Scolopendra – Softvérové rozšírenie šesťnohej robotickej platformy

[1]Ján VAŠČÁK, [2]Samuel TITKO

[1,2]Katedra Kybernetiky a Umelej Inteligencie, Fakulta Elektrotechniky a Informatiky, Technická Univerzita v Košiciach, Slovenská Republika

[1]jan.vascak@tuke.sk, [2]samuel.titko@student.tuke.sk

Abstrakt **– Cieľom tejto práce je rozšíriť softvér hexapodického robota Scolopendra navrhnutého v bakalárskej práci. Zameriava sa na vytvorenie virtuálnych modelov vo vizualizácii a simulácii, grafického užívateľského rozhrania pre riadenie robota, softvérového balíka z celého projektu a architektúr prepájateľných aplikácií. Práca je rozdelená na teoretickú analýzu, popis aplikovaných technológií a praktickú implementáciu všetkých technológií do vlastného softvéru. V závere sú zhodnotené dosiahnuté ciele celej práce.**

Kľúčové slová **– grafické užívateľské rozhranie, komunikácia, simulácia, softvérový balík, virtuálny model, vizualizácia, Webots**

ÚVOD

Softvérové rozšírenie robota Scolopendra bude zamerané na vytvorenie usporiadanej štruktúry adresárov a súborov, ktoré majú vzájomný logický súvis a budú tak vytvárať vhodné prostredie pre vývoj aplikácií. Ďalej bude rozšírená o virtuálne modely v prostredí vizualizácie a simulácie. Virtuálne modely uľahčia prácu s modelom robota každému programátorovi, ktorý je neskúsený v oblasti elektroniky, 3D tlače, obsluhy robota a jeho servisu. Vývoj a testovanie softvéru tak bude možný aj bez jeho aplikovania na fyzický model. Pri vytváraní komplexného softvéru zloženého z viacerých aplikácií, ktoré musia navzájom komunikovať, bude potrebné implementovať štandardné komunikačné rozhranie. Posledným rozšírením softvéru bude vytvorenie aplikácie užívateľského rozhrania pre ovládanie ľubovoľného modelu robota, ktorého obsluha bude intuitívna a prijateľná aj pre ľudí bez programátorských znalostí.

I. PREHĽAD IMPLEMENTOVANÝCH TECHNOLÓGIÍ

V súčasnosti najpopulárnejšou technológiou na vytváranie prostredia pre vývoj robotického softvéru je middleware ROS (Robot Operating System). Má široké uplatnenie v praxi no jeho najväčšou nevýhodou je však viazanosť na Linuxové operačné systémy. Preto cieľom tejto kapitoly bude vybrať také technológie, ktoré budú nahrádzať softvér poskytovaný ROS-om ako sú RViz alebo Gazebo. Zároveň bude možné ich používať aj pod ostatnými operačnými systémami. Primárne však boli vyberané tie technológie, ktoré majú podporu operačného Windows 10 a Windows 11, ktoré nepodporujú prácu v ROS-e. Okrem toho v teoretickej analýze boli popísané grafické rozhrania, s čím súvisí aj výber vhodnej technológie na realizáciu konkrétnej aplikácie grafického ovládača. Poslednou požiadavkou na implementáciu bude zvoliť také komunikačné rozhranie, ktoré umožní komunikáciu medzi viacerými procesmi. Implementovanými technológiami sú knižnica Open3D, simulátor Webots, softvérový rámec WPF pre tvorbu grafických rozhraní a rozhranie socket.

A. Knižnica Open3D

Robotika a počítačová 3D grafika potrebovala pre vývoj knižnicu, ktorá by bola schopná rýchlo spracovať dáta formátov 3D súborov napríklad súbory **.stl*, **.obj*, **.pcd* a manipulovať s nimi za použitia grafického procesora (GPU) v počítači. Knižnica Open3D bola vytvorená na riešenie tohto problému. Podľa [1], dovtedy neexistovala žiadna tzv. *open-source* knižnica, ktorá by bola rýchla, jednoduchá na použitie, schopná efektívne načítať a vizualizovať 3D dáta pomocou niekoľkých riadkov kódu v súlade s modernými postupmi softvérového inžinierstva. Open3D podporuje rýchly vývoj softvéru pracujúceho s 3D dátami. Kľúčovou vlastnosťou pre zvolenie tejto knižnice je implementácia v jazyku Python a nezávislosť na operačnom systéme. Backend je naprogramovaný v jazyku C++, zatiaľ čo dátové štruktúry a funkcie v jazyku Python predstavujú frontendové

rozhranie. Vývojárovi aplikácie postačuje použiť jazyk Python na načítanie 3D objektov do vizualizácie a manipulovať tak s vloženými 3D objektami prostredníctvom API. [1] Samotná knižnica Open3D nedisponuje možnosťou priameho vytvárania virtuálnych modelov robotov, ale je dobrým základom na realizáciu interaktívneho vizualizačného nástroja. Tento nástroj nahradí funkcionalitu RVizu, ktorý je súčasťou ROS-u. Jeho úlohou bude zobrazovať virtuálny model robota a poskytovať možnosť ovládať ho prostredníctvom požiadaviek z komunikačného rozhrania.

B. Simulátor Webots

Webots je open source a multiplatformová desktopová aplikácia určená na simuláciu robotov vo virtuálnom prostredí. Simulátor je navrhnutý pre profesionálne použitie v modelovaní a simuláciách v priemysle, výskume a vzdelávaní. Prostredie simulátora poskytuje prostriedky na vytvorenie sveta a modelov robotov podľa vlastných potrieb. Okrem týchto prostriedkov vlastnej tvorby dáva možnosť vyskúšať existujúce simulácie modelov robotov, celé svety s modelmi robotov a ich predprogramovanými vzorovými ovládačmi. Zdrojový kód ovládača virtuálneho robota môže byť napísaný v jazykoch C, C++, Python, Java, MATLAB. [2]

V rámci riešenia softvérového rozšírenia bude vytvorený virtuálny model robota Scolopendra spolu so základným ovládačom a modulmi pre podporu jeho programovania. Webots tak nahradí funkcionalitu simulátora Gazebo, ktorý je rovnako ako RViz súčasťou ROS-u.

C. Softvérový rámec WPF

Windows Presentation Foundation (WPF) je softvérový rámec užívateľského rozhrania (z angl. User Interface – UI) určený na tvorbu GUI v podobe klientskych aplikácií pre operačný systém Windows. WPF je navyše rozšírený o značkovací jazyk XAML (eXtensible Application Markup Language). XAML je postavený na značkovacom štandarde jazyka XML podobne ako značkovací jazyk HTML (HyperText Markup Language), ktorý je určený na štrukturalizáciu webových stránok. Použitie XAML na vývoj používateľských rozhraní oddeľuje grafickú stránku aplikácie od aplikačnej logiky - frontend od backendu. XAML poskytuje deklaratívny model pre tvorbu dizajnu aplikácie rozhrania. [3][4]

Softvérový rámec WPF spolu so značkovacím jazykom XAML bude použitý pri vytváraní grafického ovládača modelov robota Scolopendra. Navrhnutú a vytvorenú aplikáciu ovládača bude možné spúšťať iba pod operačným systémom Windows. Ovládač bude klientskou aplikáciou v komunikácii, a tým ju bude možné hocikedy nahradiť inou podobnou aplikáciou prispôsobenou pre iný operačný systém alebo použiť takú technológiu, ktorá umožní spúšťanie grafického rozhrania pod všetkými operačnými systémami.

D. Sieťový socket

Sieťový socket je dátová štruktúra sieťového uzla – koncového zariadenia počítačovej siete. Koncovému zariadeniu slúži ako softvérový bod odosielania a prijímania dát v sieti. Štruktúru a vlastnosti socketu definuje API sieťovej architektúry. Sockety sa vytvárajú, iba ak je proces aplikácie spustený na zariadení. Termínom socket sú označované aj body komunikácie interných procesov (IPC) vrámci jedného zariadenia, ktoré často používajú rovnaké API ako sieťové sockety.[5]

Sockety sa podľa prednastaveného transportného protokolu rozdeľujú na:

1) datagramové sockety,
2) streamové sockety.

Serverové aplikácie virtuálnych modelov spolu s grafickým rozhraním budú spúšťané na jednom zariadení a serverová aplikácia fyzického modelu robota bude spustená priamo na robotovi. Zároveň by bolo vhodné, aby aplikácia grafického rozhrania informovala užívateľa o tom aká aplikácia je práve spustená. Preto v realizácii komunikačného rozhrania medzi aplikáciou grafického rozhrania a serverovými aplikáciami budú implementované práve TCP sockety. Výhodou použitia socketov narozdiel od technológií vyšších internetových protokolov je možnosť spustenia v neblokovanom režime (režim kedy aplikácia neblokuje hlavný cyklus kvôli čakaniu na prichádzajúci paket), rýchlejší prenos dát a možnosť definovať si štruktúru prenášaných dát podľa potrieb riešenia.

II. NÁVRH KNIŽNICE CORE

Pre prácu so softvérovými balíkmi kĺbových robotov bola vytvorená knižnica *Core* so zdrojovými kódmi napísanými v jazyku Python.

A. Modul Communication.py

Prvý modul *Communication.py* je zameraný na realizáciu komunikácie medzi dvoma spustenými aplikáciami prostredníctvom rozhrania socket využívajúceho TCP protokol transportnej vrstvy. Streamové sockety boli zvolené kvôli ich spoľahlivému transportu paketov, schopnosti vzájomnej notifikácie aplikácií pri ukončení prepojenia a architektúre klient-server. Všetky dáta prenášané medzi klientskými a serverovými aplikáciami sú v štandardnom formáte JSON. Sockety sú komponentom dvoch tried vystupujúcich v komunikácii:

1) *JsonServer* – server,
2) *JsonClient* – klient.

B. Modul Kinematics.py

Druhým modulom je *Kinematics.py*, ktorý je optimalizovanou verziou modulu kinematiky z bakalárskej práce. Štruktúra kinematického modelu robota je rovnako ako v module bakalárskej práce načítaná prostredníctvom súboru formátu XML, avšak s menšími zmenami názvov značiek a ich atribútov. Optimalizácia kódu bola nevyhnutná z dôvodu pomalého výpočtu inverznej kinematiky. Pomalý výpočet by znižoval obnovovaciu frekvenciu vizualizácie a simulácie a znemožňoval použitie v reálnom čase na zariadeniach s menej výkonným hardvérom. [6]

C. Modul Visualization.py

Tretí modul je zameraný na vizualizáciu modelu robota. Využíva prvky knižnice Open3D, ktorá umožňuje načítavať súbory formátov 3D objektov, vykresliť ich do okna vizualizácie a manipulovať s nimi. Konštrukcia kĺbového robota sa skladá z veľkého množstva 3D objektov vzájomne viazaných kĺbmi alebo pevnými spojmi, ktoré sú definované štruktúrou jeho kinematického modelu. Túto funkcionalitu však knižnica Open3D nemá zahrnutú.

D. Modul Package.py

Modul *Package.py* je jadrom celej knižnice, ktorý obsahuje iba jednu triedu *Package*. Funkciou inštancie triedy *Package* je generovanie dátových štruktúr z modulov knižnice *Core* inicializovaných obsahom súborov softvérového balíka. Týmto spôsobom sú všetky dátové štruktúry knižnice spravované z jedného centrálneho miesta. Zníži sa tak množstvo importovaných modulov v zdrojovom kóde vytváranej aplikácie, ktorá bude inštanciu implementovať.

III. Návrh softvérového balíka Scolopendra

Prostredníctvom knižnice *Core* bol vytvorený softvérový balík pre robota Scolopendra. Rozsiahle programové projekty si vyžadujú vytvorenie usporiadanej štruktúry adresárov a súborov. Softvérový balík je skupina na seba nadväzujúcich súborov a programov, ktoré majú spoločný logický súvis. Pokiaľ je balík určený pre vývoj alebo individuálne prispôsobenie, môže navyše obsahovať zdrojové kódy, vzorové súbory, spustiteľné aplikácie, dokumentáciu k zdrojovým kódom alebo fungovaniu súčastí balíka a manuály k ovládaniu aplikácií. Všetky komponenty softvérového balíka môžu byť určené na výrazne odlišné veci, ale pri spúšťaní alebo kompilovaní aplikácie sú komponenty spojené do jedného funkčného celku. [7]

Balík nemusí obsahovať úplne všetky zdroje potrebné na spustenie jeho obsahu, pokiaľ existuje spôsob, ako nezahrnuté zdroje externe dodať používateľovi, poprípade sú súčasťou operačného systému (externé knižnice, externé programy, kompilátory, interpretery atď.). Tým je zabezpečená kompaktnosť – nižšia náročnosť na úložný priestor a rýchlejšia distribúcia balíka zo zariadenia na zariadenie. Na druhú stranu programátor by mal pri publikácii balíka dodať potrebnú dokumentáciu, ktorá obsahuje zoznam všetkých použitých externých zdrojov a ich verzií pri spúšťaní jeho obsahu. [8]

IV. Kombinácie prepájateľných aplikácií

Spustené aplikácie softvérového balíka robota Scolopendra je možné prepájať v rôznych kombináciách podľa potrieb užívateľa alebo podľa výkonu hardvéru. Každá aplikácia ľubovoľného modelu, či už virtuálneho alebo fyzického robota Scolopendra, má svoj vlastný nezávislý výpočet kinematického modelu a trajektórií. Rôznym prepojením vytvorených aplikácií softvérového balíka robota Scolopendra je možné dosiahnuť 10 možných kombinácií prepojení:

1) Scolopendra UI a vizualizácia Open3D bez multicastu,
2) Scolopendra UI a simulácia Webots bez multicastu,
3) Scolopendra UI a fyzický robot bez multicastu,
4) Scolopendra UI, vizualizácia Open3D a multicast.

5) Scolopendra UI, simulácia Webots a multicast.
6) Scolopendra UI, fyzický robot a multicast.
7) Scolopendra UI, vizualizácia Open3D, simulácia Webots a multicast.
8) Scolopendra UI, vizualizácia Open3D, fyzický robot a multicast.
9) Scolopendra UI, simulácia Webots, fyzický robot a multicast.
10) Scolopendra UI, vizualizácia Open3D, simulácia Webots, fyzický robot a multicast.

A. *Scolopendra UI a vizualizácia Open3D*

Prvá kombinácia prepojenia je medzi klientskou aplikáciou grafického užívateľského rozhrania a serverovou aplikáciou Open3D vizualizácie virtuálneho modelu robota Scolopendra. Vizualizácia je spustiteľná iba na operačnom systéme Windows. Obrázok (1) napravo zobrazuje stav prepojenia týchto dvoch aplikácií.

B. *Scolopendra UI a simulácia Webots*

Druhá kombinácia prepojenia je medzi klientskou aplikáciou grafického užívateľského rozhrania a serverovou aplikáciou virtuálneho modelu robota Scolopendra v prostredí simulátora Webots. Obrázok (1) v strede zobrazuje stav prepojenia týchto dvoch aplikácií.

C. *Scolopendra UI a fyzický robot*

Tretia kombinácia prepojenia je medzi klientskou aplikáciou grafického užívateľského rozhrania a serverovou aplikáciou fyzického modelu robota Scolopendra. Obrázok (1) naľavo zobrazuje stav prepojenia fyzického robota a grafického rozhrania.

Obr. 1 Pohľad na prepojené serverové aplikácie s GUI

D. *Scolopendra UI a multicast*

Zvyšných sedem kombinácií prepojenia je medzi klientskou aplikáciou grafického užívateľského rozhrania a serverovými aplikáciami prostredníctvom aplikácie multicastu. Aplikácia multicastingu sa správa ako rozdeľovač prijatého TCP paketu od klienta. Prijatý paket je rozposlaný všetkým serverom, na ktorý sa multicasting dokázal pripojiť počas inicializácie. Multicasting odosiela klientovi spätne paket od posledného pripojeného servera v poradí. Pre fungovanie tejto architektúry je nevyhnutné mať spustenú aspoň jednu serverovú aplikáciu.

V. Experiment merania výpočtového času

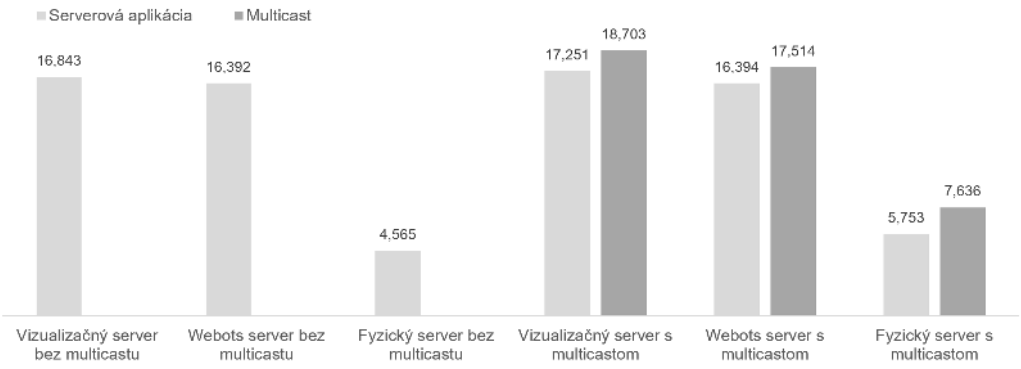
Cieľom experimentu bolo meranie času jedného výpočtu hlavnej slučky programov všetkých serverových aplikácií. Čas bol meraný pri aplikovaní translačného pohybu na model robota prostredníctvom virtuálneho joysticku z grafického rozhrania. Diaľkové ovládanie spúšťa generovanie trajektórií a ich aplikovanie na kinematický model robota. Virtuálne modely boli spúšťané na desktopovom počítači. Hardvérové parametre testovacieho desktopového zariadenia sú:

- Centrálny procesor – AMD Ryzen 7 5700X
- Grafický procesor – NVIDIA RTX 3070
- Operačná pamäť – 32GB DDR4 3600MHz

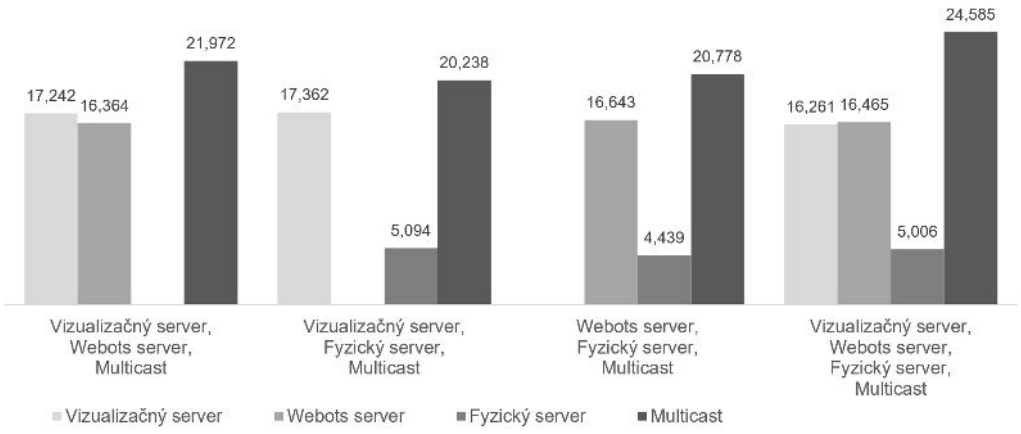
Fyzický model bol riadený jednodoskovým mikropočítačom Raspberry Pi 3B+. Riadiaca jednotka bola pripojená na počítač, ktorý mal spustenú funkcionalitu Wi-Fi prístupového bodu. Meraním času jedného výpočtu sa overovalo, či generovanie trajektórií a výpočet priamej a inverznej kinematiky nadmerne nezaťažuje výpočtové jednotky. Ďalej bol meraný aj výpočtový čas multicastu. Z nameraných hodnôt bolo možné zistiť činnosť aplikácií v reálnom čase a vplyv multicastu na výpočtový čas serverových aplikácií. Nadmerná záťaž výpočtovej jednotky by mala za následok pomalú odozvu na užívateľský vstup z grafického rozhrania a nízku obnovovaciu frekvenciu aktualizácie zobrazovaných virtuálnych modelov.

A. *Výstupy experimentu*

Výstupom experimentu bolo desať meraní, ktoré boli rozdelené na dve skupiny. Ku každému meraniu bol vytvorený stĺpcový graf (Obrázok 2 a 3) pre lepšie porovnanie nameraných hodnôt. Prvú skupinu predstavovalo šesť meraní časov jednej spustenej serverovej aplikácie pripojenej na klientskú aplikáciu bez zapnutého multicastu a so zapnutým multicastom. Zvyšné štyri merania prebiehali pri zapnutom multicaste a viacerých spustených serverových aplikáciách. Priemerný čas bol vypočítaný z 1000 nameraných vzoriek spustenej aplikácie v milisekundách.



Obr. 2 Graf časov výpočtov jednej spustenej serverovej aplikácie



Obr. 3 Graf časov výpočtov kombinácií spustených serverových aplikácií

B. *Vyhodnotenie experimentu*

Z prvej časti experimentu zameraného na jednu serverovú aplikáciu bolo zistené, že najnižší čas výpočtu má aplikácia fyzického modelu robota a najvyšší čas výpočtu má aplikácia vizualizácie. Desktopový počítač je oveľa výkonnejším zariadením ako jednodoskový mikropočítač Raspberry Pi 3B+, no jeho limitujúcim faktorom je neustále grafické vyobrazovanie virtuálneho modelu robota. Raspberry Pi vykonáva namiesto vyobrazovania modelu priame riadenie servo motorov, čo nemá až taký zásadný vplyv na výpočtovú záťaž procesora. Taktiež je možné si povšimnúť, že vizualizácia Open3D je oproti simulácii Webots výpočtovo náročnejšia. Tento fakt je daný tým, že aplikácia simulátora Webots je naprogramovaná v programovacom jazyku C++ a logika aplikácie vizualizácie s využitím knižnice Open3D je naprogramovaná v jazyku Python. Interpretovaný jazyk Python je mnohonásobne pomalší ako kompilovaný jazyk C++, a tým je aj výsledná aplikácia pomalšia. Pre všetky tri aplikácie však platí, že výpočtový čas so spusteným multicastom bol o trochu vyšší ako bez neho. Dôvodom je dvojúrovňové odosielanie paketov z klientskej aplikácie do multicastu a následne z multicastu na serverovú aplikáciu.

Z druhej časti vykonaného experimentu merania zameraného na kombinácie serverových aplikácií vyplýva, že najviac zaťažujúcou kombináciou pre desktopové zariadenie je trojkombinácia

všetkých aplikácií. Prejavilo sa to hlavne na multicaste, ktorý mal najvyšší čas výpočtu. Pri simultánnom ovládaní všetkých troch modelov z grafického rozhrania nebola registrovaná extrémne pomalá reakcia na užívateľský zásah v porovnaní s ostatnými kombináciami. Pri dostatočne výkonnom zariadení dokážu byť paralelne spustené všetky serverové aplikácie s grafickým rozhraním a reagovať na užívateľský vstup takmer okamžite. Pri menej výkonnom zariadení je možné znížiť záťaž tak, že sa zvolí dvojkombinácia alebo iba jedna spustená aplikácia.

ZÁVER

Vytvorená knižnica *Core* v jazyku Python je vhodným základom pre prácu so softvérovými balíkmi mobilný kĺbových robotov alebo kĺbových robotov s pevne viazanou základňou. Knižnica zahrňuje optimalizovaný zdrojový kód modulu kinematiky z bakalárskej práce a taktiež modul vizualizácie využívajúci prvky knižnice *Open3D*, ktorým je možné vizualizovať virtuálny model kĺbového robota zo súboru formátu XML. Vytvorené komunikačné rozhranie s využitím TCP socketov a obsahu paketu formatovaného v štandarde JSON je funkčné, spoľahlivé a rýchle. Pre robota Scolopendra bol prostredníctvom knižnice *Core* vytvorený softvérový balík s názvom *Scolopendra* obsahujúci súbory zdrojových kódov a inicializačné súbory pre rýchle programovanie aplikácií. Súčasťou balíka sú vytvorené serverové aplikácie virtuálnych modelov robota Scolopendra v prostredí vizualizácie a simulácie. Vizualizácia bola realizovaná prvkami vlastného modulu vizualizácie postavenom na dátových štruktúrach knižnice Open3D. Pre simuláciu virtuálneho modelu robota bol použitý voľne dostupný robotický simulátor Webots. K programovaniu virtuálneho modelu robota v simulácii bol vytvorený doplnkový balík, ktorý obsahuje modul ovládača virtuálnych servo motorov a ovládač robota. Balík obsahuje okrem iného aj plne funkčnú klientskú aplikáciu grafického užívateľského rozhrania s názvom Scolopendra UI. V závere práce experimentálne merané časy rôznych kombinácií architektúr vzájomného prepojenia grafického rozhrania a serverových aplikácií.

LITERATÚRA

[1] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[2] Webots, "http://www.cyberbotics.com," open-source Mobile Robot Simulation Software. [Online]. Available: http://www.cyberbotics.com

[3] L. A. MacVittie, *XAML in a Nutshell*. Ö'Reilly Media, Inc.", 2006.

[4] Microsoft, "What is XAML - WPF .NET," 2023, [online]. cit. 2023-03-04. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-7.0&redirectedfrom=MSDN&viewFallbackFrom=netdesktop-5.0

[5] L. Kalita, "Socket programming," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 4802–4807, 2014.

[6] S. Titko, "Scolopendra – návrh riadenia šesťnohej robotickej platformy," in *Technická univerzita v Košiciach, bakalárska práca*, 2021.

[7] S. Schaal, "The sl simulation and real-time control software package," Citeseer, Tech. Rep., 2009. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a729eb21ff7d95d2d247a220283c4490daf06411

[8] W. Khalil, A. Vijayalingam, B. Khomutenko, I. Mukhanov, P. Lemoine, and G. Ecorchard, "Opensymoro: An open-source software package for symbolic modelling of robots," in *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2014, pp. 1206–1211.