№ 6(6) 2022
НАУКА і ТЕХНІКА СЬОГОДНІ
серії: право, економіка, педагогіка, техніка, фізико-математичні науки

УДК: 62:374:004.231.3

https://doi.org/10.52058/2786-6025-2022-6(6)-175-187

Кривонос Олександр Миколайович кандидат педагогічних наук, доцент кафедри комп'ютерних наук та інформаційних технологій, Житомирський державний університет імені Івана Франка, В. Бердичівська, 40, м. Житомир, 10008, тел.: (0412) 43-14-17, https://orcid.org/0000–0002–4211–6541

Жуковський Сергій Станіславович кандидат педагогічних наук, доцент кафедри комп'ютерних наук та інформаційних технологій, Житомирський державний університет імені Івана Франка, В. Бердичівська, 40, м. Житомир, 10008, тел.: (0412) 43-14-17, https://orcid.org/0000–0001–5826–0751

Кривонос Мирослава Петрівна асистент кафедри комп'ютерних наук та інформаційних технологій, Житомирський державний університет імені Івана Франка, В. Бердичівська, 40, м. Житомир, 10008, тел.: (0412) 43-14-17, https://orcid.org/0000–0001–7563–2692

ПОРІВНЯННЯ СЕРЕДОВИЩ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ РОБОТІВ ДЛЯ ПОТРЕБ НАВЧАЛЬНОГО ПРОЦЕСУ

Анотація. У статті здійснено огляд роботизованих конструкторів, що використовуються в шкільній освіті, проаналізовано сучасні середовища програмування роботів LabVIEW, NXT–G, Robolab, EV3–G, Microsoft Robotics Developer Studio, SCRATCH, Blockly, AppInventor, 12Blocks, Open Roberta. Критерії порівняння середовищ програмування були обрані виходячи з досвіду спілкування зі шкільними вчителями та викладачами вузів, які викладають робототехніку та інформатику, а також особистого педагогічного досвіду. Критерії вибиралися так, щоб, де це можливо, оцінка середовища була об'єктивною і легко перевірялася. Критерії розбиті на кілька груп для зручності представлення результатів та порівняння середовищ. Пера група критеріїв, пов'язаних із візуальною силою засобів програмування – підтримка запису математичних виразів у «натуральній» формі; модель обчислень (потік даних/потік управління); підтримка звичайних алгоритмічних конструкцій. Друга група критеріїв, пов'язаних із наявністю інструментальних засобів – засоби дистанційного керування роботом при інтерпретації програми на комп'ютері; засоба автономного виконання програми; можливість візуального представлення генерування придатного для перегляду коду текстовою мовою; можливість виконання програми на симуляторі; засобив налагодження та контролю стану програми, що виконується. Третя група критеріїв, пов'язаних із

специфікою використання середовища в освіті – простота, сучасність та візуальна привабливість інтерфейсу; наявність методичних посібників; наявність вбудованих засобів перевірки завдань; вартість. Остання група критеріїв, пов'язаних із технологічними аспектами середовища – кросплатформенність; підтримка популярних робототехнічних конструкторів; ліцензія; наявність підтримки та розвитку середовища.

Ключові слова: освітня робототехніка, візуальне програмування, середовища програмування.

Kryvonos Oleksandr Mykolaiovych Candidate of Pedagogical Sciences (PhD in Pedagogy), Docent Department of Computer Science and Information Technology (Zhytomyr Ivan Franko State University), 40, Velyka Berdychivska St., Zhytomyr, 10008, tel.: (0412) 43-14-17, https://orcid.org/0000–0002–4211–6541

Zhukovskyi Serhii Stanislavovych Candidate of Pedagogical Sciences (PhD in Pedagogy), Docent Department of Computer Science and Information Technology (Zhytomyr Ivan Franko State University), 40, Velyka Berdychivska St., Zhytomyr, 10008, tel.: (0412) 43-14-17, https://orcid.org/0000–0001–5826–0751

Kryvonos Myroslava Petrivna Assistent Department of Computer Science and Information Technology (Zhytomyr Ivan Franko State University), 40, Velyka Berdychivska St., Zhytomyr, 10008, tel.: (0412) 43-14-17, https://orcid.org/0000–0001–7563–2692

EDUCATIONAL ENVIRONMENTS COMPARISON FOR VISUAL ROBOT PROGRAMMING

Abstract. The article deals with the review of robot constructors, used in school education, analyzes modern robot programming environments such as LabVIEW, NXT–G, Robolab, EV3–G, Microsoft Robotics Developer Studio, SCRATCH, Blockly, AppInventor, 12Blocks, Open Roberta. The comparing criteria for the programming environments are chosen on the basis of experience with school teachers and university professors who teach robotics and computer science, as well as personal pedagogical experience. The criteria are chosen so that, where possible, the programming environments evaluation is objective and easily verifiable. The criteria are divided into several groups for easy results presentation and environments comparison. The first group criteria relate to the visual programming tools power, in particularly: support for writing mathematical expressions in "natural" form; computation model (data flow/control flow); support for conventional algorithmic constructions. The second group criteria relate to the availability of tools: the robot remote control means when interpreting the program on the computer; autonomous

execution mean of the program; visual representation and generation possibility of readable code in text language; executing possibility to code the program on a simulator; setting up and checking means of the executed program status. The third group criteria relate to the specific use of the programming environment in education.These are simplicity, modernity and interface visual attractiveness; methodological aids availability; built–in task checking tools availability and cost. The last group criteria relate to the technological aspects of the programming environment: cross–platform; support for popular robotics constructors; licensing; maintenance and development availability of this environment.

Keywords: educational robotics, visual programming, programming environments

Постановка проблеми. Серед важливих напрямів розвитку країни, що забезпечують її економічну міць і безпеку вважають інформаційні технології, нано– та біотехнології. Якщо для підтримки цих галузей достатньо ефективно організувати роботу відповідних спеціалістів, то в подальшій перспективі розвитку суспільства та підвищення загального потенціалу потрібно спиратись на систему освіти, що гармонійно поєднує технічний і гуманітарний аспекти у виборі можливих парадигм розвитку. Також потрібно враховувати те, що з розвитком комунікаційних засобів, Інтернету та використанням сучасних інформаційних технологій у досить широких колах кіноіндустрії, театрі, телебаченні, спорті та інших сферах діяльності, поділ на «фізиків і ліриків» стає суто академічним і не має реального протиставлення різноспрямованих нахилів учнів.

Віднедавна, середня школа досить динамічними темпами освоює нову науку – робототехніку. У багатьох школах вона вивчається в гуртках, інші її опановують у межах шкільної інформатики, і навіть додається до навчального плану як окремий предмет. При вивченні робототехніки відбувається розвиток інженерно–конструкторських і програмістських навичок, причому програмування ведеться дуже часто на низькому рівні. Це досить складна та незвична для викладача дисципліна, що вимагає не лише інженерних та програмістських знань, а й абсолютно нового методичного досвіду.

Мета статті – здійснити огляд роботизованих конструкторів, що використовуються в шкільній освіті та проаналізувати сучасні середовища програмування роботів LabVIEW, NXT–G, Robolab, EV3–G, Microsoft Robotics Developer Studio, SCRATCH, Blockly, AppInventor, 12Blocks, Open Roberta.

Поняття «Виконавець» давно використовується у викладанні інформатики, з метою пояснення школярам основних засад програмування. Досі одним із найпопулярніших виконавців є «черепашка» LOGO, запропонована Сеймур Пейпертом ще в 1967 році. Черепашка переміщується по екрану та малює лінії, тим самим візуалізуючи хід виконання програми [1].

Проте, виконавець, що переміщається по екрану комп'ютера, все ж таки недостатньо наочний. Сеймур Пейперт у своїх експериментах використовував механічного роботачерепашку, керованого з комп'ютера, що робило виконання програми ще наочнішим, а процес програмування захоплюючим. На той час створення подібних пристроїв було складним, тому механічна черепашка не набула широкого поширення. Сучасна електроніка дозволяє створювати недорогі пристрої управління, і це дає можливість повернутися до ідей Пейперта та розпочати масове впровадження реальних виконавців у освітній процес [1].

Виклад основного матеріалу. Сьогодні в школі використовуються різні робототехнічні конструктори. Серед тих, які набули найбільшого поширення в практиці, можна назвати наступні:

- Lego Mindstorms NXT 2.0: один з найпоширеніших у школі наборів. Крім безлічі деталей та процесорного блоку, має програмну підтримку у вигляді великої кількості середовищ програмування.
- Lego Mindstorms RCX: попередня версія набору сьогодні зустрічається набагато рідше.
- Lego Mindstorms EV3: остання (2013) версія набору. Має більш потужний процесор і зміни як у самому наборі, так і в середовищі програмування. Існують також програмно–апаратні платформи, які можуть бути використані навіть у шкільній робототехніці:
- Arduino: апаратна обчислювальна платформа. На її основі можна створювати свої роботи.
- Raspberry Pi: повноцінний комп'ютер, що вимагає установки операційної системи, і при цьому має розмір банківської картки.

Продукція компанії Lego виділяється на загальному тлі, і насамперед своєю пристосованістю до шкільних реалій. Серія Mindstorms побудована на основі педагогічної концепції Lego, яка передбачає простоту та інтуїтивність, але водночас дозволяє створювати дуже складні об'єкти із простих елементів. Тому в набір входять деталі, з більшістю з яких діти знайомі ще з раннього віку. Проблема підключення реєструючих та виконавчих пристроїв (у нашому випадку – сенсорів і двигунів) вирішена за допомогою стандартного інтерфейсу, загального для всіх систем, що підключаються. Найголовніше – не потрібна робота паяльником; в іншому випадку, час створення моделей збільшився б у кілька разів і, швидше за все, не можна було вести мови про вивчення робототехніки в початковій школі.

Робот, зібраний з конструктора, складніший у програмуванні, ніж «черепашка», оскільки конструктор дозволяє створювати довільні конструкції, які доводиться програмувати в термінах потужностей двигунів на портах, а не конкретних рухів та поворотів. Тому при використанні таких конструкторів у освітніх цілях велика увага приділяється засобам програмування. В освітній

робототехніці дуже популярні середовища, що використовують візуальні мови, оскільки візуальні мови простіші у використанні і наочніше, ніж текстові – іноді на візуальному середовищі програмувати робота можуть навіть дошкільнята, які не мають навіть базових навичок читання.

Таких середовищ зараз існує досить багато, всі вони мають свої переваги та недоліки. Кожне середовище програмування, якщо воно використовується, у чомусь перевершує аналоги і краще підходить для того завдання, яке вирішує, але задачі, що вирішуються, для всіх існуючих середовищ різні. Автори в попередніх роботах вже розглядали дидактичні можливості середовища моделювання пристроїв які використовуються в робототехніці [2], а зараз пропонується зробити певний аналіз середовищ програмування роботів з позиції доцільності використання в навчальному процесі Розглянемо деякі з них.

Середовише LabVIEW [3] – кросплатформове середовище моделювання та розробки загального призначення. Середовище є пропрієтарним, воно створене американською компанією National Instruments у 1986 році і підтримується досі. Програмування в LabVIEW здійснюється візуальною мовою програмування потоків даних G. Мова G моделює процес обчислень, орієнтованій на дані, в якому явно задається не послідовність виконання операторів, а зв'язки між блоками за даними. Блок програми може надавати деякі вихідні дані, які можуть бути вхідними для іншого блоку. Блоки починають виконуватися, коли мають дані на усіх входах. Якщо кілька блоків мають дані на вході, то вони виконуються паралельно. Такий підхід досить сильно відрізняється від того, що прийнято в імперативному програмуванні, проте він широко поширений серед інженерів і вчених. Наприклад, на аналогічних принципах засноване інше відоме візуальне середовище програмування наукових обчислень та моделювання Matlab/Simulink.

LabVIEW застосовується дослідниками та інженерами по всьому світу для автоматизації лабораторних та технологічних процесів. Середовище підтримує безліч апаратних платформ і надає десятки бібліотек програмних компонентів від складних математичних методів обробки інформації та алгоритмів комп'ютерного зору до засобів взаємодії з системами управління базами даних, створення інтерфейсів і зовнішнього вигляду приладових панелей. Зокрема, є модулі, що забезпечують підтримку LabVIEW робототехнічних конструкторів Lego Mindstorms NXT і Lego Mindstorms EV3.

Порівняно з іншими платформами, представленими в цій роботі, система LabVIEW в «оригінальному» вигляді досить складна для вивчення, проте часто використовується в освіті. Освітні експерименти із застосуванням LabVIEW у середніх школах були особливо популярні у 1990–х роках [4] (що дало початок різним освітнім адаптаціям LabVIEW, таким як Robolab), значно частіше нині середовище використовується у ЗВО, тематичних гуртках та таборах [5] .

Середовище має можливості генерації коду для автономного виконання на роботі, а також налагодження програм на комп'ютері з посиланням команд на робота. Алгоритмічні конструкції підтримані у повному вигляді, проте модель виконання є потоковою, що може бути дуже зручним для програмування багатьох алгоритмічних конструкцій, та водночас складним для вивчення та розуміння.

Інтерфейс користувача програми та створювані діаграми не виглядають сучасно (що пояснюється віком системи). Створювані у середовищі програми досить часто виходять значними за обсягом і важко сприймаються наочно, особливо якщо малюються користувачами–новачками. Система надає тисячі блоків, піктограми яких часто відрізняються дрібними деталями. Ці фактори з урахуванням вартості ліцензії середовища та її складності роблять досить рідкісними застосування LabVIEW у шкільній освіті, набагато частіше для цього використовуються різні його адаптації.

Середовище NXT–G [6] – засіб програмування, що постачається у комплекті з конструктором Lego Mindstorms NXT. NXT–G базується на системі LabVIEW, тому «успадковує» мову потоків даних G. Блоки автоматично розміщуються на діаграмі, властивості можуть бути відредаговані прямо на графічному поданні блоку. Засобами LabVIEW можливе додавання сторонніх блоків, крім того, сам NXT–G дозволяє виділити набір блоків підпрограму і використовувати її як новий блок [7].

Основна проблема NXT–G полягає у слабкій підтримці математичних виразів. Математичні формули, як і вся програма, будуються візуально, з блоків. Існують блоки читання та запису значення змінних, блок, що зчитує значення константи, блоки, що зчитують показання з сенсорів, і блоки елементарних арифметичних операцій. Таким чином, навіть щоб запрограмувати нескладну формулу, потрібно зображати блоками дерево розбору виразу, яке задає цю формулу. Серйозність цієї проблеми ілюструється програмою, що реалізує пропорційно–диференціальний регулятор для руху робота вздовж лінії або довкола перешкоди. Мовою С така програма займає близько десятка рядків, тоді як на NXT–G не міститься на одному екрані, містить безліч потокових зв'язків і дуже складна для розуміння. Таким чином, одному із запропонованих критеріїв – придатності для ілюстрації змістовного матеріалу інформатики та кібернетики – NXT–G не відповідає. Відповідно тому NXT–G і не набув широкого поширення в школах.

Середовище NXT–G спеціально створювалося для початківців, тому досить просте і зручне в роботі. На думку деяких користувачів, вона навіть занадто ергономічна, оскільки не дає довільно розміщувати блоки на діаграмі, автоматично (і не завжди вдало) прокладає з'єднувальні лінії між блоками тощо. Крім того, для застосування NXT–G у шкільних класах виявилася важливою така особливість: більшість властивостей елементів не

відображається на діаграмі, а доступна лише через редактор властивостей, що унеможливлює відображення всієї програми на проекторі.

Ніяких засобів налагодження NXT–G немає, текстова форма програми не породжується. До переваг продукту слід віднести те, що він поширюється разом з конструктором і доступний для завантаження з сайту виробника безкоштовно.

Ще одне середовище, що базується на LabVIEW — Robolab, спеціально створювалося як адаптація LabVIEW для шкільної освіти і з самого початку враховувало побажання шкільних вчителів та специфіку викладання в школах [8, 9]. Приклад специфічного для шкіл рішення, реалізованого в Robolab, – наявність двох рівнів складності мови середовища. Кожен із цих рівнів розбивається ще кілька підрівнів, тому користувачеві «відкривається» нова функціональність. На найпростішому, пілотному рівні доступні лише деякі можливості візуальної мови, і програма будується заповненням порожніх місць у шаблоні за допомогою вибору блоків спливаючого меню. Це дозволяє створювати тільки найпростіші програми, що мають стандартну структуру: команди управління моторами, за якими слідує блок, що очікує настання будь–якої події. Ідея такого поділу в тому, щоб дати можливість учням у початковій школі. На другому рівні складності користувачі можуть малювати діаграми, розміщуючи довільним чином блоки з палітри та з'єднуючи їх лініями, що визначають потік керування. Розбиття на рівні та підрівні організовано у такий спосіб, щоб учні могли освоювати середовище програмування практично без допомоги вчителя, керуючись лише інтуїцією.

На відміну від NXT–G, Robolab дозволяє описувати довільні математичні висловлювання у текстовому вигляді. Для завдання формул можуть використовуватися тригонометричні функції та звернення безпосередньо до значень показань сенсорів. Цикли в Robolab описуються за допомогою блоків "label" і "go to" – передача управління більше не візуалізується. Є умовні оператори, можливість запуску паралельних процесів, блоки управління цими процесами, і навіть засоби роботи з підпрограмами. На Robolab можна реалізувати досить складні програми, і Robolab цілком підходить для ілюстрації матеріалу з кібернетики аж до молодших курсів вузів.

За іншими критеріями Robolab показує дещо гірші характеристики. Додаток було створено в кінці 90–х років, з того часу його інтерфейс практично не змінювався, тому він виглядає несучасно, діаграми, що візуалізують потік управління, на великих розмірах важко читати. Спеціалізованих засобів налагодження в Robolab немає, хоча є можливість зняття показань пристроїв введення з робота та відображення їх на екрані комп'ютера. Можливості генерації діаграми текстове уявлення не підтримано. Robolab не безкоштовний, одну ліцензію за вартістю можна порівняти з робототехнічним набором, що для шкіл досить дорого. Розвиток Robolab йде в основному шляхом додавання

нових блоків, саме середовище давно не змінювалося. Зокрема в середовищі відсутня підтримка Lego EV3.

Незважаючи на зазначені недоліки, Robolab на даний момент є одним з найбільш широко використовуваних у школах середовищ програмування роботів. За відгуками викладачів, у багатьох є бажання відмовитися і замінити більш сучасну систему.

Середовище EV3–G – програмне забезпечення, яке постачається в комплекті з конструктором Lego Mindstorms EV3. EV3–G також створена на основі LabVIEW і дозволяє програмувати контролери NXT і EV3 мовою G. Середовище багато в чому аналогічне NXT–G: має сучасний інтерфейс користувача, набір прикладів виконання діаграм на роботі. Блоки, з яких створюється програма, автоматично «зчіплюються» один з одним (аналогічно NXT–G), мають зрозуміле та зручне для редагування графічне уявлення. У середовищі також виправлено один значний недолік NXT–G – математичні вирази можуть задаватися значно зручніше, зокрема текстом.

Тим не менш, деякі критерії EV3–G не задовольняє. У середовищі немає можливості налагодити програму на симуляторі, а також не підтримується ОС Linux. Є відомі проблеми сумісності із NXT. У робототехнічній спільноті поширена думка, що середовище погано підходить для створення великих та складних програм. Система безкоштовна для індивідуальної експлуатації, проте версія для освітніх закладів є платною.

Microsoft Robotics Developer Studio [10, 11] – розробка компанії Microsoft, призначена для програмування складних багатопотокових додатків з реактивною моделлю поведінки, що використовуються для управління робототехнічними системами. Необхідність створення таких додатків є не тільки в робототехніці, тому Robotics Developer Studio використовується і для створення відповідних додатків, що не належать. Програми в Robotics Developer Studio малюються у вигляді діаграм на мові VPL (Visual Programming Language), які візуалізують зв'язки між окремими компонентами, що виконуються паралельно (веб–сервісами), і з яких складається програма.

Слід зазначити, що у сфері шкільної освіти Microsoft Robotics Developer Studio використовується дуже рідко. Головна причина цього полягає в тому, що середовище розраховане переважно на симуляцію і не може ефективно взаємодіяти з реальним роботом. Для LEGO Mindstorms NXT є можливість керування каналом Bluetooth, але немає можливості завантаження програми на робота для автономного її виконання, оскільки на практиці немає можливості запустити віртуальну машину .NET. Реальні роботи, керовані MRDS, зазвичай набагато складніші і дорожчі за те, що можна використовувати в школах (стандартна платформа, наприклад, має у своєму складі ноутбук). Лише управління по Bluetooth недостатньо для вирішення завдань, що вимагають малого часу реакції робота, через великі затримки посилки–приймання

Bluetooth–пакетів. Це робить MRDS непридатною для великої області завдань, що вирішуються в школі. Симуляції ж, в свою чергу, теж виявляється недостатньо, бо навіть із гарною фізичною моделлю світу MRDS створює модель деякого ідеального світу, в якому великої кількості проблем, які вирішуються алгоритмами кібернетики, просто не виникає. Навіть просте завдання, яке вирішується на реальному роботі, може виявитися наочнішим і кориснішим школярам, ніж складна програма, що виконується на моделі в симуляторі

Друга важлива причина дуже вузького поширення MRDS у школах – модель обчислень, що використовується в ній. Подання програми у вигляді набору взаємопов'язаних розподілених веб–сервісів може бути зручним для досвідчених програмістів, але початківцям важко зрозуміти принципи, що лежать в основі такої моделі. Складні механізми взаємодії веб–сервісів багато в чому «заховані» за допомогою візуальної мови VPL, але все ж таки потрібне хоч мінімальне розуміння процесів, що відбуваються в системі, для того щоб малювати змістовні діаграми. Загалом можна сказати, що MRDS більше підходить для студентів чи професійних програмістів, ніж для школярів. Середовище хоч і дозволяє писати як завгодно складні розподілені програми, але робити це можна досить нетривіальним і специфічним чином, що сильно знижує її цінність як ілюстративного матеріалу.

Що стосується інших критеріїв, система досить зручна в роботі, має засоби налагодження та генерації коду, проте, відповідно до своєї специфіки, ці засоби є досить складними для використання школярами. З 2014 MRDS офіційно не підтримується компанією Microsoft.

SCRATCH І SCRATCH–подібні засоби Scratch [11] — кросплатформове візуальне середовище програмування з відкритим вихідним кодом, що розробляється в Массачусетському Технологічному Інституті для навчання школярів основ інформатики. Програмування здійснюється за допомогою з'єднання блоків, які нагадують елементи мозаїки. Scratch дозволяє намалювати та запрограмувати прості графічні об'єкти, що називаються спрайтами.

У «чистому» вигляді Scratch не дозволяє програмувати роботів, проте існує велика кількість розширень та середовищ на базі Scratch, які дозволяють програмувати роботи Lego WeDo, Lego NXT, Lego EV3 та Arduino. Серед таких «самостійних» проєктів, створених на базі Scratch, згадаємо S4A та mBlock для програмування Arduino та Enchanting для програмування NXT, а також російський проект ScratchDuino. Спільними для Scratch–подібних середовищ плюсами є легкість у вивченні, привабливий інтерфейс користувача, відкритість і безкоштовність, можливість налагодження віддаленого управління роботом з комп'ютера та завантаження коду для автономного виконання (останнє доступне не у всіх Scratch–системах). Існує також можливість виконання програми на віртуальному роботі, що може розглядатися за нашими

критеріями як налагодження на симуляторі (проте про наближеність такої симуляції до реальності не йдеться).

До негативних сторін віднесемо відсутність модернізованих засобів навчання програмування. Наприклад, відсутня можливість генерації коду, котра «читається» за візуальною моделлю, що могло б значно полегшити перехід учнів на текстові мови. Алгоритмічні аспекти підтримані в повному обсязі, наприклад, відсутня підтримка масивів розмірності більше 1. Кошти автоматичної перевірки коректності рішення завдань також відсутні. Таким чином, Scratch добре підходить для вивчення інформатики та робототехніки у молодших та середніх класах і не настільки добре – для старшого віку та «просунутих» занять.

Blockly – візуальне Scratch–подібне середовище, що розробляється компанією Google Inc. для навчання дітей програмування. Середовище Blockly саме по собі не має можливості програмування реальних роботів, а репрезентує собою модуль, що перевикористовується, який може бути вбудований в сторонні програми (у тому числі системи програмування роботів), що активно використовується розробниками по всьому світу (наприклад, в проектах AppInventor і Wonder Workshop, див. секції нижче). Інтерфейс середовища та візуальна мова практично не відрізняються від Scratch.

Існує набір віртуальних виконавців та завдань для цих виконавців, для вирішення яких використовується Blockly. Для кожного завдання перевіряється коректність його вирішення. При цьому для кожного рішення можна переглянути код мовою JavaScript, що відповідає візуальній діаграмі.

AppInventor [13] – середовище візуального програмування програм для платформи Android. AppInventor використовує ядро Blockly як редактор діаграм, а також дозволяє намалювати макет інтерфейсу Android–програми. Програма далі генерується в код і може бути запущена на Android–пристрої. Одна з особливостей AppInventor, цікава в контексті цієї статті – можливість взаємодії програмованих програм з пристроями Lego NXT за протоколом Bluetooth. Це часто використовується для програмування пультів віддаленого управління Lego–роботом.

12Blocks – ще один Scratch–подібний інструмент для програмування Lego Mindstorms NXT і Arduino–роботів, доступні й інші менш популярні платформи (наприклад Scribbler). Середовище кросплатформове, доступне для ОС Windows, Linux і Mac OS X. Є можливість виконання програм на тривимірному симуляторі Cogmation, генерації коду по діаграмі, інтеграції з ROS7. У мові підтримані основні алгоритмічні конструкції і типи даних, доступна можливість виділення коду в підпрограму. Існують також можливості автономного виконання програми роботом, налагодження програми на комп'ютері з посилкою команд на роботу, а також побудови графіків із сенсорів у реальному часі. Розробники позиціонують систему як придатну на всіх етапах

освіти – від початкової школи до закладу вищої освіти. До негативних сторін віднесемо такі: 12Blocks має слабку методичну підтримку, в Інтернеті практично немає спільнот, що обговорють важливі питання пов'язані з цим середовищем (проте є набір англомовних відео–інструкцій щодо користування основними можливостями середовища). Також у системі відсутні будь–які засоби автоматичної перевірки завдань. 12Blocks поширюється за комерційною ліцензією, проте будь–якої активності на офіційному сайті з 2014 року відсутня.

Open Roberta [13] – хмарне середовище візуального програмування роботів Lego Mindstorms EV3, що з'явилося в 2014 році. Програмування в Open Roberta здійснюється візуальною мовою NEPO, дуже схожою на мову Scratch. Це пояснюється використанням Blockly як візуального редактора. Проект безкоштовний, має відкритий вихідний код і зараз активно поширюється в країнах Європи. Важливою особливістю Open Roberta є двомірний симулятор робота, інтегрованого з середовищем. До плюсів віднесемо і методичну підтримку середовища, проте лише англомовну; проте проєкт повністю відкритий і може бути адаптований активістами. Найбільший недолік проєкту полягає в утрудненні комунікації з реальним роботом через те, що система працює в браузері. Єдиний спосіб спілкування середовища Open Roberta з контролером робота за допомогою Wi–Fi модуля, при цьому середовище надсилає Ajax–запити на віддалений сервер, а робот у цьому випадку діє як HTTP–клієнт. З інших мінусів відзначимо повну відсутність засобу генерації коду та перевірки завдань, інтерфейс не локалізований російською мовою.

З наведеного огляду освітніх візуальних середовищ програмування роботів можна зробити наступні **висновки**:

Для потреб освіти існує не така велика кількість засобів програмування роботів, більшість із існуючих з'явилися в 2010–х роках, і постійно з'являються нові, наприклад, проект Root від Гарвардського Університету, згадки про який почали з'являтися у 2016 році;

Незважаючи на нечисленність наявних рішень, серед засобів програмування роботів вже існують інструменти, якість яких, на думку авторів, перебуває на задовільному рівні. Зовсім інша ситуація з інструментами, що застосовуються для середньої освіти. Розкид досить великий – від великих і складних комерційних інструментів, таких як LabVIEW і MRDS, до серед програмування конкретних робототехнічних платформ, що надають «мінімальну» функціональність (редактор діаграм + можливість запуску на роботі).

«Просунуті» функції, такі як генерація текстового коду по діаграмі, можливість симуляції на віртуальному пристрої або вбудована автоматична перевірка обмежень, практично не зустрічаються в популярних на сьогоднішній день середовищах, а якщо такі функції і реалізовані, то не всі відразу. До того ж продукти, що пропонують таку функціональність, є комерційними, і багато

навчальних закладів просто не можуть собі дозволити їх (не кажучи про індивідуальне використання). Більше того, не існує єдиного рішення, яке б охоплювало всі популярні робототехнічні освітні платформи.

Література:

1. Papert S.. Mindstorms: Children, Computers, and Powerful Ideas. New York, NY, USA: Basic Books, Inc., 1980. p. 230.
2. Прикладне програмне забезпечення для моделювання електронних пристроїв на базі платформи arduino / О. Кривонос, Є. Кузьменко, М. Кривонос, С. Кузьменко // Інформаційні технології в освіті. - 2020. - № 43. - С. 39-51.
3. Kodosky J., MacCrisken J., Rymar G. Visual programming using structured data flow // Visual Languages, 1991., Proceedings. 1991 IEEE Workshop on / IEEE, 1991. p. 34–39.
4. A low–cost, innovative methodology for teaching engineering through experimentation / M. Cyr, V. Miragila, T. Nocera // Journal of Engineering Education. Washington, 1997. V. 86. p. 16–172.
5. A LabVIEW–based remote laboratory experiments for control engineering education / M. Stefanovic, V. Cvijetkovic, M. Matijevic // Computer Applications in Engineering Education, 2011. V. 19, № 3. p. 538–549.
6. Floyd K. J. Lego Mindstorms NXT–G Programming Guide. Apress, 2007. p. 336. Nguyen Khuong A. A case study on the usability of NXT–G programming language // Proc. of 23rd Conf. in Psychology of Programming. 2011.
7. Portsmore M. ROBOLAB: Intuitive Robotic Programming Software to Support Life Long Learning // APPLE Learning Technology Review. 1999.
8. Ben E., Martha C., Chris R. Lego engineer and robolab: Teaching engineering with labview from kindergarten to graduate school // International Journal of Engineering Education, 2000. V. 16, № 3. p. 181– 192.
9. Jared J., Microsoft robotics studio: A technical introduction // Robotics & Automation Magazine, IEEE, 2007. T. 14, № 4. p. 82–87. [in English]
10. An overview of the Microsoft Robotics Developer Studio https://acodez.in/microsoft–robotics–developer–studio/
11. Scratch: programming for all / M. Resnick, J.Maloney, A. Monroy–Hern´andez // Communications of the ACM, 2009. V. 52, № 11. p. 60–67.
12. App Inventor / D. Wolber, H. Abelson, E. Spertus. O'Reilly Media, Inc., 2011.

References:

1. Papert S.. Mindstorms: Children, Computers, and Powerful Ideas. New York, NY, USA: Basic Books, Inc., 1980. p. 230 [in English].
2. Krivonos, O., Kuz'menko, Є. , Krivonos, M. , Kuz'menko, S. Prikladne programne zabezpechennja dlja modeljuvannja elektronnih pristroïv na bazi platformi arduino [Applied software for modeling electronic devices based on the arduino platform]. *Informacijni tehnologiï v osviti - Information technologies in education*, 43, 39-51 [in Ukrainian].
3. Kodosky J., MacCrisken J., Rymar G. Visual programming using structured data flow // Visual Languages, 1991., Proceedings. 1991 IEEE Workshop on / IEEE, 1991. p. 34–39 [in English].
4. A low–cost, innovative methodology for teaching engineering through experimentation / M. Cyr, V. Miragila, T. Nocera // Journal of Engineering Education. Washington, 1997. V. 86. p. 16–172 [in English].

5. A LabVIEW–based remote laboratory experiments for control engineering education / M. Stefanovic, V. Cvijetkovic, M. Matijevic // Computer Applications in Engineering Education, 2011. V. 19, № 3. p. 538–549 [in English].

6. Floyd K. J. Lego Mindstorms NXT–G Programming Guide. Apress, 2007. p. 336. Nguyen Khuong A. A case study on the usability of NXT–G programming language // Proc. of 23rd Conf. in Psychology of Programming. 2011 [in English].

7. Portsmore M. ROBOLAB: Intuitive Robotic Programming Software to Support Life Long Learning // APPLE Learning Technology Review. 1999 [in English].

8. Ben E., Martha C., Chris R. Lego engineer and robolab: Teaching engineering with labview from kindergarten to graduate school // International Journal of Engineering Education, 2000. V. 16, № 3. p. 181– 192 [in English].

9. Jared J., Microsoft robotics studio: A technical introduction // Robotics & Automation Magazine, IEEE, 2007. T. 14, № 4. p. 82–87. [in English]

10. An overview of the Microsoft Robotics Developer Studio https://acodez.in/microsoft–robotics–developer–studio/ [in English].

11. Scratch: programming for all / M. Resnick, J.Maloney, A. Monroy–Hern´andez // Communications of the ACM, 2009. V. 52, № 11. p. 60–67 [in English].

12. App Inventor / D. Wolber, H. Abelson, E. Spertus. O'Reilly Media, Inc., 2011 [in English].