experimenty

3 Proces integrácie experimentu do vzdialeného laboratória

Architektúra vzdialeného laboratória IOLab je modulárna, čo umožňuje flexibilnú integráciu rôznych hardvérových a softvérových zariadení do systému. Táto modularita umožňuje používateľom vykonávať vzdialené experimenty v rôznych simulačných prostrediach za predpokladu správneho postupu integrácie. Pre úspešné začlenenie nového experimentu do online laboratória je potrebné splniť konkrétne podmienky, ktoré zabezpečia spoľahlivú komunikáciu medzi komponentmi hybridnej architektúry, čím sa zaistí plynulá a efektívna interakcia používateľa s laboratórnym zariadením.

Pre zabezpečenie plnohodnotnej interakcie so vzdialeným experimentom je potrebné mať k dispozícii:

1. Experimentálny hardvér, ktorý má zdokumentované komunikačné rozhranie a komunikačný protokol alebo poskytuje API alebo SDK na účely integrácie.
2. Simulačné prostredie, ktoré:
   (a) podporuje priamo komunikačné rozhranie experimentálneho hardvéru,
   (b) umožňuje definovanie vlastného komunikačného rozhrania prostredníctvom vstavaných tried a metód,
   (c) má otvorený zdrojový kód umožňujúci úpravy v rámci licenčných podmienok za účelom rozšírenia alebo modifikácie funkcionality,
   (d) poskytuje mechanizmus pre definovanie a spúšťanie simulačných scenárov alebo skriptov nezávisle od grafického používateľského rozhrania.
3. Spoľahlivú integráciu experimentálneho hardvéru so simulačným prostredím, ktorá zabezpečuje výmenu dát a riadenie procesov.
4. Spoľahlivú integráciu simulačného prostredia s webovou aplikáciou, ktorá umožní vzdialenú interakciu používateľa s experimentom.
5. Definované a implementované algoritmy, ktoré určujú logiku a postupnosť operácií v rámci experimentu.

3.1 Experimentálny hardvér

Prvou a kľúčovou podmienkou pre vzdialené experimentovanie v systéme IOLab je dôkladná znalosť špecifikácie experimentálneho hardvéru, vrátane jeho komunikačného rozhrania a protokolu. Architektúra IOLab umožňuje integráciu zariadení s rôznymi

typmi komunikačných rozhraní — pevnými (káblovými) aj bezdrôtovými. Káblové rozhrania môžu byť napríklad USB, RS-232, RS-485, Ethernet alebo CAN, zatiaľ čo bezdrôtové možnosti zahŕňajú Wi-Fi, Bluetooth, ZigBee, LoRa či NFC. Výber technológie závisí od požiadaviek na prenosovú rýchlosť, dosah a energetickú efektívnosť zariadenia.

Káblové a bezdrôtové komunikačné rozhrania majú špecifické výhody a nevýhody, ktoré treba zvážiť pri návrhu experimentálneho zariadenia, aby bol zabezpečený spoľahlivý prenos dát. Prehľadné porovnanie vlastností týchto rozhraní ilustruje tab. 1.

Káblové rozhrania poskytujú vysokú stabilitu a sú menej citlivé na elektromagnetické rušenie (napr. rádiové alebo mikrovlnné signály) [45]. Okrem prenosu dát môžu cez rovnaký kábel zabezpečiť aj napájanie zariadení, ako napríklad USB pre nízkonapäťové aplikácie alebo PoE pre zariadenia s vyššou spotrebou [46]. Nevýhodou je obmedzená mobilita a maximálna dĺžka káblov, napríklad USB kábel bez zosilňovača môže mať maximálne okolo 5 metrov [47].

Bezdrôtové technológie prinášajú väčšiu flexibilitu a jednoduchšiu inštaláciu, keďže eliminujú potrebu fyzickej kabeláže. Umožňujú tiež pripojenie viacerých zariadení v rámci jednej siete a komunikáciu na väčšie vzdialenosti (v závislosti od technológie). Nevýhodou je vyššia spotreba energie, citlivosť na elektromagnetické rušenie a bezpečnostné riziká pri prenose dát vo vzduchu. Preto je nutné implementovať vhodné kryptografické a šifrovacie mechanizmy na ochranu integrity a dôvernosti prenášaných informácií [48].

Pre implementáciu komunikácie so zariadením je nevyhnutné poznať jeho komunikačný protokol, ktorý definuje pravidlá a inštrukcie pre interakciu so senzormi a aktuátormi zariadenia. Ak protokol nie je známy, môže sa využiť API alebo SDK od výrobcu, prípadne metódy reverzného inžinierstva na analýzu dátových tokov a objasnenie komunikácie.

V rámci našej architektúry sme integrovali tri zariadenia so sériovým USB 2.0 rozhraním, ktoré komunikujú pomocou vlastného ASCII textového protokolu. Dve zariadenia majú uzavretý protokol, ale sú dostupné známe príkazy pre prácu s perifériami. Komunikácia tretieho zariadenia bola navrhnutá a implementovaná priamo v rámci vývoja systému ABMS.

3.2 Simulačné prostredie

Simulačné prostredie je softvér, kde používateľ definuje algoritmus a parametre experimentu. Existujú dva základné prístupy k realizácii experimentov na reálnych zariadeniach:

Tabuľka 1: Porovnanie káblových a bezdrôtových rozhraní

Kategória	**Káblové rozhrania**	**Bezdrôtové rozhrania**
Spoľahlivosť	Vysoká, odolné voči elektromagnetickému rušeniu (EMI, RFI).	Nižšia, citlivé na rušenie inými bezdrôtovými signálmi.
Mobilita	Obmedzená fyzickou dĺžkou kábla a nutnosťou pevného spojenia.	Vysoká, umožňuje mobilitu a flexibilnú inštaláciu zariadení.
Energetická náročnosť	Nižšia; možné napájanie cez kábel (USB, PoE).	Vyššia; vyžaduje batérie alebo externé napájanie.
Maximálna vzdialenosť	Limitovaná (napr. USB $\sim 5m$, RS-232 $\sim 15m$ bez zosilnenia).	Závislá od technológie (Wi-Fi $\sim 30m$, LoRa až niekoľko km).
Prenosová rýchlosť	Vysoká (USB 3.x až 5–10 Gbps, Ethernet až 100 Gbps).	Nižšia v porovnaní s káblovými; výnimkou je 5G alebo Wi-Fi 6.
Bezpečnosť	Vysoká, ťažšie odpočúvať bez fyzického prístupu.	Vyžaduje šifrovanie a zabezpečenie proti neoprávnenému prístupu.
Inštalácia	Náročnejšia pri dlhých vedeniach; pevná infraštruktúra.	Jednoduchšia, nevyžaduje káble ani fyzickú infraštruktúru.

1. Experiment je definovaný a spustený priamo v simulačnom prostredí. Výpočty sa vykonávajú v softvéri, pričom zariadenie slúži hlavne na čítanie senzorických údajov a ovládanie aktuátorov cez komunikáciu.

2. Experiment je definovaný v simulačnom prostredí, ale celý program s riadiacou logikou sa prenesie a spustí priamo na zariadení. Výpočty prebiehajú na hardvéri zariadenia a výsledky môžu byť spätne zaslané do simulačného prostredia na ďalšie spracovanie a vizualizáciu.

Pre zabezpečenie plnohodnotnej simulácie je potrebné implementovať komunikáciu simulačného prostredia s experimentálnym hardvérom aj s architektúrou vzdialeného laboratória. V tomto prípade rozlišujeme štyri prípady:

- Simulačné prostredie s natívnou podporou hardvéru: Výrobca poskytuje softvérový balík (napr. knižnice pre MATLAB alebo LabVIEW) priamo kompatibilný so zariadením. Výhodou je jednoduchá integrácia, nevýhodou uzavretosť a nemožnosť úprav zdrojového kódu.

- Simulačné prostredie s podporou používateľom definovaného rozhrania: Prostredníctvom API alebo SDK je možné vytvárať vlastné komunikačné moduly, ak sú známe a zdokumentované protokoly zariadení. Umožňuje prispôsobenie funkcionality podľa špecifických potrieb experimentu.

- Simulačné prostredie s otvoreným zdrojovým kódom: Umožňuje priamo meniť zdrojový kód simulačného softvéru a implementovať vlastné funkcie pre komunikáciu s hardvérom. Tento prístup je flexibilný, no časovo náročnejší a vyžaduje hlbšiu znalosť kódu. Príkladom je integrácia programu pysimCoder do vzdialeného laboratória.

- Spustenie simulácií nezávisle od grafického používateľského rozhrania (GUI): Kľúčové pre vzdialenú automatizáciu experimentov je, aby bolo možné simulácie definovať a spúšťať cez API, CLI alebo sieťové protokoly bez potreby GUI. Takýto prístup umožňuje webovým aplikáciám efektívne riadiť experimenty v klient-server architektúre.

3.3 Integrácia hardvéru a simulačného prostredia

Integrácia experimentálneho hardvéru so simulačným prostredím spočíva v implementácii komunikačného rozhrania, ktoré umožňuje interakciu používateľa so zariadením. Väčšina komerčných produktov prichádza so softvérovým balíkom pripraveným na okamžité použitie, ktorý je však často proprietárny, neumožňuje modifikácie a podporuje iba vybrané platformy a simulačné prostredia. Ak balík nevyhovuje požiadavkám na vzdialené experimentovanie a komunikačný protokol hardvéru je neznámy alebo neverejný, integrácia je náročná na čas a technickú realizáciu.

Pri rôznych typoch softvéru sa integrácia realizuje pomocou spustiteľných skriptov využívajúcich knižnice a metódy simulačného softvéru alebo programovacieho jazyka, v ktorom je rozhranie vytvorené. Komplexnejšie riešenia umožňujú tvorbu knižníc či balíkov, ktoré obsahujú pokročilú logiku pre čítanie, zápis a výpočty, často zostavené do samostatných programov. V rámci vzdialeného laboratória je úlohou administrátora zabezpečiť prostredníctvom týchto mechanizmov plnohodnotnú obojsmernú komunikáciu s hardvérom vrátane možnosti diaľkového nastavenia vstupných a výstupných parametrov simulácie.

Na základe analýzy bola v rámci navrhovaného riešenia realizovaná integrácia zariadení v dvoch simulačných prostrediach – MATLAB a pysimCoder. Tieto predstavujú zástupcov dvoch odlišných kategórií softvéru. Pre všetky dostupné experimentálne zariadenia, ako aj pre vyvíjaný systém ABMS, boli vytvorené komplexné knižnice určené pre uvedené prostredia. Uvedené knižnice zabezpečujú plnohodnotnú obojsmernú

komunikáciu so zariadeniami vrátane čítania a zápisu údajov z periférií a konfigurácie podporovaných parametrov zariadení. Implementácia knižníc bola realizovaná s dôrazom na požiadavky experimentovania v reálnom čase.

3.4 Integrácia simulačného prostredia a webovej aplikácie

Pri návrhu všeobecného riešenia integrácie simulačných prostredí do serverovej aplikácie je kľúčovým krokom vytvorenie unifikovaného rozhrania medzi experimentovým serverom a samotným simulačným softvérom. Toto rozhranie tvorí spojovací bod medzi jednoduchým používateľským prostredím webovej aplikácie a komplexným prostredím, v ktorom prebiehajú simulácie alebo experimenty. Aby bolo možné zabezpečiť efektívnu interakciu, je nevyhnutné, aby simulačné prostredie podporovalo externú definíciu parametrov a spúšťanie experimentov bez závislosti na grafickom používateľskom rozhraní (GUI). Túto funkcionalitu môžu poskytovať štandardizované API, softvérové knižnice SDK alebo iné programovateľné rozhrania.

Jednou z možností je využitie samostatných výpočtových modulov simulačného prostredia, ktoré sa dajú zoskupiť do skriptov a spúšťať v dávkovom režime s možnosťou parametrizácie vstupov. Tento prístup umožňuje flexibilné definovanie vstupných hodnôt a dynamickú zmenu parametrov experimentov. Alternatívne, ak to simulačný softvér umožňuje, môže sa výpočtové jadro spúšťať v príkazovom režime, čo umožňuje programátorovi vykonávať experimentálne skripty a odovzdávať im potrebné argumenty cez komunikačné rozhranie.

V prípade softvérov s otvoreným zdrojovým kódom je ďalšou možnosťou upraviť ich interné mechanizmy tak, aby sa umožnilo spúšťanie experimentov nezávisle od GUI. Tento prístup si vyžaduje hlbokú znalosť zdrojového kódu a návrh vhodného mechanizmu pre komunikáciu s webovou aplikáciou. Často ide o kombináciu priamych volaní interných tried a metód s pomocou riadiacich skriptov, čím sa vytvorí jednotné rozhranie pre serverovú aplikáciu.

V praktickej implementácii je možné tieto prístupy kombinovať a vytvoriť unifikovaný algoritmus, ktorý zabezpečí spúšťanie simulácií v prostredí experimentového serveru. Takéto riešenie umožňuje webovej aplikácii efektívne spravovať experimenty, sledovať priebeh procesov, definovať vstupné argumenty a kontrolovať prístup k systémovým zdrojom.

Konkrétny návrh riešenia v tomto prípade zahŕňal integráciu dvoch odlišných simulačných prostredí, MATLAB a pysimCoder, do serverovej aplikácie postavenej na frameworku Laravel a nasadenej na serveri Nginx v prostredí Ubuntu Linux. V

prípade MATLABu sa využíva *Shared Engine*, ktorý umožňuje spúšťanie simulácií cez *MATLAB Engine API for Python*. Pre pysimCoder bol navrhnutý mechanizmus, ktorý priamo využíva časť jeho zdrojového kódu v Pythone na zostavenie blokových schém a spúšťanie experimentov. Obe riešenia boli zjednotené do Python skriptov, ktoré umožňujú efektívne spúšťanie a správu experimentov prostredníctvom webovej aplikácie, vrátane kontroly času spustenia, plánovania a správy fronty požiadaviek.

3.5 Definícia návrhu experimentu

Návrh experimentu je základným nástrojom na systematické testovanie hypotéz v reálnom aj simulačnom prostredí. Definuje vstupy, výstupy, logiku a podmienky experimentu vrátane komunikácie so zariadeniami a operácií s nimi. V reálnom prostredí zahŕňa aj nastavenie hardvéru a zber dát.

Dôležitým cieľom návrhu je zabezpečiť prepojenie a porovnateľnosť medzi simuláciou a reálnym experimentom, čo vyžaduje kalibráciu modelov a zohľadnenie reálnych odchýlok. Kvalitná dokumentácia experimentu (cieľ, postup, merané veličiny, analýza) zabezpečuje jeho reprodukovateľnosť a transparentnosť.

Celkovo návrh experimentu formálne popisuje všetky potrebné prvky na jeho opakovateľné a prenosné vykonanie na rôznych platformách a hardvéroch, čím podporuje modularitu a zdieľanie medzi laboratóriami.

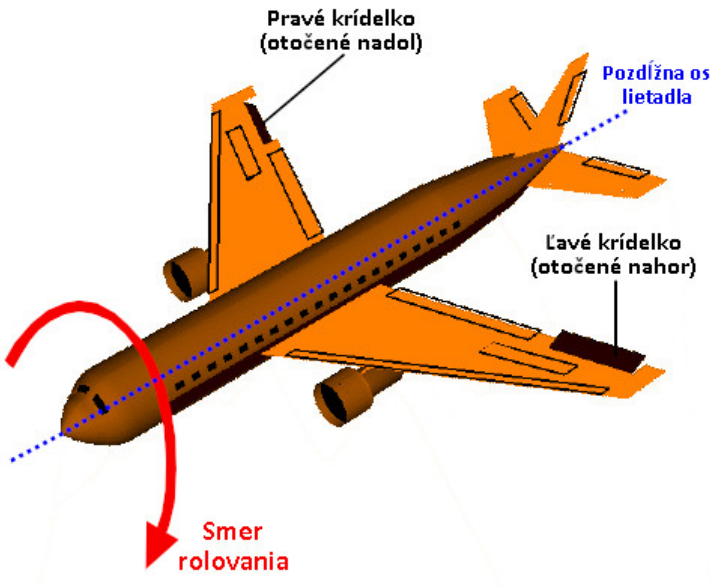
4 Mechatronický systém Guľa na ramene

Hlavným cieľom práce je navrhnúť a otestovať flexibilnú architektúru pre vzdialené riadenie experimentov s využitím otvorených technológií. Ako ukážkový príklad bude implementácia realizovaná na mechatronickom systéme Ball on Beam (guľa na ramene). Výsledkom je funkčný prototyp, ktorý môže slúžiť ako základ pre ďalší vývoj a vzdelávanie v oblasti vzdialeného riadenia experimentov.

4.1 Charakteristika systému

Systém Ball on Beam je jednoduchý mechatronický model, v ktorom pohyb gule po lineárnej dráhe vzniká nakláňaním nosníka pomocou aktuátora (napr. servomotora, elektromotora alebo hydraulického/pneumatického mechanizmu).

Praktickou analógiou je valivý pohyb lietadla pri rolovaní, keď sa trup nakláňa okolo pozdĺžnej osi. Tento pohyb zabezpečujú balančné krídelká (ailerons) umiestnené na zadnej časti krídel (obr. 5), ktoré pracujú v protismerných polohách a umožňujú lietadlu ovládať priečny náklon [49].



Obr. 5: Funkcia krídelok na lietadle

Systém gule na ramene slúži ako didaktický nástroj v oblasti výučby teórie riadenia. Jeho jednoduchá fyzikálna realizácia umožňuje pochopenie základných princípov modelovania a simulácie dynamických systémov. Zároveň poskytuje možnosti pre vedecký výskum v oblasti návrhu a implementácie moderných riadiacich algoritmov a

overenie postupov modelovania a riadenia nestabilných nelineárnych dynamických SISO (Single-Input Single-Output) systémov.

Úlohou riadiaceho algoritmu je zabezpečiť stabilizáciu gule v žiadanej polohe na ramene. Pri náklone ramena sa dôsledkom pôsobenia tiažovej sily Zeme guľa po ramene voľne pohybuje so zrýchlením, ktoré závisí od veľkosti uhla náklonu ramena. Z pohľadu riadenia ide o typický príklad dynamického systému, ktorý je nestabilný v otvorenej slučke. Pri konštantnom vstupe (uhol náklonu ramena) systém produkuje neobmedzený rastúci výstup (polohu gule). Na stabilizáciu systému je nevyhnutné zaviesť riadenie so spätnou väzbou, ktoré kontinuálne koriguje polohu gule. Účinné riadenie systému vyžaduje presné určenie aktuálnej polohy gule a následnú úpravu uhla náklonu ramena pomocou aktuátora, pričom veľkosť a smer zmeny náklonu musia byť úmerné odchýlke gule od požadovanej polohy.

4.2 Konfigurácie systému

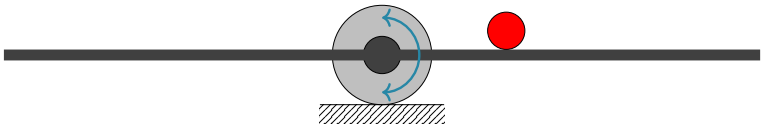
Konštrukcia systému gule na ramene má niekoľko možností realizácie. Jednotlivé implementácie sa líšia hlavne miestom, kde sa nachádza otočný kĺb, v ktorom sa rameno nakláňa, a miestom, kde na rameno pôsobí aktuátor, ktorý ho vychyľuje z vodorovnej polohy. Existujú dva hlavné spôsoby upevnenia ramena systému, ktoré sa ďalej modifikujú a upravujú podľa potrieb konkrétnej implementácie.

Analýza konštrukčných riešení systému gule na ramene ukazuje širokú škálu možných variantov systému. Kľúčové parametre ovplyvňujúce správanie systému sú poloha otočného kĺbu, umožňujúca náklon ramena, miesto pôsobenia aktuátora a konštrukcia mechanizmu aktuátora. Z hľadiska uchytenia ramena možno identifikovať dva hlavné koncepty, ktoré sa ďalej modifikujú v závislosti od konkrétnych prevádzkových podmienok a funkčných požiadaviek.

Aktuátor pôsobiaci v mieste výkyvu

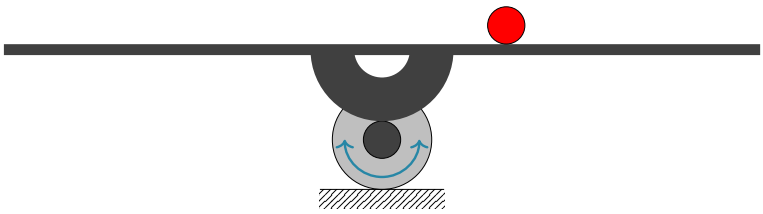
Prvá možnosť konfigurácie systému je charakteristická umiestnením výkyvného bodu ramena do jeho stredovej osi. V tomto bode je kinematicky spojený výstupný hriadeľ aktuátora, ktorý slúži ako primárny zdroj náklonu ramena (obr. 6). Z hľadiska mechanickej konštrukcie predstavuje tento variant najjednoduchšie riešenie, nakoľko nevyžaduje zložité komponenty. Presnosť polohovania ramena je v tomto prípade limitovaná rozlíšením použitého aktuátora, konkrétne minimálnym uhlom natočenia jeho výstupného hriadeľa. Tento typ konfigurácie bol použitý a overený aj v praxi, čo dokumentujú napr. publikácie autorov Wang [50], Lieberman [51] alebo Hirsch [52].

Základná konštrukcia systému, charakterizovaná symetrickým usporiadaním ramena vzhľadom k otočnému bodu, môže byť modifikovaná zavedením prevodového mechanizmu



Obr. 6: Konfigurácia č. 1: aktuátor pôsobiaci v osi výkyvu ramena

medzi ramenom a aktuátorom. Typickým príkladom je kinematické spojenie realizované pomocou valivého kontaktu medzi polkruhovým profilom na ramene a menším kolesom na hriadeli aktuátora (obr. 7). Tento mechanizmus spôsobí opačný smer otáčania ramena vzhľadom k pohybu aktuátora. Avšak, vzhľadom na možnosť vzniku nežiaduúeho preklzovania je tento typ prevodu často nahradzovaný ozubeným prevodom, ktorý garantuje presnejší a stabilnejší prenos krútiaceho momentu. Praktické využitie takto modifikovaného systému opisujú napr. publikácie Rosales a kol. [53] alebo Ito [54].



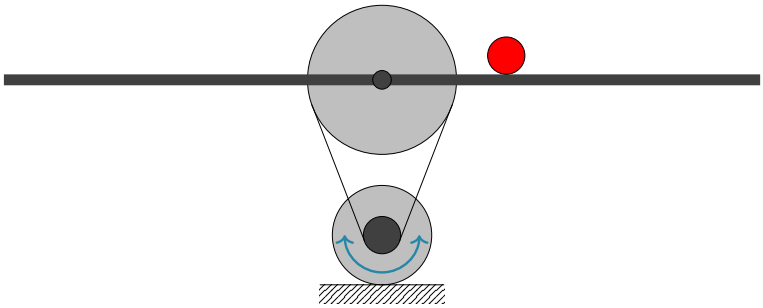
Obr. 7: Konfigurácia č. 1: aktuátor pôsobiaci cez hladký kolesový resp. ozubený prevod

Alternatívnym riešením je využitie remeňového prevodu. Väčšia remenica je v tomto prípade integrálnou súčasťou ramena a je s ním pevne spojená v otočnom bode. Menšia remenica je pripojená k výstupnému hriadeľu aktuátora, pričom obe remenice sú spojené pružným remeňom (obr. 8). Tento typ prevodu zabezpečuje rovnaký smer otáčania ramena a hriadeľa aktuátora. S cieľom zvýšiť spoľahlivosť prevodu je možné nahradiť pružný remeň ozubeným remeňom alebo reťazou, ktoré môžu znížiť pravdepodobnosť výskytu nežiaduceho preklzovania a zabezpečujú presnejší prenos pohybu z aktuátora na rameno.

Implementáciu tohto konceptu môžeme nájsť v komerčných systémoch gule na ramene Amira BW500 [55] alebo TecQuipment CE106 [56], ktorých praktickú aplikáciu opisujú publikácie Chien a kol. [57], Colón a kol. [58] alebo Kagami a kol. [59].

Aktuátor pôsobiaci mimo miesta výkyvu

Druhou možnou konfiguráciou systému je pripevnenie jedného konca ramena pomocou výkyvného bodu k podpornej konštrukcii. Druhý koniec ramena je spojený s aktuátorom prostredníctvom pákového prevodu, pomocou ktorého sa nastavuje sklon ramena (obr. 9). Táto konfigurácia je konštrukčne síce zložitejšia a vyžaduje viacero komponentov, oproti prvej konfigurácii však často nevyžaduje dodatočný redukčný prevod. Rozsah

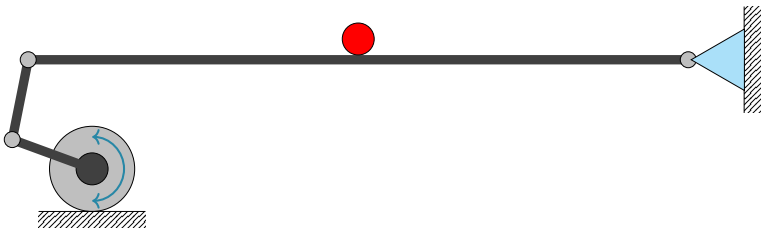


Obr. 8: Konfigurácia č. 1: aktuátor pôsobiaci cez remeňový/reťazový prevod

a presnosť náklonu ramena je možné priamo ovplyvniť zmenou dĺžky komponentov pákového spojenia alebo zmenou miesta pôsobenia pákového prevodu.

Vo všeobecnej praxi sa najčastejšie používa práve táto konfigurácia, najmä v komerčných riešeniach Quanser [60], RYC-BB [61] alebo GBB1004 [62], ktoré boli použité v publikáciách Al-Dujaili a kol. [63] alebo Yongun a kol. [64].

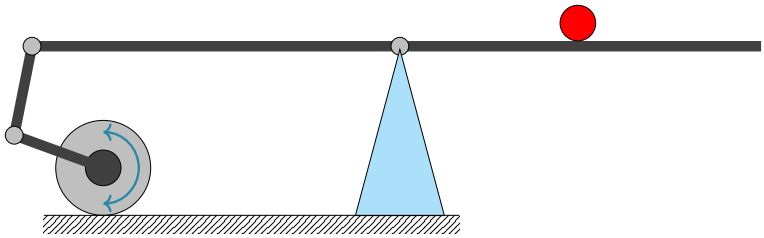
Okrem komerčných riešení sa môžeme v praxi aj v rozličných publikáciách stretnúť s vlastnými konštrukčnými riešeniami systému, ktoré vychádzajú z tohto typu konfigurácie. Vlastné konštrukcie systémov bývajú využívané najmä v akademickom prostredí na overenie návrhov riadiacich štruktúr alebo v domácich podmienkach ako voľnočasové projekty. Medzi takéto systémy patria napr. zariadenia opísané v publikáciách Nowoposki [65], Taifour a kol. [66], Hadipour a kol. [67], alebo Versloot a kol. [68].



Obr. 9: Konfigurácia č. 2: aktuátor pôsobiaci cez pákový prevod

Modifikáciou druhej konfigurácie posunutím výkyvného bodu ramena smerom k stredu ramena môžeme dosiahnuť väčší rozsah uhla náklonu pri zachovaní pôvodnej dĺžky ramena, bez nutnosti modifikácie pákového mechanizmu aktuátora (obr. 10). Táto konštrukcia má výhodu najmä pri aktuátoroch s obmedzeným rozsahom otočenia výstupného hriadeľa alebo pri inštalácii aktuátora a pákového mechanizmu v stiesnených podmienkach. Potenciálna nevýhoda tejto konfigurácie sa môže prejaviť pri použití aktuátorov s nižšou presnosťou, čo môže viesť k problémom pri nastavení uhla sklonu ramena.

Tento typ konštrukcie je implementovaný pri komerčných systémoch Acrome [69] a GBB2004 [62], ktorých praktické využitie opisuje publikácia Bao a kol. [70].



Obr. 10: Konfigurácia č. 3: aktuátor pôsobiaci cez pákový prevod

Z analýzy existujúcich systémov gule na ramene vyplýva, že prvá konfigurácia je jednoduchšia a priamočiarejšia z hľadiska konštrukcie, keďže vyžaduje menej mechanických a pohyblivých častí. Pohyb gule je pri nej symetrický vzhľadom na výkyvný bod na oboch stranách ramena, čo uľahčuje modelovanie systému.

Druhá konfigurácia je mechanicky náročnejšia kvôli zložitejšiemu mechanizmu aktuátora, ale umožňuje presnejšie nastavenie uhla sklonu ramena, čím sa zlepšuje presnosť riadenia polohy gule. Pákový mechanizmus navyše umožňuje použiť kompaktnejšie aktuátory a znižuje opotrebovanie prevodov motora, čím sa predlžuje životnosť systému.

5 Návrh a realizácia experimentu

Pre túto prácu bol navrhnutý a skonštruovaný vlastný systém gule na ramene, založený na druhej konfigurácii s pákovým mechanizmom medzi aktuátorom a ramenom. Architektúra systému využíva modulárny prístup, ktorý umožňuje jednoduché prispôsobenie rôznym aplikáciám. Vďaka tomu podporuje rôzne konfigurácie aktuátorov, snímačov a riadiacich jednotiek, čo zvyšuje jeho flexibilitu a adaptabilitu.

5.1 ABMS: Automatický balančný mechatronický systém

Experimentálna sústava ABMS: Automatický balančný mechatronický systém (obr. 11) predstavuje kompaktný a prenosný laboratórny model gule na ramene, určený na vzdelávacie účely. Umožňuje používateľom testovať a overovať riadiace algoritmy pre nelineárne systémy a je navrhnutá tak, aby bola robustná a vhodná na rôzne experimenty.

Obr. 11: Systém ABMS

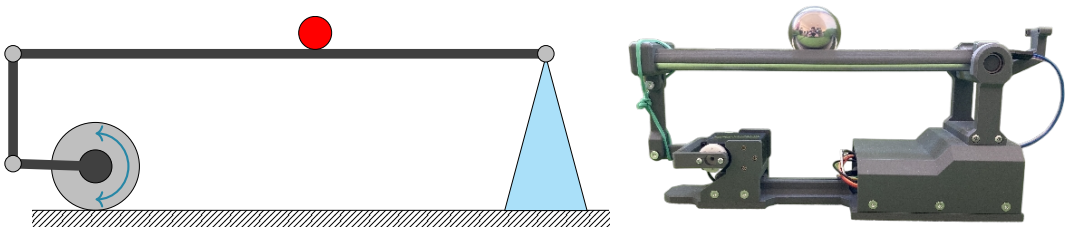
Sústava ABMS vznikla ako výsledok postupného vývoja na Ústave automobilovej mechatroniky, pričom jej aktuálnej verzii predchádzalo viacero prototypov. Prvý model bol zhotovený z lepenkových dielov, následne ho nahradil kovový prototyp zo stavebnice Merkur s malým vozíkom namiesto gule. Najnovšie prototypy vychádzajú z počítačových modelov a sú vyrobené technológiou aditívnej výroby (angl. *additive manufacturing*), s dôrazom na úsporu materiálu a minimalizáciu nevyrobiteľných komponentov. Momentálne sú v prevádzke dva funkčné prototypy, ktoré slúžia na ďalší vývoj hardvéru, softvéru a testovanie celého systému ABMS.

Pri návrhu nového zariadenia sme sa rozhodli definovať nasledujúce kľúčové vlastnosti, ktorými by mal výsledný systém disponovať:

- modularita a variabilita – systém disponuje dvomi redundantnými senzormi polohy, je možné použitie dvoch rôznych riadiacich jednotiek, podporuje proprietárne aj open-source simulačné prostredia.
- konektivita s počítačom prostredníctvom štandardného USB portu,
- kompatibilita s operačným systémom Windows a Linux,
- ľahká prenosnosť a kompaktné rozmery: 250x180x150mm,
- možnosť použitia zariadenia pre experimentovanie v univerzitnom laboratóriu, doma alebo prostredníctvom vzdialených experimentov.

Konštrukcia systému

Konštrukcia sústavy ABMS reprezentuje konfiguráciu systému, pri ktorej aktuátor pôsobí na rameno prostredníctvom pákového prevodu mimo výkyvný bod ramena, ktorý sa nachádza na opačnom konci ramena (Obr. 12).

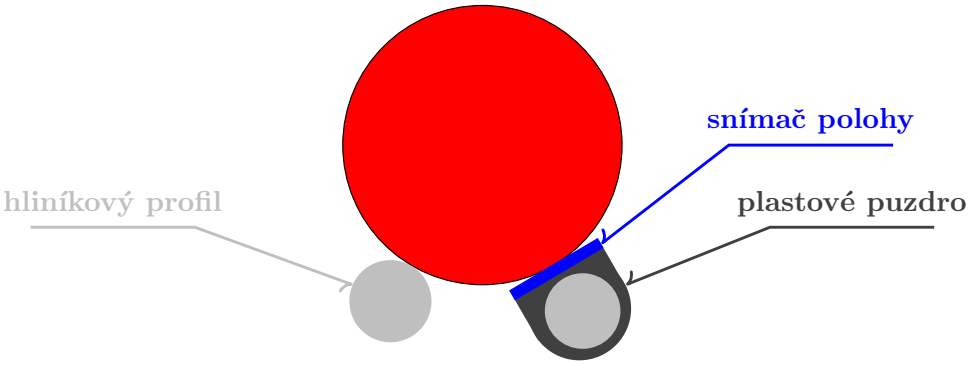


Obr. 12: Schéma konfigurácie a súčasná podoba systému ABMS

Rameno systému ABMS tvorí dvojica dutých hliníkových kruhových profilov s priemerom 10mm a dĺžkou 250mm, pričom elektrická vodivosť materiálu nie je potrebná (na rozdiel od komerčných systémov Quanser alebo RYC-BB). Po ramenách sa pohybuje leštená oceľová guľa. Na jeden profil je nasunuté plastové puzdro s tlakovým snímačom polohy. Schematický prierez ramenom ilustruje obr. 13.

Na jednom konci sú profily ukončené plastovým dielom s otočným kĺbom a držiakom pre laserový senzor a vertikálny výkyv ramena zabezpečujú dve guľôčkové ložiská (10x19x5mm). Druhý koniec je pripojený na výstup aktuátora cez pákový mechanizmus.

Systém ABMS má modulárnu architektúru, ktorá umožňuje prispôsobenie rôznym experimentálnym požiadavkám. Vyvinuté komponenty zvyšujú adaptabilitu a kompatibilitu so širokou škálou aktuátorov, senzorov a riadiacich jednotiek. Zostava je riešená pomocou kovových skrutiek bez lepenia, čo umožňuje jednoduchú výmenu dielov. Komponenty sú vyrábané 3D tlačou metódou FFF z ľahkých a pevných materiálov PLA alebo PETG, čo znižuje hmotnosť a zvyšuje stabilitu. Táto koncepcia umožňuje



Obr. 13: Schematický prierez ramenom systému ABMS

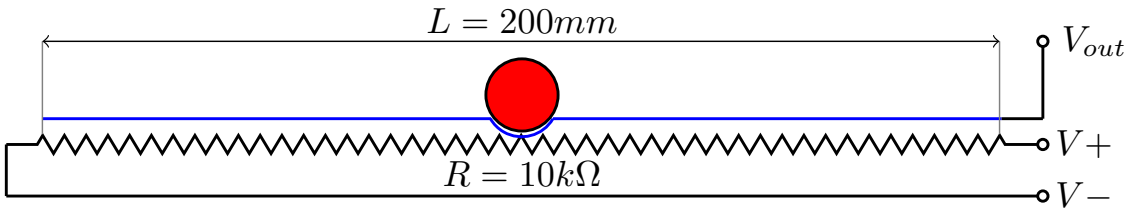
používateľovi jednoducho prispôsobiť, opraviť alebo vyrobiť nové časti bez nutnosti servisu či nákupu drahých náhrad.

Určenie polohy gule

Systém ABMS môže využívať dve nezávislé metódy určenia polohy gule na ramene.

Kontaktné snímanie

Poloha gule sa primárne zisťuje kontaktnou detekciou pomocou tlakom aktivovaného membránového potenciometra. Kovová guľa pri pohybe po naklonenom ramene vyvíja tlak na potenciometer, ktorý je umiestnený v plastovom puzdre na hliníkovom profile ramena (viď modrý pruh na obr. 13).



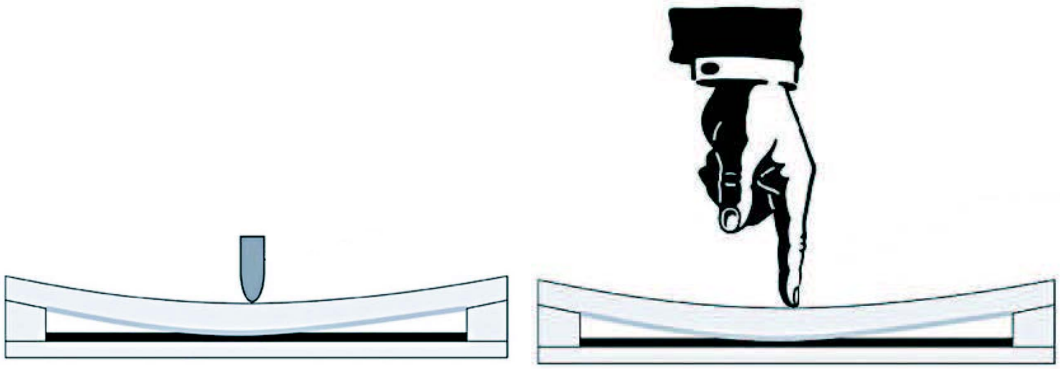
Obr. 14: Schéma zapojenia potenciometra v systéme ABMS

Potenciometer je zapojený ako delič napätia (obr. 14), pričom poloha gule na ramene sa určuje rozdielom napätí na svorkách potenciometra. V súčasnosti používame oceľovú guľu s priemerom $36,4mm$ a hmotnosťou $0,2kg$, ktorá je schopná vyvinúť dostatočný tlak na membránu. Potenciometer použitý v sústave ABMS má dĺžku aktívnej plochy $200mm$ a menovitý odpor súčiastky je $10k\Omega$.

Nevýhodou tejto metódy je, že hmotnosť gule spôsobuje zatlačenie do membrány, čo vedie k strate citlivosti v dôsledku suchého trenia, najmä pri malých vychýleniach ramena. Tento efekt je potrebné zohľadniť pri návrhu riadiacej jednotky, aby sa minimalizovali jeho negatívne vplyvy a zabezpečila optimálna presnosť merania polohy.

V systéme ABMS je v súčasnosti implementovaný potenciometer *Spectra Symbol ThinPot* (obr. 15). Ide o trojpinový odporový prvok, ktorý sa skladá z vodivého rezistora a jazdca – kolektora v uzavretom puzdre. Jazdec môže byť reprezentovaný ľubovoľným nevodivým mechanizmom – valec, koleso, guľa, ľudský prst a pod., ktorý vonkajším zásahom stláča vrchný obvod – kolektor smerom nadol, čím aktivuje potenciometer (obr. 16).

Obr. 15: Lineárny membránový potenciometer Spectra Symbol ThinPot



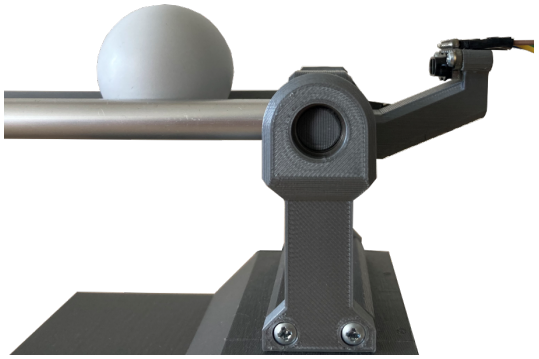
Obr. 16: Aktivácia potenciometra vonkajším zásahom [71]

Bezkontaktné snímanie

Sekundárnou metódou určenia polohy gule je bezkontaktné meranie vzdialenosti pomocou laserového senzora ST VL53L1X. Snímač VL53L1X využíva technológiu Time of Flight (ToF) na určenie vzdialenosti medzi senzorom a guľou.

Redundancia senzorov polohy môže byť využitá pri porovnávaní rôznych metód určovania polohy gule. Môže tiež slúžiť ako doplnkový, pomocný spôsob určenia polohy gule pri riadení systému.

Pri určovaní polohy gule je potrebné zohľadniť rozdiel medzi kontaktným a bezkontaktným meraním. Potenciometer určuje polohu stredu gule, zatiaľ čo laserový senzor meria vzdialenosť k jej povrchu, pričom treba započítať aj polomer gule. Táto rozdielnosť spôsobuje posun nulových polôh oboch snímačov, ktorý je nutné kompenzovať softvérovo alebo hardvérovo. Navyše, laserový senzor má minimálny detekčný prah $4cm$, ktorý bol vyriešený použitím adaptéra na zabezpečenie dostatočného odstupu od nulovej polohy (obr. 17).



Obr. 17: Detail umiestnenia laserového senzora pomocou adaptéra

Pohyb ramena

Pohyb ramena v systéme zabezpečuje aktuátor – servomotor, ktorý ovláda jeho sklon a tým aj pohyb gule. Môže ísť o digitálny alebo analógový servomotor riadený PWM signálom. Servomotor je pripevnený modulárnym a jednoducho vymeniteľným komponentom. Pri vývoji sa testovali digitálne servomotory Waveshare ST3020 a SC15, ktoré majú vyšší krútiaci moment a presnejšie nastavenie uhla, no vyžadujú dodatočný UART prevodník a samostatné napájanie.

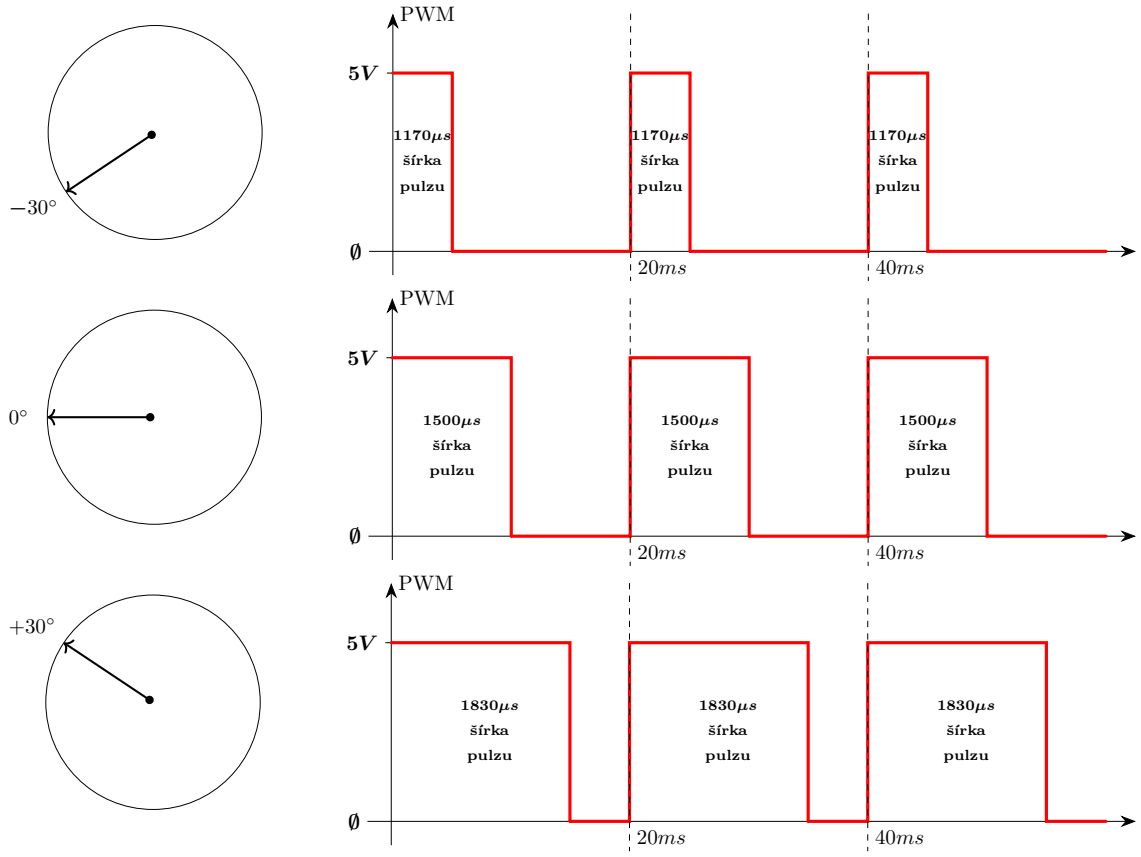
Druhou testovanou možnosťou boli analógové servomotory Waveshare MG996R a JX RD-5622-MG, ktoré sa ovládajú pomocou PWM signálu. Ich hlavnou výhodou je jednoduchšia integrácia, keďže nevyžadujú dodatočné komunikačné rozhranie. Parametre týchto servomotorov spolu s digitálnymi modelmi sú uvedené v tabuľke 2.

Tabuľka 2: Prehľad vybraných parametrov použitých servomotorov

	ST3020 [72]	SC15 [73]	MG996R [74]	RD-5622-MG [75]
max. točivý moment	$25kg.cm$	$17kg.cm$	$9kg.cm$	$22kg.cm$
uhol natočenia	360°	180°	180°	180°
rýchlosť otočenia	$0.16s/60°$	$0.167s/60°$	$0.18s/60°$	$0.18s/60°$
rozsah pozícií	360°/4096	180°/1024	180°/255	180°/255
min. otočenie	0.018°	0.17°	0.7°	0.7°

Momentálne sú dva používané systémy osadené servomotorom JX RD-5622-MG, ktorý má pracovný rozsah natočenia hriadeľa od $-90°$ do $+90°$, podobne ako MG996R alebo SC15. V systéme ABMS sa využíva len časť tohto rozsahu, konkrétne $(-30°, +30°)$, čo zlepšuje presnosť nastavenia sklonu ramena pri zachovaní rozlíšenia signálu. Podrobnejší príklad, ktorý demonštruje vplyv šírky impulzu vstupného signálu na otočenie servomotora, ilustruje publikácia Kajaman [76].

Možnosť zameniteľnosti digitálneho a analógového servomotora rozširuje flexibilitu systému a umožňuje porovnávanie rôznych aktuátorov v experimentoch, čím sa zvyšuje variabilita použitia systému.



Obr. 18: Vplyv šírky impulzu PWM signálu na natočenie motora.

Riadiaca jednotka

Riadiaca jednotka systému ABMS zabezpečuje komunikáciu so senzormi, spracovanie dát a ovládanie aktuátora. Konštrukcia je kompatibilná s vývojovými doskami Arduino Due (s procesorom Atmel SAM3X8E ARM Cortex-M3) a STM32 Nucleo64 (s procesorom STM32 ARM Cortex-M4 F446RE). Obidve platformy používajú 32-bitové ARM mikrokontroléry. Mikrokontrolér je umiestnený pod konštrukciou krytu, ktorý zároveň umožňuje montáž podpier výkyvného ramena.

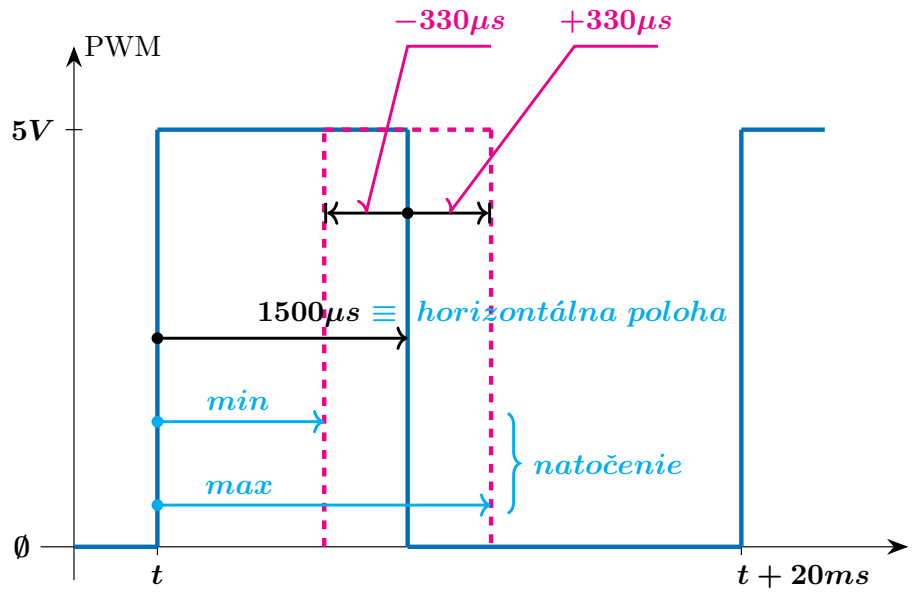
5.2 Komunikácia s experimentálnym zariadením

Komunikácia so sústavou ABMS bola navrhnutá pre integráciu do vzdialeného laboratória IOLab a inšpirovaná systémami TOM1A a uDAQ28/LT. Realizuje sa cez sériové

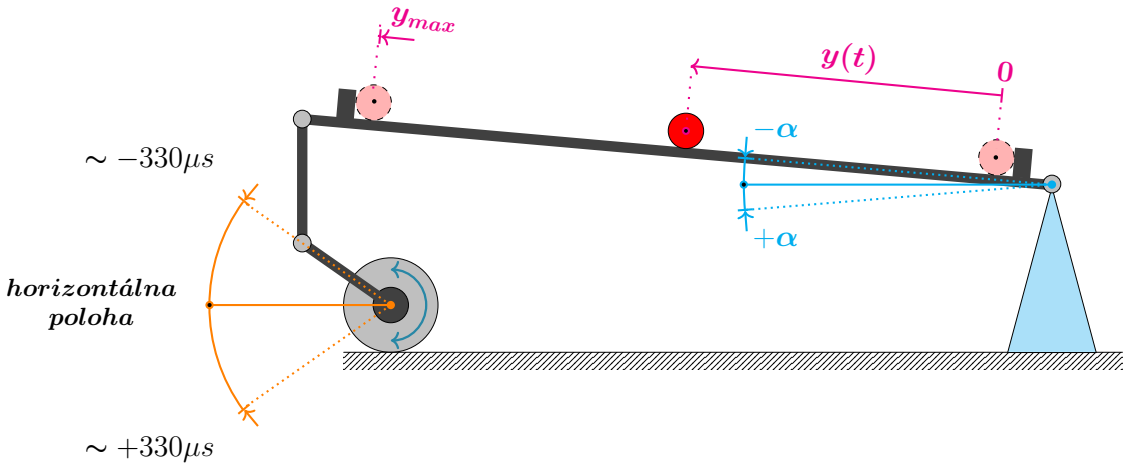
rozhranie s USB pripojením a využíva textový komunikačný protokol v ASCII kódovaní, ktorý umožňuje nastavovať uhol aktuátora a čítať údaje zo senzorov.

Navrhnutý systém využíva servosystém JX RD-5622-MG, ktorý na vstupe akceptuje PWM signál s frekvenciou $f_{PWM} = 50Hz$. Natočenie aktuátora sa riadi nastavením trvania logickej hodnoty 1 v mikrosekundách (μs). Nulová (horizontálna) poloha zodpovedá impulzu 1500 μs, pričom rozsah ± 330 μs umožňuje uhol natočenia ramena ABMS $\alpha = \pm 5°$.

V rámci úloh tejto práce nebola riešená identifikácia systému a návrh regulátora pre zariadenie ABMS. Tejto problematike sa venuje publikácia [P1].



Obr. 19: PWM signál pre servosystém ABMS



Obr. 20: Schéma systému ABMS s intervalmi natočenia aktuátora a ramena

Komunikačný protokol

Komunikačný protokol obsahuje príkaz na nastavenie sklonu ramena, ktorý umožňuje presne definovať smer a uhol natočenia aktuátora vzhľadom na horizontálnu polohu. Hodnota parametra `NUM` musí byť celé číslo v rozsahu $< -330, +330 >$, kde kladné čísla určujú natočenie nahor a hodnota 0 nastaví rameno do horizontálnej polohy. Príkaz obsahuje aj softvérový bezpečnostný mechanizmus, ktorý obmedzuje maximálne prírastky natočenia, aby chránil zariadenie pred nesprávnou manipuláciou a poškodením.

```
"SRV:NUM\n"
```
(1)

```
"SRV:330\n"
```
(2)

Systém ABMS generuje výstupné správy s informáciami o aktuálnej pozícii gule a natočení aktuátora, čo umožňuje používateľovi sledovať dynamiku systému v reálnom čase. Na príkaz nastavenia natočenia odpovedá textovým reťazcom obsahujúcim identifikátor správy (`POS`), údaje o polohe gule v milimetroch zo snímačov a hodnotu trvania logickej 1 PWM signálu (`SRV`) v mikrosekundách.

Výstupný reťazec rešpektuje formát (3), kde:

- `POT` udáva hodnotu kontaktného senzora - potenciomentra,
- `TOF` udáva hodnotu bezkontaktného senzora - laserového snímača,

```
"POS:POT;TOF;SRV\n"
```
(3)

V aktuálnej verzii firmvéru ABMS udávajú senzory pozície hodnotu vzdialenosti gule od bodu výkyvu ramena. V prípade, že snímač nie je nainštalovaný alebo vykazuje chybu, hodnota premennej konkrétneho snímača je `NA` (*not available*). Príklad možných výstupov ilustrujú (4), (5) a (6)

```
"POS:122;126;1450\n"
```
(4)

```
"POS:NA;126;1500\n"
```
(5)

```
"POS:90;NA;1250\n"
```
(6)

Podpora simulačných prostredí

Pre efektívne testovanie a overovanie funkcií systému ABMS bola realizovaná jeho integrácia do simulačných prostredí, ktorá umožňuje lokálne aj vzdialené experimentovanie. Kľúčovým prvkom je vlastný komunikačný protokol zabezpečujúci spoľahlivý a rozšíriteľný prenos dát medzi zariadením a počítačom s dôrazom na nízku latenciu.

Zariadenie ABMS bolo integrované s dvoma prostrediami: MATLAB/Simulink (komerčné) a pysimCoder (open-source), pričom pre obe boli vyvinuté funkcie a rozhrania na plnohodnotnú interakciu a zber dát v reálnom čase.

Integrácia s MATLAB/Simulink

Implementácia pre MATLAB bola vytvorená v jazyku MATLAB s využitím *Level-2 MATLAB S-Function*. Pre zariadenie boli vyvinuté:

- skripty a funkcie na komunikáciu so zariadením, čítanie/zápis údajov, konfiguráciu protokolu a demonštračný súbor na ich ukážku,
- vlastný Simulink blok založený na *Level-2 MATLAB S-Function*, ktorý umožňuje integráciu zariadenia do blokových diagramov a podporuje zadanie vstupných parametrov pre kompatibilitu s rôznymi verziami hardvéru a firmvéru.

Integrácia s pysimCoder

Implementácia systému ABMS pre prostredie pysimCoder bola rozdelená na dve vrstvy:

- vnútorná logika (spracovanie vstupov a komunikácia cez protokol) bola implementovaná v jazyku C kvôli kompatibilite s vnorenými systémami a platformou Linux, na ktorej je pysimCoder postavený,
- vizuálne prvky (blokový editor a GUI) boli vytvorené v Pythone, ktorý tvorí základ grafického rozhrania pysimCoder a umožňuje intuitívne vytváranie blokových schém a simulácií.

Implementácia vzdialeného experimentovania

Vzdialené experimentovanie so zariadením ABMS je realizované integráciou hardvéru so simulačnými prostrediami. V MATLAB/Simulink sa využíva *Level-2 MATLAB S-Function* na prepojenie blokovej schémy s fyzickým zariadením cez sériové rozhranie a synchronizáciu dát, pričom *MATLAB Engine API for Python* umožňuje serverovej aplikácii experiment riadiť a spracovávať dáta v reálnom čase.

V prostredí pysimCoder je integrácia založená na blokovej schéme a Python skriptoch, ktoré konfigurujú experiment, generujú spustiteľný kód a riadia komunikáciu medzi softvérom a hardvérom.

Integrácia zariadenia do vzdialeného laboratória využíva jednotnú architektúru obojsmernej komunikácie medzi hardvérom, simulačným prostredím a webovou aplikáciou. Webová aplikácia poskytuje používateľom rozhranie na nastavenie parametrov, spustenie experimentu a sledovanie výsledkov v reálnom čase, zatiaľ čo server so spusteným simulačným softvérom (MATLAB alebo pysimCoder) zabezpečuje riadenie zariadenia a prenos meraných dát späť na vizualizáciu a analýzu.

6 Zovšeobecnenie postupu návrhu a integrácie experimentov do online laboratória

Online a vzdialené laboratóriá poskytujú moderný spôsob výučby technických a prírodovedných disciplín s prístupom k reálnym experimentom cez internet. S rastúcim záujmom o tento prístup narastá potreba vyvíjať technicky spoľahlivé, pedagogicky hodnotné a flexibilné experimenty. Pri ich návrhu je kľúčové zohľadniť modularitu a škálovateľnosť, ktoré podporujú udržateľnosť, jednoduchú rozšíriteľnosť a opätovnú použiteľnosť systémov. Kapitola sa zameriava na systematický prístup k návrhu a integrácii experimentov založený na otvorených technológiách a kompatibilite s rôznymi prostrediami, čo umožňuje efektívny rozvoj a správu online laboratórií.

6.1 Požiadavky na experiment v online laboratóriu

Návrh experimentu pre online laboratórium musí zohľadňovať nielen technickú realizovateľnosť a spoľahlivosť, ale aj jeho hodnotu a používateľskú dostupnosť. Pri vývoji je nevyhnutné identifikovať a reflektovať kľúčové kategórie požiadaviek, ktoré ovplyvňujú funkčnosť a prevádzku zariadení.

Technické požiadavky

Hlavné technické požiadavky na návrh experimentov pre vzdialené laboratóriá zahŕňajú kompatibilitu, spoľahlivosť a bezpečnosť hardvéru a softvéru. Experiment musí byť schopný pracovať s rôznymi riadiacimi jednotkami, komunikačnými protokolmi a vizualizačnými nástrojmi, pričom využíva štandardizované rozhrania pre jednoduchú integráciu. Zariadenie ABMS používa štandardné USB rozhranie, čo uľahčuje jeho nasadenie v rôznych infraštruktúrach.

Spoľahlivá obojsmerná komunikácia medzi používateľom a experimentom je kľúčová, pričom moderné systémy využívajú webové technológie umožňujúce súbežný prístup viacerých používateľov a podporujúce širokú škálu klientskych zariadení. Riešenie IOLab je založené na otvorených webových štandardoch, čím zabezpečuje vysokú kompatibilitu s rôznymi platformami vrátane PC a mobilných zariadení.

Pri experimentoch v reálnom čase je nevyhnutné minimalizovať oneskorenia medzi používateľskou interakciou a odozvou systému a zároveň zabezpečiť odolnosť voči výpadkom spojenia, elektrickej energie či chybným vstupom bez poškodenia hardvéru. Laboratórne riešenie IOLab implementuje viaceré úrovne validácie používateľských vstupov a umožňuje dávkové vykonávanie experimentov po obnove napájania. Dáta zo

spustených experimentov sú pri prerušení spojenia uchovávané na serveri, čo umožňuje ich neskoršie získanie.

Zariadenie ABMS disponuje softvérovými mechanizmami na ochranu pred poškodením pri chybných vstupoch a zabezpečuje automatický návrat do počiatočného stavu po obnovení spojenia. Napájanie a dátový prenos sú realizované cez jediný USB kábel, pričom synchronizované napájanie zariadení a serveru zvyšuje spoľahlivosť systému.

Prevádzkové a systémové požiadavky

Udržateľnosť a rozvoj riešení pre vzdialené experimenty si vyžadujú zohľadnenie nielen v etapách návrhu a implementácie, ale aj počas celej prevádzky laboratória. Kľúčové je zabezpečiť obsluhu viacerých používateľov prostredníctvom rezervačných systémov a umožniť konfiguráciu a aktualizáciu experimentov bez fyzického zásahu, ideálne s podporou verziovania softvéru.

Súčasná verzia zariadenia ABMS zatiaľ neumožňuje monitorovanie stavu experimentu ani vzdialenú aktualizáciu firmvéru, tieto funkcie sú však plánované v budúcom vývoji. Architektúra IOLab poskytuje možnosti videoprenosu a režimu údržby, v ktorom možno naplánovať automatizované úlohy (napr. aktualizácie) v čase minimálnej záťaže, čím sa minimalizuje riziko prerušenia používateľských relácií a zvyšuje robustnosť systému.

6.2 Modulárna architektúra experimentu

Modularita je zásadným princípom pri návrhu experimentov pre vzdialené laboratóriá, pretože zvyšuje efektivitu vývoja, udržateľnosť a rozšíriteľnosť systémov. Modularita umožňuje rozdeliť systém na nezávislé časti, ktoré možno samostatne vyvíjať, testovať a meniť bez zásahu do ostatných komponentov. Dôraz na otvorenosť zároveň zabezpečuje, že úpravy a rozšírenia je možné realizovať lokálne s využitím dostupných prostriedkov, bez závislosti od výrobcu alebo spoplatnených riešení.

Komponenty modulárneho experimentálneho systému

Experimentálne zariadenie pre vzdialené laboratórium možno charakterizovať ako súbor základných modulov:

- fyzický model (hardvérový modul),
- riadiaci a komunikačný modul,
- softvérový modul.

Hardvérový modul zahŕňa samotné zariadenie – mechanickú a elektrickú sústavu vrátane snímačov, aktuátorov a konštrukčných komponentov. V prípade systému ABMS sú

tieto komponenty navrhnuté ako modifikovateľné a ľahko replikovateľné prostredníctvom 3D tlače, pričom funkčné prvky je možné flexibilne nahrádzať ekvivalentmi.

Riadiaci a komunikačný modul zabezpečuje spracovanie používateľských vstupov, spúšťanie algoritmov a prepojenie medzi fyzickým zariadením a používateľom prostredníctvom štandardizovaných alebo vlastných protokolov; jeho súčasťou je aj otvorený firmvér dostupný vo verejnom repozitári.

Softvérový modul sa stará o interakciu s používateľom, validáciu vstupov a vizualizáciu dát prostredníctvom funkčných knižníc (napr. pre MATLAB, Simulink, Python) alebo webových rozhraní. Pre úplnosť systému je dôležitá aj sprievodná dokumentácia obsahujúca informácie o konfigurácii, verziách a metadátach zariadenia, ktorá podporuje jeho správu a ďalší rozvoj.

6.3 Škálovateľnosť experimentov

Škálovateľnosť experimentálneho systému predstavuje jeho schopnosť adaptovať sa na rastúce požiadavky a rozširujúce sa aplikačné scenáre. V kontexte vzdialených laboratórií zahŕňa predovšetkým možnosť rozšírenia funkcionality, modernizácie hardvérových komponentov a automatizácie procesov. Takto koncipovaný systém umožňuje pridávať nové typy vstupov, režimy ovládania, vizualizácie či experimentálne scenáre bez zásahov do základnej infraštruktúry, čím sa podporuje jeho využitie na rôznych úrovniach vzdelávania, od stredoškolskej výučby až po vysokoškolské a výskumné prostredie. Dôraz na škálovateľnosť v návrhovej fáze je kľúčový pre vytvorenie flexibilného a dlhodobo udržateľného laboratórneho prostredia, ktoré reflektuje rozmanité potreby používateľov a dynamický technologický vývoj.

6.4 Zabezpečenie kompatibility a rozšíriteľnosti

Zabezpečenie kompatibility a rozšíriteľnosti predstavuje kľúčový predpoklad pre dlhodobú udržateľnosť a evolúciu platformy.

Hardvérová vrstva systému by mala využívať štandardizované konektory, napájacie a komunikačné rozhrania, čím sa umožní bezproblémová výmena komponentov a minimalizuje sa potreba zásahov do vyšších softvérových vrstiev.

Na softvérovej úrovni je nevyhnutné definovať jednoznačné aplikačné rozhrania medzi jednotlivými vrstvami systému, čo zabezpečuje izoláciu zmien a kompatibilitu naprieč rôznymi simulačnými prostrediami a operačnými systémami. Dôraz na spätnú kompatibilitu a modulárnu architektúru umožňuje efektívnu integráciu nových funkcionalít a komponentov bez nutnosti budovania unikátnych riešení.

Záver

Dizertačná práca sa zaoberala návrhom a implementáciou riešení pre vzdialené experimentovanie s dôrazom na využitie otvorených technológií v oblasti riadenia mechatronických systémov. V úvodnej časti bola analyzovaná problematika online laboratórií z hľadiska typov experimentov, existujúcich architektúr (peer-to-peer, IoT, klient-server, SOA, cloud a hybridné riešenia) a dostupných simulačných prostredí. Táto analýza poukázala na kľúčové výhody a limity jednotlivých prístupov a vytvorila východisko pre návrh unifikovanej architektúry.

Navrhnutá architektúra vzdialeného laboratória sa opiera o otvorené softvérové a hardvérové komponenty. Jej základom je modulárny experimentový server, ktorý sprostredkúva obojsmernú komunikáciu medzi klientskou webovou aplikáciou a experimentálnym zariadením a umožňuje synchronizáciu dát v reálnom čase. Architektúra bola navrhnutá tak, aby podporovala rôzne typy experimentov vrátane simulácií v prostrediach ako MATLAB alebo pysimCoder.

Riešenie bolo overené implementáciou experimentu s automatickým balančným mechatronickým systémom (ABMS). Tento experiment demonštroval riadenie systému s využitím senzorovej redundancie a prezentoval praktické možnosti synchronizácie medzi reálnym zariadením a simulačným prostredím.

V závere bola spracovaná metodológia návrhu a integrácie nových experimentov do online laboratória s dôrazom na modularitu, škálovateľnosť, bezpečnosť a kompatibilitu. Takto navrhnutý prístup umožňuje efektívnu integráciu experimentov a vytvára predpoklady pre rozšírenie systému o nové zariadenia bez nutnosti zásahu do základnej architektúry.

Hlavné prínosy dizertačnej práce môžeme zovšeobecniť do nasledujúcich bodov:

- vytvorenie nového funkčného mechatronického systému využívajúceho senzorovú redundanciu, ktorý je navrhnutý ako otvorený, personalizovateľný a kompatibilný s rozhraniami a simulačnými prostrediami uvažovanými v rámci navrhnutej architektúry,
- vytvorenie metódy integrácie zariadení do otvoreného simulačného softvéru, ktorá je plne aplikovateľná v kontexte vytvorenej architektúry vzdialeného laboratória,
- prestavba architektúry experimentového servera so zreteľom na otvorenosť poskytovaných SDK a API, čím sa dosiahla vyššia flexibilita a interoperabilita systému,

- návrh metodiky umožňujúcej efektívnu integráciu existujúcich experimentov do online laboratória, ktorá podporuje modulárnosť a škálovateľnosť navrhnutého riešenia.

Potenciálne vylepšenia zahŕňajú najmä prechod na distribuovanú cloudovú architektúru, ktorá umožní paralelnú prácu viacerých používateľov a modernizáciu experimentálneho hardvéru s cieľom zvýšiť energetickú efektívnosť a spoľahlivosť. Práca tak vytvára pevný základ pre budúci výskum v oblasti automatizácie, distribuovaných riadiacich systémov a otvorených platforiem pre vzdialené experimentovanie.

Literatúra

1. GOMES, L.; BOGOSYAN, S. Current Trends in Remote Laboratories. *IEEE Transactions on Industrial Electronics*. 2009, **56**(12), 4744–4756. Dostupné z DOI: 10.1109/TIE.2009.2033293.

2. RODRIGUEZ-GIL, L.; GARCÍA-ZUBIA, J.; ORDUÑA, P.; LÓPEZ-DE-IPIÑA, D. Towards New Multiplatform Hybrid Online Laboratory Models. *IEEE Transactions on Learning Technologies*. 2017, **10**(3), 318–330. Dostupné z DOI: 10.1109/TLT.2016.2591953.

3. MUÑOZ, G. M. H.; FLORES, L. S. R.; IBARRA, F. M.; VELAZQUEZ, A. R. R. Simulators and virtual laboratories in online engineering education: A student perspective. In: *2022 XII International Conference on Virtual Campus (JICV)*. 2022, s. 1–3. Dostupné z DOI: 10.1109/JICV56113.2022.9934211.

4. VINOGRADOV, M. M.; ANTANENKOVA, I. S.; USTYUZHANIN, E. E. Methods of Measuring Temperature and Pressure on Virtual Laboratory Stands. In: *2022 VI International Conference on Information Technologies in Engineering Education (Inforino)*. 2022, s. 1–5. Dostupné z DOI: 10.1109/Inforino53888.2022.9782932.

5. MARTELL-CHAVEZ, F.; LÓPEZ-TÉLLEZ, J. M.; PAREDES-ORTA, C. A.; ESPINOSA-LUNA, R. Virtual laboratories for teaching automation, robotics, and optomechatronics. In: *2023 IEEE World Engineering Education Conference (EDUNINE)*. 2023, s. 1–4. Dostupné z DOI: 10.1109/EDUNINE57531.2023.10102906.

6. AGARWAL, B.; RAVICHANDRA, L.; KUMAR, M.; THAKKAR, P. D.; M, V. Design and Implementation of Virtual Laboratory using Unity with a Web Interface. In: *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*. 2024, s. 728–732. Dostupné z DOI: 10.1109/ICETSIS61505.2024.10459676.

7. JONG, T.; LINN, M.; ZACHARIA, Z. Physical and Virtual Laboratories in Science and Engineering Education. *Science (New York, N.Y.)* 2013, **340**, 305–308. Dostupné z DOI: 10.1126/science.1230579.

8. BRINSON, J. R. Learning outcome achievement in non-traditional (virtual and remote) versus traditional (hands-on) laboratories: A review of the empirical research. *Computers & Education*. 2015, **87**, 218–237. ISSN 0360-1315. Dostupné z DOI: https://doi.org/10.1016/j.compedu.2015.07.003.

9. HUSSEIN, R.; WILSON, D. Remote Versus In-hand Hardware Laboratory in Digital Circuits Courses. In: *2021 ASEE Virtual Annual Conference Content Access*. Virtual Conference: ASEE Conferences, 2021. Č. 10.18260/1-2–37662. https://peer.asee.org/37662.

10. MA, J.; NICKERSON, J. Hands-on, simulated, and remote laboratories: A comparative literature review. *ACM Comput. Surv.* 2006, **38**. Dostupné z DOI: 10.1145/1132960.1132961.

11. ALVES, G. et al. Remote experimentation network - yielding an inter-university peer-to-peer e-service. In: *2005 IEEE Conference on Emerging Technologies and Factory Automation*. 2005, zv. 2, 8 pp.–1030. ISBN 0-7803-9401-1. Dostupné z DOI: 10.1109/ETFA.2005.1612784.

12. KIST, A. A. et al. Overlay network architectures for peer-to-peer Remote Access Laboratories. In: *2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. 2014, s. 274–280. Dostupné z DOI: 10.1109/REV.2014.6784274.

13. MAITI, A.; KIST, A. A.; MAXWELL, A. D. Real-Time Remote Access Laboratory With Distributed and Modular Design. *IEEE Transactions on Industrial Electronics*. 2015, **62**(6), 3607–3618. Dostupné z DOI: 10.1109/TIE.2014.2374572.

14. KUSTIJA, J.; JAYANTO, N. D. IoT Implementation for Development of Remote Laboratory (Case Study on Microscope Practice). *REKA ELKOMIKA: Jurnal Pengabdian kepada Masyarakat*. 2022, **3**(1), 20–29.

15. EL-HASAN, T. S. Internet of Thing (IoT) Based Remote Labs in Engineering. In: *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*. 2019, s. 976–982. Dostupné z DOI: 10.1109/CoDIT.2019.8820591.

16. RAMYA, M. V.; PURUSHOTHAMA, G. K.; PRAKASH, K. R. Design and Implementation of IoT Based Remote Laboratory for Sensor Experiments. *International Journal of Interactive Mobile Technologies (iJIM)*. 2020, **14**(09), pp. 227–238. Dostupné z DOI: 10.3991/ijim.v14i09.13991.

17. AZAD, A. K. Use of Internet of Things for Remote Laboratory Settings. In: *ASEE Annual Conference and Exposition, Conference Proceedings*. 2020. Dostupné z eprint: 21535965.

18. AZAD, A. K. M. Design and development of remote laboratories with internet of things setting. *Advances in Internet of Things*. 2021, **11**(3), 95–112.

19. TRAINI, E.; AWOUDA, A.; ASRANOV, M.; CHIABERT, P. Open-Source IoT Lab for Fully Remote Teaching. In: CANCIGLIERI JUNIOR, O.; NOËL, F.; RIVEST, L.; BOURAS, A. (ed.). *Product Lifecycle Management. Green and Blue Technologies to Support Smart and Sustainable Organizations*. Cham: Springer International Publishing, 2022, s. 353–368. ISBN 978-3-030-94399-8. Dostupné z DOI: 10.1007/978-3-030-94399-8_26.

20. AL-OBAIDI, M. Q.; DERBEL, N. Design of IoT Based Remote Renewable Energy Laboratory. *International Journal of Emerging Technologies in Learning (iJET)*. 2023, **18**(12), pp. 75–87. Dostupné z DOI: 10.3991/ijet.v18i12.38659.

21. MENDES, L. A.; LI, L.; BAILEY, P. H.; DELONG, K. R.; ALAMO, J. A. del. Experiment lab server architecture: A web services approach to supporting interactive LabVIEW-based remote experiments under MIT's iLab shared architecture. In: *2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. 2016, s. 293–305. Dostupné z DOI: 10.1109/REV.2016.7444486.

22. KURIŠČÁK, P.; ROSSA, P.; FERNANDES, H.; SILVA, J. N. Design and implementation of a Framework for remote experiments in education. In: *2022 8th International Engineering, Sciences and Technology Conference (IESTEC)*. 2022, s. 258–265. Dostupné z DOI: 10.1109/IESTEC54539.2022.00046.

23. CALLAGHAN, M. J.; HARKIN, J.; MCGINNITY, M.; MAGUIRE, L. Client-Server Architecture for Remote Experimentation for Embedded Systems. *International Journal of Online and Biomedical Engineering (iJOE)*. 2006, **2**(4). Dostupné z DOI: 10.3991/ijoe.v2i4.339.

24. ZÁRATE-MOEDANO, R.; CANCHOLA-MAGDALENO, S. L.; ARRINGTON-BÁEZ, A. A. Remote Laboratory, Based on Raspberry Pi, to Facilitate Scientific Experimentation for Secondary School Students. *International Journal of Online and Biomedical Engineering (iJOE)*. 2021, **17**(14), 154–163. Dostupné z DOI: 10.3991/ijoe.v17i14.25525.

25. CALLAGHAN, M.; HARKIN, J.; MCCOLGAN, E.; MCGINNITY, T.; MAGUIRE, L. Client–server architecture for collaborative remote experimentation. *Journal of Network and Computer Applications*. 2007, **30**(4), 1295–1308. ISSN 1084-8045. Dostupné z DOI: https://doi.org/10.1016/j.jnca.2006.09.006. Special issue on Information technology.

26. OCAYA, R. A framework for collaborative remote experimentation for a physical laboratory using a low cost embedded web server. *Journal of Network and Computer Applications*. 2011, **34**(4), 1408–1415. ISSN 1084-8045. Dostupné z DOI: https://doi.org/10.1016/j.jnca.2011.03.024. Advanced Topics in Cloud Computing.

27. HERRERA, O.; FULLER, D. Collaborative model for remote experimentation laboratories used by non-hierarchical distributed groups of engineering students. *Australasian Journal of Educational Technology*. 2011, **27**, 428–445. Dostupné z DOI: 10.14742/ajet.953.

28. XIE, C.; LI, C.; SUNG, S.; JIANG, R. Engaging Students in Distance Learning of Science With Remote Labs 2.0. *IEEE Transactions on Learning Technologies*. 2022, **15**, 1–1. Dostupné z DOI: 10.1109/TLT.2022.3153005.

29. VANEGAS-GUILLÉN, O.; PARRA-ROSERO, P.; MUÑOZ-ANTÓN, J. M.; ZUMBA-GAMBOA, J.; DILLON, C. Remote Labs Meet Computational Notebooks: An Architecture for Simplifying the Workflow of Remote Educational Experiments. *IEEE Access*. 2023, **11**, 132496–132515. Dostupné z DOI: 10.1109/ACCESS.2023.3336287.

30. XING, Y.; YAO, E. Remote collaborative experiments based on service-oriented architecture(SOA). In: *2010 2nd International Conference on Signal Processing Systems*. 2010, zv. 2, s. V2-605-V2–608. Dostupné z DOI: 10.1109/ICSPS.2010.5555722.

31. MOUSSA, M.; BENACHENHOU, A.; BELGHIT, S.; ADDA BENATTIA, A.; BOUMEHDI, A. An Implementation of Microservices Based Architecture for Remote Laboratories. In: AUER, M. E.; MAY, D. (ed.). *Cross Reality and Data Science in Engineering*. Cham: Springer International Publishing, 2021, s. 154–161. ISBN 978-3-030-52575-0. Dostupné z DOI: 10.1007/978-3-030-52575-0_12.

32. CORNETTA, G.; TOUHAFI, A.; TOGOU, M. A.; MUNTEAN, G.-M. Fabrication-as-a-Service: A Web-Based Solution for STEM Education Using Internet of Things. *IEEE Internet of Things Journal*. 2020, **7**(2), 1519–1530. Dostupné z DOI: 10.1109/JIOT.2019.2956401.

33. TAWFIK, M. et al. Laboratory as a Service (LaaS): a Novel Paradigm for Developing and Implementing Modular Remote Laboratories. *International Journal of Online and Biomedical Engineering (iJOE)*. 2014, **10**(4), 13–21. Dostupné z DOI: 10.3991/ijoe.v10i4.3654.

34. DA SILVA, A. F.; OHTA, R. L.; DOS SANTOS, M. N.; BINOTTO, A. P. A Cloud-based Architecture for the Internet of Things targeting Industrial Devices Remote Monitoring and Control. *IFAC-PapersOnLine*. 2016, **49**(30), 108–113. ISSN 2405-8963. Dostupné z DOI: https://doi.org/10.1016/j.ifacol.2016.11.137. 4th IFAC Symposium on Telematics Applications TA 2016.

35. ORDOÑEZ URBANO, C. F.; MUÑOZ, J. A.; FIERRO, L. P.; MARULANDA, J. F. F.; VARGAS-CAÑAS, R. Remote Laboratories for Physics Education: A Proposal Toward Interactive Learning for Engineering Students. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*. 2023, **18**(3), 305–312. Dostupné z DOI: 10.1109/RITA.2023.3303100.

36. ORDUÑA, P. et al. The WebLab-Deusto Remote Laboratory Management System Architecture: Achieving Scalability, Interoperability, and Federation of Remote Experimentation. In: *Cyber-Physical Laboratories in Engineering and Science Education*. Ed. AUER, M. E.; AZAD, A. K.; EDWARDS, A.; JONG, T. de. Cham: Springer International Publishing, 2018, s. 17–42. ISBN 978-3-319-76935-6. Dostupné z DOI: 10.1007/978-3-319-76935-6_2.

37. AGRAWAL, A.; SRIVASTAVA, S. WebLab: A Generic Architecture for Remote Laboratories. In: *15th International Conference on Advanced Computing and Communications (ADCOM 2007)*. 2007, s. 301–306. Dostupné z DOI: 10.1109/ADCOM.2007.71.

38. VILLAR-MARTÍNEZ, A. et al. Improving the Scalability and Replicability of Embedded Systems Remote Laboratories Through a Cost-Effective Architecture. *IEEE Access*. 2019, **7**, 164164–164185. Dostupné z DOI: 10.1109/ACCESS.2019.2952321.

39. VILLAR-MARTÍNEZ, A.; GARCÍA-ZUBÍA, J.; ANGULO, I.; RODRÍGUEZ-GIL, L. Towards Reliable Remote Laboratory Experiences: A Model for Maximizing Availability Through Fault-Detection and Replication. *IEEE Access*. 2021, **9**, 45032–45054. Dostupné z DOI: 10.1109/ACCESS.2021.3065742.

40. RÁBEK, M.; ŽÁKOVÁ, K. Simple experiment integration into modular online laboratory enviroment. In: *2017 4th Experiment@International Conference (exp.at'17)*. 2017, s. 264–268. Dostupné z DOI: 10.1109/EXPAT.2017.7984377.

41. RÁBEK, M.; ŽÁKOVÁ, K. Scalable Remote Experiment Manager. *IFAC-PapersOnLine*. 2020, **53**(2), 17246–17251. ISSN 2405-8963. Dostupné z DOI: https://doi.org/10.1016/j.ifacol.2020.12.1793. 21st IFAC World Congress.

42. RÁBEK, M.; ŽÁKOVÁ, K. Development of Control Experiments for an Online Laboratory System. *Information*. 2020, **11**(3). ISSN 2078-2489. Dostupné z DOI: 10.3390/info11030131.

43. MATHWORKS INC. *Comparison of Custom Block Functionality: Code Generation* [online]. [cit. 2025-01-08]. Dostupné z: https://www.mathworks.com/help/simulink/ug/comparison-of-custom-block-functionality.html#bvntyiv-1.

44. MATHWORKS INC. *Write Level-2 MATLAB S-Functions* [online]. [cit. 2025-01-08]. Dostupné z: https://www.mathworks.com/help/simulink/sfg/writing-level-2-matlab-s-functions.html.

45. TANENBAUM, A. S.; WETHERALL, D. J. *Computer Networks*. 5. vyd. Pearson, 2010. ISBN 9780132126953.

46. AXELSON, J. *USB Complete: The Developer's Guide*. 5. vyd. Lakeview Research, 2022. ISBN 9781931448284.

47. USB IMPLEMENTERS FORUM. *USB 2.0 Specifications* [online]. 2025. [cit. 2025-07-05]. Dostupné z: https://www.usb.org/document-library/usb-20-specification.

48. STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. 8. vyd. Pearson, 2022. ISBN 9780136794103.

49. NEDELKA, M. a. k. *Slovenský letecký slovník terminologický a výkladový*. Bratislava: Magnet-Press Slovakia, 2003. ISBN 80-968073-0-7.

50. WANG, W. *Control of a Ball and Beam System*. The University of Adelaide: School of Mechanical Engineering, 2007.

51. LIEBERMAN, J. A Robotic Ball Balancing Beam. In: 2004. Dostupné tiež z: https://api.semanticscholar.org/CorpusID:7793825.

52. HIRSCH, R. *EDUMECH Mechatronic Instructional Systems: Ball on Beam System*. Shandor Motion Systems, 1998.

53. ROSALES, E.; ITO, B.; LILIENKAMP, K.; LUNDBERG, K. An open-ended ball-balancing laboratory project for undergraduates. In: *Proceedings of the 2004 American Control Conference*. 2004, zv. 2, 1314–1318 vol.2. Dostupné z DOI: 10.23919/ACC.2004.1386756.

54. ITO, B. T. *Stabilizing the Ball on Beam System with Analog Feedback*. Massachusetts Institute of Technology: Department of Mechanical Engineering, 2004.

55. AMIRA GMBH DUISBURG. *BW500 Laboratory Setup Ball and Beam*. Duisburg, Germany: amira GmbH Duisburg, 1999.

56. TECQUIPMENT LTD. *Ball and Beam Apparatus - A self-contained, benchtop apparatus to demonstrate basic and advanced principles of control in naturally unstable systems.* Long Eaton, Nottingham: TecQuipment Ltd., 2024.

57. CHIEN, T.-L.; CHEN, C.-C.; TSAI, M.-C.; CHEN, Y.-C. Control of AMIRA's ball and beam system via improved fuzzy feedback linearization approach. *Applied Mathematical Modelling*. 2010, **34**(12), 3791–3804. ISSN 0307-904X. Dostupné z DOI: https://doi.org/10.1016/j.apm.2010.03.020.

58. COLÓN, D.; ANDRADE, Y.; BUENO, A.; DINIZ, I.; BALTHAZAR, J. Modeling, Control and Implementation of a Ball and Beam System. In: 2013.

59. KAGAMI, R. M.; COSTA, G. K. da; UHLMANN, T. S.; MENDES, L. A.; FREIRE, R. Z. A Generic WebLab Control Tuning Experience Using the Ball and Beam Process and Multiobjective Optimization Approach. *Information*. 2020, **11**(3). ISSN 2078-2489. Dostupné z DOI: 10.3390/info11030132.

60. QUANSER INC. *Quanser Ball and Beam: User Manual*. Markham, Ontario: Quanser Inc., 2011.

61. EDIBON. *RYC-BB Ball and Beam Module: Tender Specifications* [online]. Móstoles, Madrid: Edibon, 2021 [cit. 2022-12-29]. Dostupné z: https://www.edibon.com/EQUIPOS/RYC-BB/RYC-BB_EN.doc.

62. GOOGOL TECHNOLOGY LTD. *Ball and Beam Control System*. Shenzen, China: Googol Technology Ltd., 2023. Dostupné tiež z: http://googoltech.com/down-384.html.

63. AL-DUJAILI, A. Q.; HUMAIDI, A. J.; PEREIRA, D. A.; IBRAHEEM, I. K. Adaptive backstepping control design for ball and beam system. *International Review of Applied Sciences and Engineering*. 2021.

64. KIM, Y.; KIM, S.-K.; AHN, C. K. Variable Cut-Off Frequency Observer-Based Positioning for Ball-Beam Systems Without Velocity and Current Feedback Considering Actuator Dynamics. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2021.

65. NOWOPOLSKI, K. Implementation of ball-and-beam control system as an instance of Simulink to 32-bit microcontroller interface. *Poznan University of Technology Academic Journals. Electrical Engineering*. 2013.

66. ALI, A. T.; M., A. A.; A., A. H.; TAHA, O. A.; A., A. N. Design and Implementation of Ball and Beam System Using PID Controller. *Automatic Control and Information Sciences*. 2017, **3**(1), 1–4. ISSN 2375-1630. Dostupné z DOI: 10.12691/acis-3-1-1.

67. HADIPOUR, M.; HOSSEINZADEH, A.; SADIDI, M. Investigation of the Stability of the Ball and Beam by the PID Controller. *Journal of Manufacturing Technology Management*. 2021, 51–58. Dostupné z DOI: 10.30495/admt.2021.1897179.1189.

68. VERSLOOT, J.; PARROTT, E.; DUBAY, R. Adaptive Control of a Ball and Beam System. In: *2020 IEEE International Systems Conference (SysCon)*. 2020.

69. ACROME LTD. *Acrome Ball and Beam: User Manual*. Maslak, Istanbul: Acrome Ltd., 2015.

70. BAO, X. et al. Extended State Observer Based Sliding Mode Control for Ball-Beam System. In: *2021 China Automation Congress (CAC)*. 2021, s. 7007–7012. Dostupné z DOI: 10.1109/CAC53003.2021.9728532.

71. SPECTRA SYMBOL. *Spectra Symbol ThinPot* [online]. [cit. 2024-02-13]. Dostupné z: https://www.spectrasymbol.com/linear-position-sensors/thin-form-linear-pots-thinpot.

72. WAVESHARE INTERNATIONAL LIMITED. *Motors/Servos: 25kg.cm Wide Range Voltage Serial Bus Servo* [online]. [cit. 2025-06-02]. Dostupné z: https://www.waveshare.com/product/st3020-servo.htm.

73. WAVESHARE INTERNATIONAL LIMITED. *Motors/Servos: SC15 17kg Large Torque Programmable Serial Bus Servo* [online]. [cit. 2025-06-02]. Dostupné z: https://www.waveshare.com/sc15-servo.htm.

74. WAVESHARE INTERNATIONAL LIMITED. *Motors/Servos: MG996R Servo* [online]. [cit. 2025-06-02]. Dostupné z: https://www.waveshare.com/catalog/product/view/id/3614/s/mg996r-servo/category/37/.

75. SHANTOU JIANXIAN ELECTRONIC TECHNOLOGY CO., LTD. *Robot Servo: RD-5622MG-180* [online]. [cit. 2025-06-02]. Dostupné z: http://www.jx-servo.com/en/Product/ROBOT/423.html.

76. KAJAMAN, A.; ALDEEB, E.; ALJAYAR, M. Design and Control of Ball-On-Plate Balancing Dynamic System using PID Controller. In: *International Conference on Technical Sciences (ICST)*. 2019, s. 182–187.

Zoznam publikácií autora

P1. ŠEFČÍK, J.; CHAMRAZ, Š.; ŽÁKOVÁ, K. Compensator with Weighted Input Parameters for Automatic Ball-Balancing Mechatronic System. *Electronics*. 2025, **14**(9). ISSN 2079-9292. Dostupné z DOI: https://doi.org/10.3390/electronics14091695.

P2. ŠEFČÍK, J.; ŽÁKOVÁ, K. The ABMS - Automatic Balancing Mechatronic System. *IFAC-PapersOnLine*. 2024, **58**(9), 253–256. ISSN 2405-8963. Dostupné z DOI: https://doi.org/10.1016/j.ifacol.2024.07.405. 18th IFAC Conference on Programmable Devices and Embedded Systems PDES 2024.

P3. ŽÁKOVÁ, K.; MATIŠÁK, J.; ŠEFČÍK, J. Contribution to PID and PIDA Interactive Educational Tools. *IFAC-PapersOnLine*. 2024, **58**(7), 115–119. ISSN 2405-8963. Dostupné z DOI: https://doi.org/10.1016/j.ifacol.2024.08.020. 4th IFAC Conference on Advances in Proportional-Integral-Derivate Control PID 2024.

P4. ŠEFČÍK, J.; ŽÁKOVÁ, K. Open-Source Based Remote Control of Thermo-Optical Plant uDAQ28/LT. In: AUER, M. E.; TSIATSOS, T. (ed.). *Smart Mobile Communication & Artificial Intelligence*. Cham: Springer Nature Switzerland, 2024, s. 334–341.

P5. ŠEFČÍK, J.; ŽÁKOVÁ, K. Multiplatform Support for uDAQ28/LT Thermo-Optical Plant. In: *2022 Cybernetics & Informatics (K&I)*. 2022, s. 1–5. Dostupné z DOI: 10.1109/KI55792.2022.9925959.

P6. HRYHAROUSKAYA, H.; ŽÁKOVÁ, K.; MATIŠÁK, J.; ŠEFČÍK, J. Interactive 3D Model of Ball on Plate. In: *2022 Cybernetics & Informatics (K&I)*. 2022, s. 1–5. Dostupné z DOI: 10.1109/KI55792.2022.9925958.

P7. ŠEFČÍK, J.; ŽÁKOVÁ, K. Open-source support for thermo-optical plant uDAQ28-LT. In: *ELITECH´23: 25th Conference of Doctoral Students*. Bratislava: Vydavateľstvo Spektrum STU, 2023. ISBN 978-80-227-5298-5.

P8. ŠEFČÍK, J.; ŽÁKOVÁ, K. Refurbishment of thermal-optical plant uDAQ28/LT for control education. In: *ELITECH´22: 24th Conference of Doctoral Students*. Bratislava: Vydavateľstvo Spektrum STU, 2022. ISBN 978-80-227-5192-6.