**УДК 004.896:37.091.12**

**Блозва Андрій Ігорович,**
*к.пед.н., доцент кафедри комп'ютерних систем і мереж*
*Національний університет біоресурсів і природокористування України*
*a-blozva@it.nubip.edu.ua*

**Касаткін Дмитро Юрійович,**
*к.пед.н., завідувач кафедри комп'ютерних систем і мереж*
*Національний університет біоресурсів і природокористування України*
*d.kasatkin@it.nubip.edu.ua*

# ПЕРСПЕКТИВИ «ROBOT OPERATING SYSTEM» У РОЗВИТКУ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

***Анотація:*** *У статті виконано огляд сучасних операційних систем, які можуть бути використані в інженерній освіті, наприклад, при розробці робочих програм дисциплін, створенні прикладних навчальних курсів, коли однією з основних проблем стає необхідність наявності недорогого лабораторного обладнання та програмного забезпечення. Розглянуті системи, що дозволяють моделювати електронні схеми і робототехнічні системи. Проведений аналіз заснований на використанні наукових праць в організованому авторами науковому студентському гуртку «Інтернет речей». Показано особливості, відмінності та недоліки роботизованих операційних систем, проаналізовано практичне їх застосування з усіма аспектами розробки робототехнічних систем без використання спеціального обладнання.*

***Ключові слова:*** *роботехнічні системи, робототехніка, операційні системи, компютерна інженерія, ROS, Robot Operating System, Player Stage Gazebo.*

Незважаючи на зростаючі потреби ринку праці в інженерах нового покоління, серед переваг абітурієнтів до сих пір спостерігається перекіс в сторону юридичних, економічних та управлінських спеціальностей. Тому одне з основних завдань сьогодні – популярізація інженерної освіти.

Сучасні інженери починають все частіше замислюватися про створення єдиної стандартної ОС для роботів, що значно прискорить появу нових розробок. Проте, для уніфікації процесу розробки потрібно вирішити ряд проблем. По-перше, практично всі сучасні роботи в корені відрізняються один від одного; часом навіть до такої міри, що простіше створити машину з нуля, ніж розбиратися в чужому середовищі, яка може в підсумку виявитися зовсім марною.

Для вирішення даної проблеми можна використовувати віртуальні лабораторії. Вони дозволяють емулювати виконання практичних вправ з побудови електричних схем,

програмування мікроконтролерів і навіть створення і програмування роботів з високим ступенем достовірності.

У цій статті робиться огляд сучасних інструментальних засобів, що надають можливості для навчання, і розглядається їх застосування в інженерному конструюванні. Даний огляд заснований на досвіді, отриманому авторами при розробці робочої програми та організації навчання студентів ОС Бакалавр спеціальності «Комп'ютерна інженерія» у науковому гуртку «Інтернет речей» на факультеті інформаційних технологій Національного університету біотехнологій та природокористування України.

Що ж таке Robot Operating System (далі ROS)? ROS - це операційна система з відкритим вихідним кодом для роботів, що забезпечує стандартні служби операційної системи:

- апаратну абстракцію,
- низькорівневий контроль пристроїв,
- реалізацію часто використовуваних функцій,
- передачу повідомлень між процесами,
- управління пакетами.

ROS, також надає інструменти і бібліотеки для отримання, побудови, написання і виконання коду на декількох комп'ютерах. По-друге, в порівнянні з комп'ютерами у роботів куди більш широке коло дотику з зовнішнім світом. Якщо майже всі комп'ютерні моделі мають, за великим рахунком один принцип роботи, то в сфері робототехніки досі немає єдиної чіткої абстракції щодо взаємодії із зовнішнім світом.

Robot Operating System - це широко використовуваний в робототехніці фреймворк, філософією якого є створення програмного забезпечення, що б працювало з різними роботами, лише з невеликими змінами в коді. Ця ідея дозволяє створювати функціональність, яка може бути перенесена без особливих зусиль для використання різними роботами, щоб раз за разом не винаходити колесо.

Проте, певні кроки в потрібному напрямку вже робляться. Першим прототипом такої системи для роботів є продукт ROS, що розробляється і в той же час активно використовується фахівцями Стенфордського університету, Масачусетського технологічного інституту, а також Технічного університету Мюнхена. Система включає в себе програмні команди для управління основними функціями машини - без необхідності розбиратися в сукупності низькорівневих методів. Подібно настільної ОС, ROS при інсталяції проводить установку всіх компонентів робота, в тому числі кінцівок і сенсорів.

Подібний проект розробляє зараз і корпорація Microsoft, ще в 2007 році випустила Robotics Developer Studio. В ідеалі будь-який власник робота в перспективі отримає можливість вийти в Інтернет і завантажити потрібну програму для робота.

ROS була розроблена в 2007 році в лабораторії штучного інтелекту Стенфорда (Stanford Artificial Intellegence Laboratory, SAIL) для підтримки стенфордського проекту AI Robot. С 2008 року, розробка продовжилася в основному в дослідницькому інституті Willow Garage, які співпрацюють з більш ніж двадцятьма різними інститутами в рамках моделі спільного розвитку.

Безліч дослідницьких інститутів почали розробляти власні проекти в ROS, додаючи підтримку свого апаратного забезпечення і ділячись прикладами власного програмного коду. Деякі компанії, що виробляють роботів, стали адаптувати свої продукти для їх використання з ROS.

Як правило, для всіх підтримуваних платформ публікується велика кількість прикладів і симулятори, що полегшують розробку власних проектів.

У ROS також підтримується безліч різних датчиків і виконавчих пристроїв (актуаторов), використовуваних в робототехніці. Щодня з'являються нові пристрої, сумісні з цим фреймворком.

У ROS є стандартні можливості операційної системи, такі як, апаратна абстракція, управління пристроями на низькому рівні, реалізована часто використовувана функціональність, передача повідомлень між процесами, і управління бібліотеками. Архітектура ROS заснована не графі з централізованої топологією. Обробка відбувається в вузлах, які можуть приймати або відправляти дані з датчиків, систем контролю стану і планування, приводів, і так далі. Бібліотека орієнтована на Unix-подібні системи (під Ubuntu Linux працює відмінно, а Fedora і Mac OS X мають статус експериментальних).

Пакет *** -ros-pkg** є загальним репозиторієм для розробки високорівневих бібліотек. Багато з можливостей часто асоційовані з ROS, такі як бібліотеки навігації і визуализатор **rviz**, зберігаються в цьому репозиторії. Ці бібліотеки надають потужний набір інструментів (різні візуалізатори, симулятори, засоби налагодження) для спрощення роботи.

Різні інструменти ROS дозволяють спостерігати за тим, що відбувається на кожному кроці.

ROS поширюється на умовах ліцензії BSD і є ПЗ з відкритим вихідним кодом. ROS безкоштовна для дослідницьких і комерційних цілей. ROS сприяє повторному використанню коду, тому розробникам робототехніки і вченим немає потреби винаходити колесо постійно. Можна отримати код зі сховищ, змінити його і знову поділитися поліпшеним ПО. Також можна написати драйвер власного датчика для ROS.

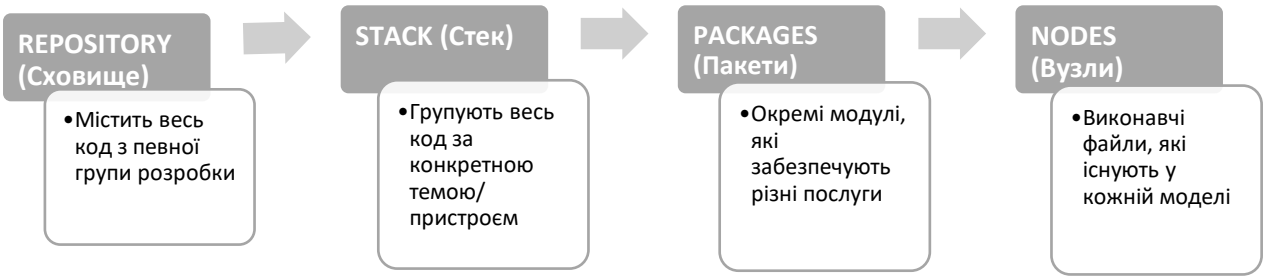


Рис. 1 – Ієрархія коду Robot Operating System

ROS підтримує паралельні обчислення, має хорошу інтеграцію з популярними C ++ бібліотеками, такими як як OpenCV, Qt, Point Cloud Library та ін., і вона може працювати на одноплатних комп'ютерах, таких як Raspberry Pi або BeagleBone Black, а також з Мікроконтроллерні платформами, наприклад , Arduino. Все це дозволяє створити свого власного робота на основі **Arduino** або **Raspberry Pi** і використовувати для його управління **Robot Operating System**.

Час є дуже важливим ресурсом і ROS є одним з інструментів, які можуть допомогти розробити комплексне рішення в найкоротші терміни і з мінімальними зусиллями. Бібліотека ROS допоможе реалізувати алгоритми і скоротити час, необхідний для інтеграції цілої купи компонентів.

ROS заснований на архітектурі графів, де обробка даних відбувається в вузлах, які можуть отримувати і передавати повідомлення між собою.

У певному сенсі, ROS схожий на «фреймворк для роботів», на кшталт: Player, YARP, Orocos,CARMEN, Orca, MOOS, Microsoft Robotics Studio.

При роботі ROS, будується «граф» - мережа точка-точка (peer-to-peer network) з процесів, які зв'язуються один з одним через інфраструктуру ROS.

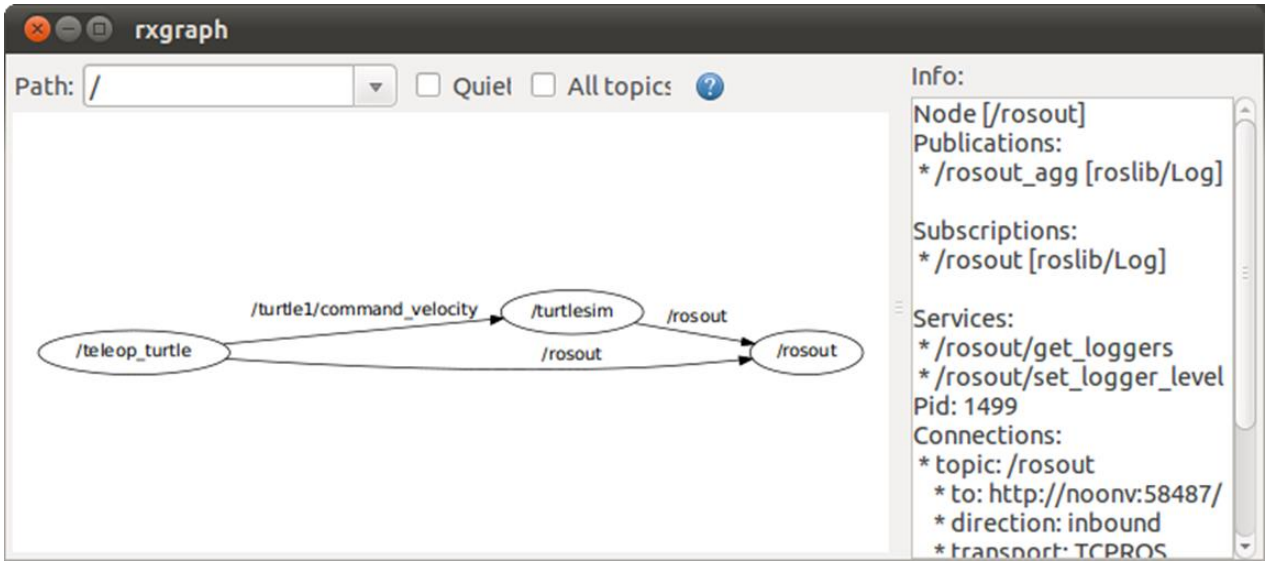


Рис. 2 – Приклад роботи ROS за побудованим «графом»

ROS реалізує кілька різних стилів спілкування:

1) синхронне (в стилі RPC) спілкування сервісів,
2) асинхронні потоки даних через Теми (Topic),
3) зберігання даних на Сервері Параметрів (Parameter Server).

Рис. 3 - Приготування кави за допомогою робота PR2 від Willow Garage.

ROS не є системою реального часу, хоча його можливо інтегрувати з кодом реального часу. Робот **PR2** від Willow Garage (дивись рис.3), використовує систему, звану

*pr2_etherCAT*, яка транслює повідомлення ROS в і з realtime-процесу. Також, ROS має повну інтеграцію з Orocos Real-time Toolkit.

Розглянемо які основні завдання та супровідні цілі переслідує впровадження ROS.

Основним завданням ROS є підтримка повторного використання коду в робототехнічних дослідженнях і розробці. ROS являє собою розподілену мережу процесів (вузлів - Nodes), що дозволяє розробляти їх індивідуально. Ці процеси можуть бути згруповані в Пакети та Стеки (Packages and Stacks), які можна легко поширювати (рис. 4.).

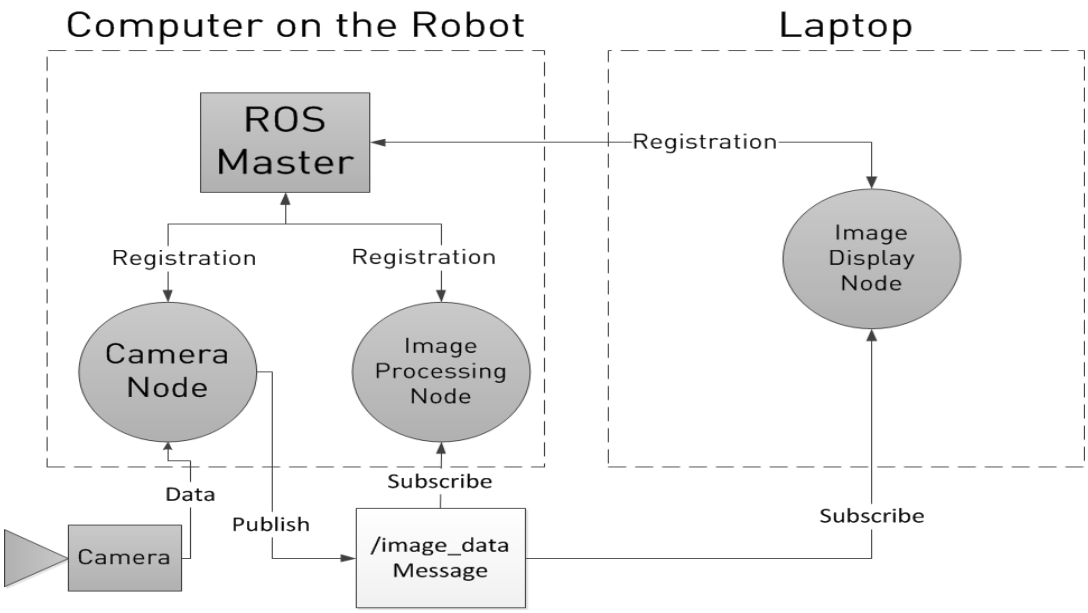


Рис. 4 – Приклад створення розподіленої мережі процесів (ROS Nodes)

Також, ROS підтримує розподілені системи репозиторіїв коду. Такий дизайн, від рівня файлової системи на рівень спільнот, дає можливість незалежно приймати рішення про розробку та реалізацію, але всі вони можуть бути об'єднані разом, засобами інфраструктури ROS.

Існує кілька цілей в рамках ROS:

1. Тонка система: ROS повинен бути якомога тонше - щоб код, написаний для ROS міг бути використаний і в інших програмних системах. Наслідком цього є те, що ROS легко інтегрується з іншими системами програмного забезпечення роботів: ROS вже інтегрований з OpenRAVE, Orocos і Player.
2. Бібліотеки: кращою моделлю розвитку є написання бібліотек з чистим функціональним інтерфейсом.
3. Незалежність від мови програмування: структуру ROS легко реалізувати на будь-якому сучасному мові програмування. ROS вже реалізований на: Python, C ++, Lisp і експериментальні бібліотеки на Java і Lua.
4. Просте тестування: ROS має вбудований фреймворк для тестування – rostest, що дозволяє легко тестувати програми.
5. Масштабування: ROS підходить для великих систем виконання і для великого процесу розробки.

Наводемо слова Brian Gerkey, учасника проектів Player і ROS, який врахував специфіку відмінностей між ROS і Player (включаючи інтеграцію OpenCV):

- «Відповідь, як завжди, залежить від багатьох чинників. Зокрема, це залежить від того, що ви намагаєтеся зробити. Player відмінно підходить для простих мобільних платформ. Він був розроблений, щоб забезпечити легкий доступ до датчиків і двигунів роботів Pioneer, обладнаних лазерним далекоміром. З іншого боку, ROS, розробляється для комплексу мобільних платформ; з маніпуляторами і всілякими датчиками. У порівнянні з Player, ROS легше використовувати в розподіленої обчислювальної середовищі, і я б сказав, що Високорівнева сторона більш розвинена в ROS, ніж в Player. У той час, як Player пропонує більше драйверів обладнання, ROS пропонує більше реалізацій алгоритмів. Я думаю, що було б справедливо сказати, що ROS є більш потужною і гнучкою системою, ніж Player, але, як завжди, велика потужність і гнучкість досягається за рахунок більшої складності. Хоча ми працюємо над тим, щоб ROS був простий у використанні, є ще значна крива навчання. Звичайно, знайомство з Player повинні допомогти у вивченні використання ROS, так як багато основні концепції схожі. Що стосується вашого конкретного питання щодо інтеграції OpenCV, я думаю, що ви знайдете більше ROS-коду, ніж Player-коду, який використовує OpenCV цікавими способами. В майбутньому, ви повинні очікувати побачити навіть більше, так як існує значна перекриття між командами-розробниками ROS і OpenCV. Мушу зазначити, що ROS використовує багато коду з проекту Player. Є вузли ROS, в яких використовується код з драйверів Player, і обидва Stage і Gazebo добре підтримується і широко використовується в ROS-співтоваристві.»

Щоб зрозуміти Браяна Герклі, необхідно розглянути особливості систему Player.

Проект Player (раніше проект Player / Stage або проект Player / Stage / Gazebo) являє собою проект зі створення вільного програмного забезпечення для дослідження робототехніки і сенсорних систем.

Проект складається з 3 основних компонентів: сервера Player і платформ для симуляції роботів - Stage (двовимірний симулятор) і Gazebo (тривимірний симулятор).

Ймовірно, Player – інтерфейс, найбільш часто використовується в робототехнічних дослідженнях. Більшість провідних журналів з робототехніки, регулярно публікують статті про застосування Player (Stage і Gazebo) для управління і моделювання робототехнічних експериментів (в академічних, урядових та індустріальних лабораторіях).

Player, забезпечує мережевий інтерфейс для різних роботів і сенсорного обладнання. Клієнт-серверна модель Player-а дозволяє програмам управління робота бути написаними на будь-якій мові програмування і працювати на будь-якому комп'ютері, підключеного до мережі разом з роботом.

Компоненти проекту працюють на POSIX-сумісних операційних системах, включаючи Linux, Mac OS X, Solaris і BSD; планується інтегрувати із платформою Microsoft Windows.

Проект був заснований в 2000 році Brian Gerkey, Richard Vaughan та Andrew Howard в Університеті Південної Каліфорнії в Лос-Анджелесі і широко використовується в робототехніці дослідженнях і навчанні.

Програмне забезпечення поширюється під ліцензією GNU General Public License з документацією під GNU Free Documentation License.

Особливості проекту:

–незалежність від робототехнической платформи;

–підтримка декількох мов програмування, включаючи C, C ++, Java, Tcl і Python;

–мінімальний і гнучкий дизайн;

–підтримка декількох пристроїв в одному інтерфейсі;

–конфігурація сервера «на льоту».

Player працює шляхом створення декількох рівнів абстракції. Він приховує низькорівневі апаратно-залежні реалізації за набором встановлених «інтерфейсів». Високорівневий контроль (логіка) взаємодіє тільки з цими інтерфейсами, що означає, що програма управління може не враховувати що ховається, за цими інтерфейсами.

Компанія Willow Garage, яку ми згадували вище, використовує проект Player для розробки програмного стека для робота PR2. Всі три компоненти проекту - Player, Stage і Gazebo були адаптовані для використання всередині ROS з метою повної інтеграції обох платформ.

Розглянувши існуючі рішення ПО для роботів, можна зробити наступні узагальнення:

–робософт працює в вигляді в проміжного шару між звичайною ОС і програмами / скриптами управління роботом

–робософт має модульну структуру, яка працює поверх базової прошарку (фреймворка)

–робософт має розподілену клієнт-серверну структуру

«Стандартний» робософт - це:

–Фреймворк, що забезпечує типові операції, шар абстракції від устаткування, міжпотокову взаємодію та інше;

–Додаткові модулі, що реалізують різні роботехнічні алгоритми (комп'ютерний зір, SLAM, машинне навчання, кінематика тощо);

–Симулятор (симулятор є важливою частиною програмного забезпечення для розробки роботів – він дозволить налагоджувати алгоритми в віртуальному середовищі, без покупки дорогого устаткування).

–Оболонка для візуального програмування і управління роботом (наочне і доступне предметно-орієнтоване програмування, уявлення стану робота у вигляді ієрархічного кінцевого автомата тощо).

Отже, з перерахованих систем, ми б виділили для подальшого впровадження у навчальний процес підготовки майбутніх комп’ютерних інженерів систему ROS.

Основним недоліком ROS є той факт, що необхідний комплект програмного забезпечення може бути встановлений тільки в операційній системі Ubuntu певних версій. Багато джерел в інтернеті, присвячені цій системі, розглядають саме процес її установки і вирішення виникаючих при цьому проблем сумісності ОС і компонентів. Таке рішення однозначно не може бути рекомендовано для використання в рамках онлайн курсів для студентів, так як більша частина потенційної аудиторії не зможе її встановити.

Висновки. Сукупність легкості установки під всі поширені в даний момент операційні системи і безкоштовна освітня ліцензія роблять ROS єдиною доступною в даний час системою, придатної для використання в якості реальної робототехнічної лабораторії для студентів. Середа ROS може бути рекомендована для використання в освітніх програмах вузів. Наявність відкритих вихідних кодів дозволяє застосовувати цю систему для вивчення і розробки алгоритмів машинного зору, зворотної кінематики, локалізації та картографування. Система ROS активно використовується в сучасних наукових дослідженнях. Однак її використання може створити проблеми з точки зору системного адміністрування, аж до установки строго фіксованих версій операційних систем.

В подальших дослідженнях ми націлені на розробки для *«Індустрія 4.0 в Україні»*, які стануть основою четвертої промислової революції. Настає епоха великих даних,

Інтернету речей, спілкування машин між собою, автоматизації, коли роботи будуть працювати в тому ж просторі, що й людина, взаємодіяти з ним. Що таке четверта промислова революція? Що таке революція? Це коли повністю змінюється існуючий лад. За прогнозами експертів, *Індустрія 4.0* змінить системи виробництва, при яких речі будуть спілкуватися між собою, а всі процеси синхронізуватися. Наприклад, системи «розумного» будинку самі будуть сповіщати інші системи на різних заводах про те, що та чи інша техніка вийшла з ладу і їй потрібна заміна. Природно, щоб такі системи функціонували, потрібні будуть не тільки програми, але і просунуті роботи, які зможуть спілкуватися як автономно, так і у взаємодії з людиною. При цьому людина зможе керувати виробництвами дистанційно.

**Список використаних джерел**


1. Антонов А.С. Что такое ROS? [Електронний ресурс]: Журнал BOTEON. Режим доступу: http://robotosha.ru/robotics/what-is-ros.html (назва з екрану).
2. Белый А.П. Введение в программирование в Операционной Системе для Роботов (ROS), (22 декабря 2015) [Електронний ресурс]: Блог Роботоша. Режим доступу: http://boteon.com/blogs/robotolik/vvedenie-v-programmirovanie-v-operacionnoi-sisteme-dlja-robotov-ros.html (назва з екрану)
3. Блозва А.І., Касаткін Д.Ю., Інформаційно-комунікаційні технології у формуванні технічної інформованості майбутніх фахівців-інженерів. Монографія. / А.І. Блозва, Д.Ю. Касаткін // -К.: ЦП Компрінт, 2015р. - 353 с.
4. Касаткін Д.Ю., Блозва А.І., Касаткіна О.М. Практикум з інформатики. Навчальний посібник. / Д.Ю. Касаткін, А.І. Блозва, О.М. Касаткіна // -К.: НУБіП України, 2016р. - 391 с.
5. Рэй Курцвейл, Восстание роботов или наше светлое будущее, Електронний ресурс: Журнал Robohunter. Режим доступу: https://robo-hunter.com/news/vosstanie-robotov-ili-nashe-svetloe-budushee (назва з екрану).
6. Autodesk CIRCUITS [Electronic resource] // Circuits.io: [site]. – San Rafael, 2017. – URL: https://circuits.io, free. – Tit. screen (usage date: 13.02.2017).
7. Brian P. Gerkey and Kurt Konolige. «Planning and Control in Unstructured Terrain». Discussion of the Trajectory Rollout algorithm in use on the LAGR robot. (2008)
8. Cyberbotics' Robot Curriculum [Electronic resource] // Wikibooks: site. – San Francisco, 2003–2016. – URL: https://en.wikibooks.org/wiki/Cyberbotics%27_Robot_Curriculum, free. – Tit. screen (usage date: 15.02.2017).
9. GAZEBO [Electronic resource]: [site]. – [Mountain View, CA, 2011–2017]. – URL: http:// gazebosim.org, free. – Tit. screen (usage date: 13.02.2017).
10. Robot Operating System: ROS [Electronic resource]: [site]. – [Kirkland, 1998–2017]. – URL: http://www.ros.org, free. – Tit. screen (usage date: 13.02.2017).
11. Teruko YATA, Akihisa OHYA, Shinichi YUTA. A Fast and Accurate SonarRing Sensor for a Mobile Robot. Proceedings of the 1999 IEEE International Conference on Robotics and Automation, May, 1999, pp. 630-636.

    V-REP: Virtual Robot Experimentation Platform [Electronic resource] // Coppelia Robotics: [site]. – Olst, 2012–2017. – URL: http://www.coppeliarobotics.com, free. – Tit. Screen (usage date: 15.02.2017).
12. Webots [Electronic resource]: [site]. – [Lausanne]: Cyberbotics Ltd, cop. 2017. – .URL: http://www.cyberbotics.com/webots.php, , free. – Tit. screen (usage date: 15.02.2017).

**Blozva Andrey**
*Candidate of Pedagogical Sciences, Associate Professor of the Department of Computer Systems and Network*
*National University of Life and Environmental Sciences of Ukraine*
*a-blozva@it.nubip.edu.ua*

**Kasatkin Dmitriy**
*Ph.D., Head of the Department of Computer Systems and Networks.*
*National University of Life and Environmental Sciences of Ukraine*
*d.kasatkin@it.nubip.edu.ua*

## PROSPECTS OF "ROBOT OPERATING SYSTEM" IN DEVELOPMENT OF COMPUTER ENGINEERING

***Abstract:*** *The article reviews the modern operating systems that can be used in engineering education, for example, when developing work programs of disciplines, creating applied training courses, when one of the main problems becomes the need for inexpensive laboratory equipment and software. The systems allowing to model electronic circuits and robotic systems are considered. The analysis is based on the use of scientific papers in the scientific student circle organized by the authors "Internet of Things". Features, differences and shortcomings of robotic operating systems are shown, their practical application is analyzed with all aspects of the development of robotic systems without the use of special equipment.*

***Key words:*** *robotic systems, robotics, operating systems, computer engineering, ROS, Robot Operating System, Player Stage Gazebo.*