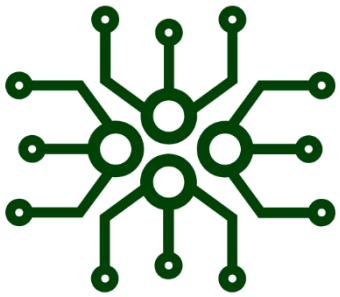
Міністерство освіти і науки України
Прикарпатський національний університет
імені Василя Стефаника

ISSN 3083-7618 eISSN 3083-7626

Information Technologies and Engineering Electronics

Інформаційні технології та інженерна електроніка

2/2025



MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

VASYL STEFANYK PRECARPATHIAN NATIONAL UNIVERSITY

INFORMATION TECHNOLOGIES AND ENGINEERING ELECTRONICS

Scientific journal

ISSN 3083-7618
eISSN 3083-7626

Founded in 2024

I S S U E 2

Ivano-Frankivsk
Vasyl Stefanyk Precarpathian
National University
2025

UDC 621.3 : 004

The scientific journal was founded by the Vasyl Stefanyk Precarpathian National University, by the decision of the Academic Council of the university, protocol No. 11 dated November 12, 2024, in the following specialties: F2 - Software Engineering; F3 - Computer Science; F7 - Computer Engineering; G5 - Electronics, Electronic Communications, Instrumentation and Radio Engineering.

Website: https://journals.pnu.edu.ua/index.php/itee/

Founder – Vasyl Stefanyk Precarpathian National University (EEDRPOU code 02125266 76018, Ivano-Frankivsk, Shevchenko St., 57 tel. (0342) 75-23-51 fax (0342) 53-15-74 e-mail: office@pnu.edu.ua).

Media identifier (National): R30-06174. Registered by the National Council of Ukraine for Television and Radio Broadcasting dated 05/29/2025 No. 1150.

The journal is published two times per year.

DOI prefix: 10.15330.

The authors are responsible for the content of the article.

Responsible for the issue: *I.T. Kogut, Dr. Sc. (Tech.), prof., I.V. Svyd, PhD, Assoc. prof.*

Recommended for publication by the editorial board of the scientific journal "Information Technologies and Engineering Electronics" on 27.06.2025.

Editorial Board address: Vasyl Stefanyk Precarpathian National University (Vasyl Stefanyk PNU), 76018, Ivano-Frankivsk, Shevchenko St., 57, office 210a, +38(0342) 59-60-07.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТЕФАНИКА

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА ІНЖЕНЕРНА ЕЛЕКТРОНІКА

Науковий журнал

ISSN 3083-7618
eISSN 3083-7626

Засновано в 2024 р.

В И П У С К 2

Івано-Франківськ
Прикарпатський національний університет
імені Василя Стефаника
2025

УДК 621.3 : 004

Науковий журнал засновано Прикарпатським національним університетом імені Василя Стефаника, рішенням Вченої ради університету протокол № 11 від 12.11.2024 року, за спеціальностями: F2 – Інженерія програмного забезпечення; F3 – Комп'ютерні науки; F7 – Комп'ютерна інженерія; G5 – Електроніка, електронні комунікації, приладобудування та радіотехніка.

Сайт: https://journals.pnu.edu.ua/index.php/itee/

Засновник – Прикарпатський національний університет імені Василя Стефаника (код ЄДРПОУ 02125266 76018, м. Івано-Франківськ, вул. Шевченка, 57 тел. (0342) 75-23-51 факс (0342) 53-15-74 e-mail: office@pnu.edu.ua).

Ідентифікатор медіа: R30-06174. Зареєстровано Національною Радою України з питань телебачення і радіомовлення від 29.05.2025 № 1150.

Журнал видається два рази на рік.

Префікс DOI: 10.15330.

За зміст статей відповідальні автори.

Відповідальні за випуск: *І.Т. Когут, д-р техн. наук, проф., І.В. Свид, канд. техн. наук, доц.*

Рекомендовано до публікації редколегією наукового журналу «Інформаційні технології та інженерна електроніка» 27.06.2025.

Адреса редколегії: Прикарпатський національний університет імені Василя Стефаника (ПНУ ім. В. Стефаника), 76018, м. Івано-Франківськ, вул. Шевченка, 57, кабінет 210а, +38(0342) 59-60-07.

CONTENT

ЗМІСТ

УДК 614.842.435 **DOI:10.15330/itee.2025.2.01**

Н. А. ОСТАПИШИН, В. І. МАНДЗЮК, док. фіз.-мат. наук

ПРОТИПОЖЕЖНА СИСТЕМА НА ОСНОВІ МОДУЛЯ NODEMCU V3 ESP8266 ДЛЯ ПОБУТОВОГО ЗАСТОСУВАННЯ

Вступ

Широке використання сучасних електронних компонентів і цифрових методів обробки інформації призводить до істотної “інтелектуалізації” технічних засобів. Цифрові системи на основі мікроконтролерів і сенсорів застосовуються для підвищення гнучкості та функціональності технічних пристроїв шляхом обробки даних у реальному часі та прийняття рішень на місці без участі центрального комп’ютера [1]. Сучасні технічні засоби охорони, до яких насамперед можна віднести системи електронних пожежних сигналізацій, накопичують та інтерпретують дані від широкого спектра сенсорів для раннього виявлення загроз [2]. Дані системи можуть бути застосовані як повністю інтегровані системи або системи, в основі яких лежать функціонально незалежні компоненти, з’єднані через цифрові шини та мережі [3]. Сучасні тенденції розвитку електротехніки та електронної техніки, в основі якої лежить мікромініатюризація, вимагають використання широкої номенклатури малопотужних і малогабаритних пристроїв та виробів, що виконані на новій конструктивній основі електрорадіоелементів, оскільки мікросхеми з високим ступенем інтеграції дозволяють значно зменшити масогабаритні характеристики складних пристроїв при одночасному підвищенні енергоефективності та швидкодії [4, 5]. На сучасному етапі розвитку досягнення науки і техніки у сфері електронної техніки дають можливість істотно зменшити масогабаритні характеристики електронних пристроїв за рахунок використання однокристальних мікроконтролерів і водночас поліпшити їх показники надійності і довговічності та якісні характеристики, що підтверджено сучасними розробками інтегрованих систем обробки сигналів і вбудованих алгоритмів розпізнавання [6].

Актуальність розробки портативних недорогих систем пожежної сигналізації на основі мікроконтролерів для побутового застосування зумовлена зростаючою кількістю пожеж у житловому секторі та необхідністю забезпечення раннього виявлення загорянь з мінімальними витратами. Пожежі в житлових приміщеннях залишаються однією з вагомих причин травматизму та смертності, що підкреслює соціальну значущість впровадження доступних автономних систем детекції [1]. Сучасні дослідження у сфері сенсорних технологій доводять, що інтеграція датчиків диму, температури та газів із мікроконтролерною обробкою сигналів дозволяє підвищити точність виявлення пожежі та зменшити кількість хибних спрацьовувань [7]. Розвиток мікромініатюризації та технологій корпусування мікросхем забезпечує зменшення масогабаритних характеристик і підвищення надійності електронних пристроїв при зниженні їх вартості [8]. Використання сучасних однокристальних мікроконтролерів із низьким енергоспоживанням створює можливість реалізації автономних, енергоефективних та функціонально гнучких систем, доступних для широкого кола користувачів у побутових умовах [9, 10]. Таким чином, поєднання соціальної потреби у підвищенні рівня пожежної безпеки населення з технічними досягненнями у галузі вбудованих систем обґрунтовує доцільність і своєчасність розробки портативних недорогих мікроконтролерних систем пожежної сигналізації для житлового сектору [11].

Метою даної роботи є розробка пожежної сигналізації на базі платформи NodeMcu V3 ESP8266, який за технічними характеристиками не поступається широко використовуваним у таких системах мікроконтролерам Arduino, давачів диму, газу, температури і вологості, вібрації, реле та GSM-модуля для обміну даними.

Принципи функціонування системи пожежної сигналізації

Система пожежної сигналізації являє собою комплекс технічних засобів, основним призначенням якого є виявлення пожежі, обробка і надання у заданому вигляді повідомлення про пожежу на об'єкті, який необхідно захистити, спеціальної інформації для видачі команд на увімкнення автоматичних пристроїв пожежогасіння та керування іншими технічними засобами. Будь-яка система пожежної сигналізації складається із пожежних оповіщувачів, приймально-контрольних приладів, звукових і світлових оповіщувачів, технічних засобів передачі інформації до пультів централізованого спостереження, пультів зв'язку пожежних частин [12].

Використовувані на даний час системи і технічні засоби пожежної сигналізації класифікуються за наступними основними ознаками:

1) за призначенням (прилади охоронної сигналізації і сигнально-пускові блоки автоматичних установок пожежогасіння, прилади пожежної сигналізації);

2) за видом технічних засобів (установки пожежної сигналізації, системи передачі сповіщень про пожежу, об'єднані пульти централізованого спостереження, прилади пожежної сигналізації спеціального призначення, зокрема, для різних видів транспорту – автомобільного, водного, залізничного, повітряного) і об'єктів спеціального призначення);

3) за конструктивним виконанням (звичайне, пилезахищене і водозахищене, вибухозахищене);

4) за видом ліній використовуваного зв'язку (лінії міської телефонної мережі, лінії електричної мережі, радіоканал, спеціальні провідні і оптоволоконні лінії);

5) за видом підключених оповіщувачів (спеціальні – призначені для роботи в певних умовах; уніфіковані – реагують на певну первинну ознаку пожежі; комбіновані – реагують на декілька первинних ознак пожежі);

6) за елементною базою функціональних вузлів (електромеханічні, на основі дискретних напівпровідникових та інтегральних мікросхем, мікропроцесорів та комбінованої бази).

Запропонована класифікація може бути розширена за рахунок врахування інших ознак, які впливають на найважливіші характеристики систем пожежної сигналізації.

Загалом будь-яка система автоматичного протипожежного захисту у своєму складі містить ряд підсистем: оповіщення людей про пожежу і керування евакуацією; автоматичне видалення диму; установки пожежогасіння; пожежна сигналізація.

Структурна схема пожежного оповіщувача у загальному вигляді може бути представлена, як показано на рис. 1.

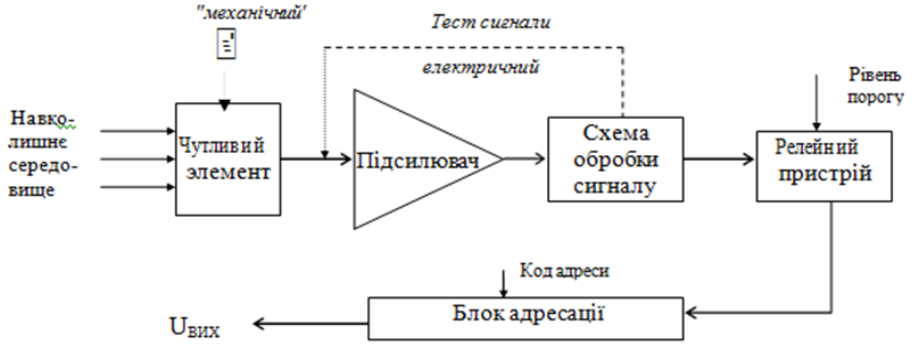


Рис. 1. Узагальнена структурна схема пожежного оповіщувача

Таким чином, при розробленні пристрою необхідно враховувати вище зазначені особливості функціонування систем пожежної сигналізації.

Основні елементи пристрою та його реалізація

1. Платформа NodeMCU V3 ESP8266. Для реалізації даного проєкту було використано платформу NodeMCU V3 ESP8266 (рис. 2), оснащений вбудованим модулем Wi-Fi, що дозволяє легко підключати пристрої до Інтернету без додаткових модулів. NodeMCU підтримує живлення через USB або через VIN 5 В і має вбудований стабілізатор, що спрощує живлення та підключення периферії. Перевагою модуля є висока тактова частота процесора ESP8266 (80–160 МГц) проти 16 МГц у Arduino Uno, що дозволяє швидше обробляти дані та виконувати складні алгоритми. NodeMCU має менший розмір та інтегрований програматор, що спрощує прошивку через USB, тоді як для Arduino іноді потрібен зовнішній програматор. Усі ці фактори роблять NodeMCU V3 ESP8266 більш придатним для IoT-проєктів, автоматизації та дистанційного моніторингу в порівнянні з класичною платформою Arduino.

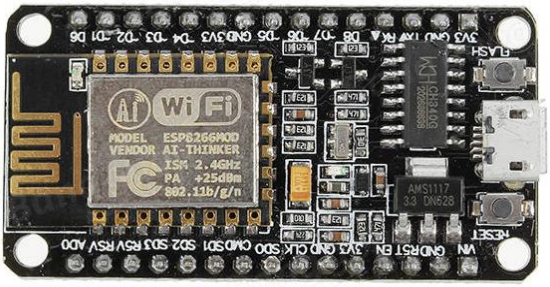
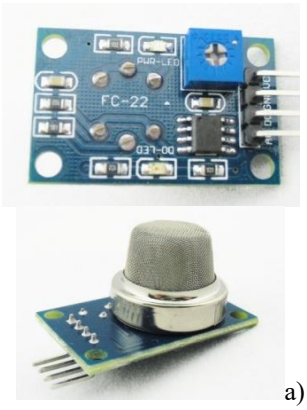
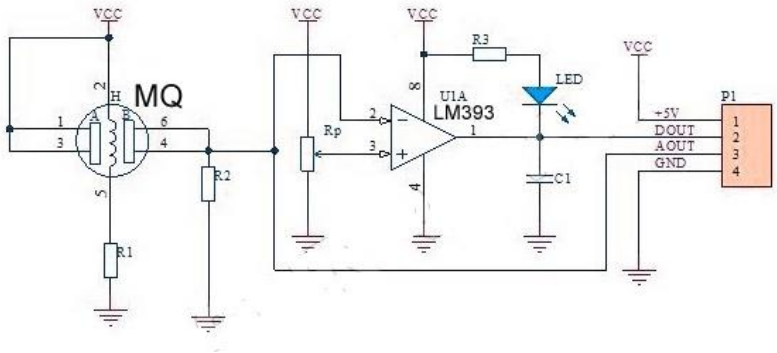


Рис. 2. Модуль NodeMCU V3 ESP8266

2. Давач газу/диму MQ-2. Давач газу, побудований на базі газоаналізатора MQ-2, дозволяє виявляти наявність в навколишньому повітрі вуглеводневих газів: пропан у кількості 0,2 ÷ 5 проміле; метан – 5 ÷ 20 проміле; бутан – 0,3 ÷ 5 проміле; водень – 0,3 ÷ 5 проміле; діапазон чутливості до диму – 300 ÷ 10000 ppm; час відгуку не перевищує 10 с). Давач можна використовувати для виявлення витоків промислового газу і задимлення. Вихідним результатом є аналоговий сигнал, пропорційний вмісту газів, до яких чутливий газоаналізатор. Візуальне зображення давача та електричної схеми подано рис. 3.



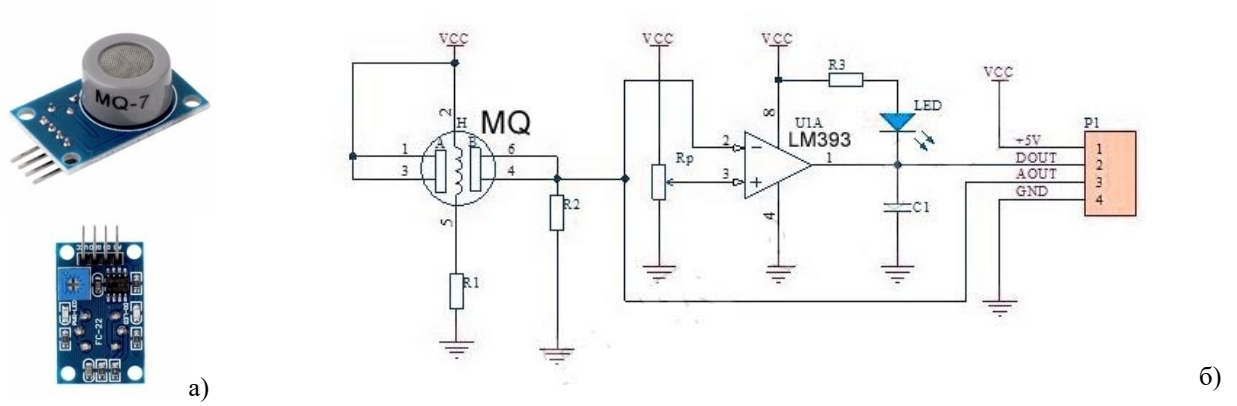
а)



б)

Рис. 3. Модуль давача газу/диму MQ-2 (а) та його електрична схема (б)

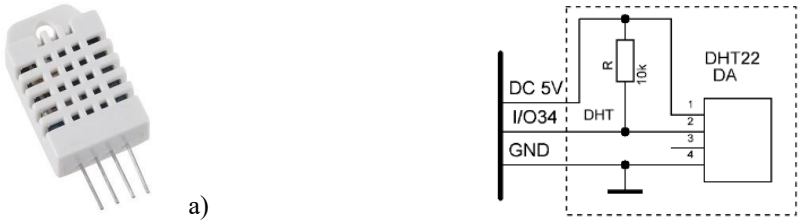
3. Давач газу MQ-7. Давач газу MQ-7 дає можливість виявляти чадний газ при його концентрації 10 ÷ 1000 ppm, чутливість давача становить 3 %; час відгуку – 1 с, час відповіді – 30 с. Візуальне зображення давача та його електричної схеми подано рис. 4.



а) б)

Рис. 4. Модуль давача газу MQ-7 (а) та його електрична схема (б)

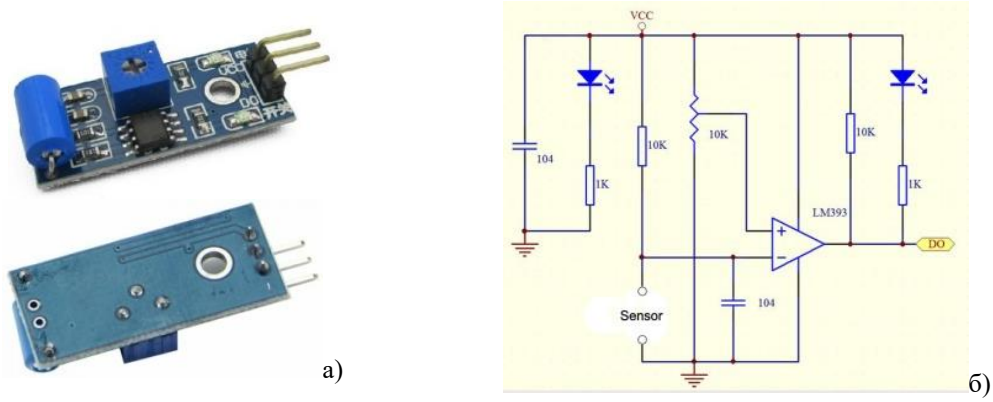
4. Давач температури і вологості DHT22. Давач DHT22 добре підходить для вимірювання вологості і температури в приміщенні, що дасть змогу для більш точного контролю пожежної сигналізації. Діапазон вимірювання вологості – 0-100 %, температури – -40 ÷ 80°C; точність вимірювання вологості – ± 2 % RH, температури – ± 0.5°C. Вигляд давача та його електрична схема подані на рис. 5.



а) б)

Рис. 5. Давач температури і вологості DHT227 (а) та його електрична схема (б)

5. Давач вібрації SW-420. Давач SW-420 реагує на вібрацію. Основна сфера застосування – детектування вібрацій в охоронних сигналізація, протиугінних системах і детектування землетрусів. На корпусі модуля містяться два світлодіоди – червоний і зелений. Червоний світлодіод підключається до виводу живлення і горить, коли на модуль подається напруга живлення. Зелений світлодіод підключається до цифрового виходу і світиться, коли давач спрацьовує. Чутливість давача регулюється потенціометром, який розташований на платі. Візуальне зображення давача та його електричної схеми подано рис. 6.



а) б)

Рис. 6. Давач вібрації SW-420 (а) та його електрична схема (б)

6. Реле. Модуль 1-канального реле 5 В, 15-20мА для спрацьовування, тобто може управлятися безпосередньо з виводів NodeMCU V3 ESP8266. Включається логічним нулем, вимикається логічною одиницею.

7. GSM модуль M590. Це вузькоспеціалізований пристрій, призначений виключно для обміну SMS та передачі даних через GPRS, що робить його ідеальним для автономних систем моніторингу та IoT-датчиків (рис. 7). Діапазони частот: 900 та 1800 МГц. Напруга від 3,3 В до 4,5 В (рекомендовано 3,9 В – 4,2 В). Модуль потребує високого струму (до 2 А) у пікові моменти передачі даних.

Рис. 7. GSM-модуль M590

Маючи декілька давачів і мікроконтролер, потрібно точно знати, які дані передаються на мікроконтролер. Кожен давач підключений до модуля NodeMCU V3 ESP8266 на пряму і передає цифровий або аналоговий сигнал, який обробляється мікроконтролером. Залежно від рівня сигналу мікроконтролер вираховує, чи потрібно подавати подальший сигнал на реле, пожежну станцію або на пожежну сигналізацію.

Давачі можна підключати на пряму до лінії передачі сигналу до пожежної станції, але такий сигнал буде досить не якісний, оскільки давачі мають різні вихідні сигнали (як цифровий і аналоговий), що може призвести до неправильної роботи пожежної сигналізації. Блок схема роботи пристрою пожежної сигналізації зображена на рис. 8, а схема підключення всіх компонентів на рис. 9.

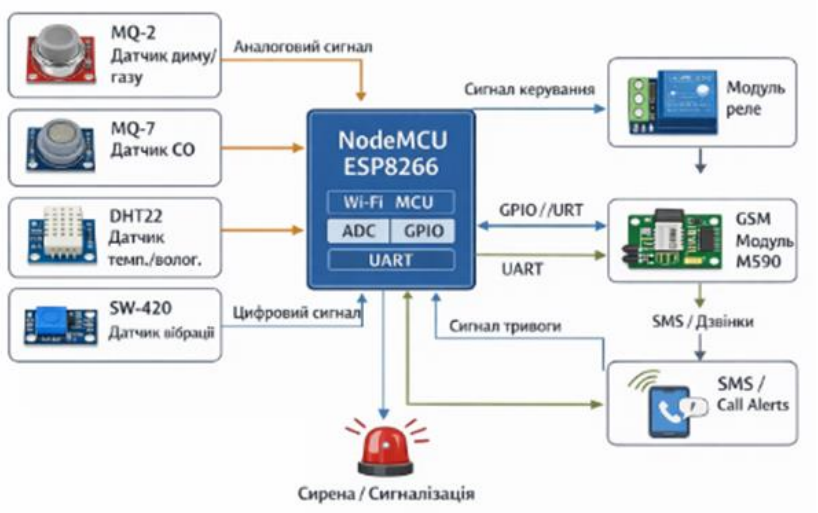


Рис. 8. Блок-схема роботи пристрою пожежної сигналізації

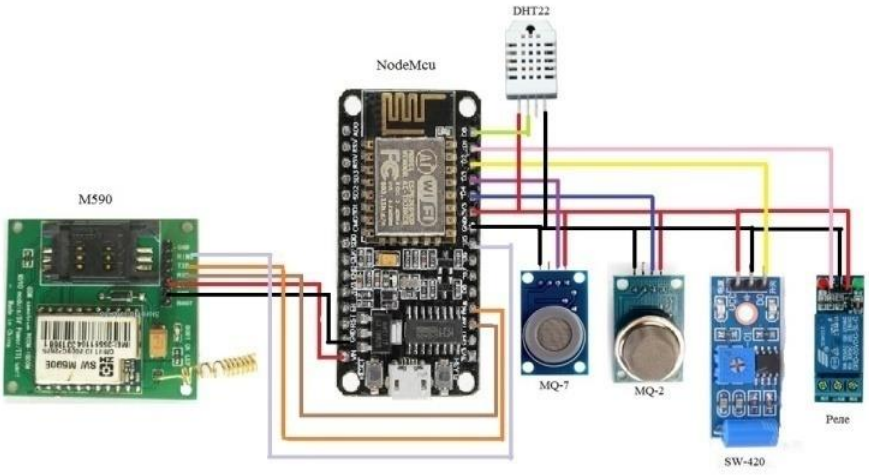


Рис. 9. Підключення давачів і модулів до мікроконтролера

Програмний код для функціонування пристрою написаний мовою C++ у середовищі розробки Arduino IDE з використанням специфічних бібліотек для роботи з периферією ESP8266. Суть програми полягає у створенні циклічного алгоритму моніторингу безпеки: мікроконтролер постійно опитує цифрові та аналогові сигнали з датчиків загрози (дим, чадний газ, вібрація) та порівнює показники температури з критичним порогом. У разі виявлення небезпеки система миттєво активує виконавчий механізм через реле та ініціює передачу тривожного сповіщення через GSM-шлюз.

Особливістю роботи коду є використання SoftwareSerial для взаємодії з модулем M590 за допомогою AT-команд, що дозволяє звільнити основний апаратний UART для налагодження системи через комп'ютер. Оскільки NodeMCU має обмежену кількість аналогових входів, код адаптований під обробку цифрових сигналів ("0" або "1") від датчиків серії MQ, де поріг спрацювання попередньо налаштовується вручну за допомогою вбудованих у модулі потенціометрів. Це забезпечує високу швидкість реакції та стабільність роботи навіть при обмежених ресурсах контролера.

Висновки

Розглянуто основні принципи функціонування систем пожежної сигналізації, запропоновано прототип пристрою для моніторингу стану приміщень щодо їх відповідності умовам пожежного захисту на базі платформи платформи NodeMCU V3 ESP8266 та давачів диму, газу, температури, вологості та вібрації. Написано програмний код для роботи пристрою. Запропоновану систему можна успішно використовувати на об’єктах, де вплив небезпечних факторів пожежі може призвести до травматизму та загибелі людей.

Список літератури:

1. Z. Song, Z. Liu, Y. Li, L. Yang, and R. Chen, “Intelligent Smoke Alarm Based on Microcontroller,” *Frontiers Sci. Eng.*, vol. 5, no. 3, pp. 407–415, Mar. 2025. doi: https://doi.org/10.54691/12nbab76.

2. A. Gaur *et al.*, “Fire Sensing Technologies: A Review,” *IEEE Sensors J.*, vol. 19, no. 9, pp. 3191–3202, May 2019. doi: https://doi.org/10.1109/jsen.2019.2894665.

3. E. N. Ifeagwu and A. D. Adebayo, “The Design and Analysis of a Micro Controller Based Fire Alarm System with Water Sprinkler”, *IJEMI*, vol. 8, no. 1, pp. 1–10, Jun. 2024.

4. P. (Dr.) A. Hussain, “Low-Power Design Techniques for Embedded Processors in Portable Devices: A Review of Methods and Applications”, Vol. 1, No. 8. Zenodo, P. 15–22, Aug. 08, 2025. doi: https://doi.org/10.5281/zenodo.16834435.

5. E. Bender, J. B. Bernstein, and D. S. Boning, “Modern Trends in Microelectronics Packaging Reliability Testing,” *Micromachines*, vol. 15, no. 3, p. 398, Mar. 2024. doi: https://doi.org/10.3390/mi15030398.

6. A. Hassan and A. I. Audu, “A Lightweight CNN Model for Vision Based Fire Detection on Embedded Systems,” *FUOYE J. Eng. Technol.*, vol. 9, no. 4, pp. 624–628, Feb. 2025. doi: https://doi.org/10.4314/fuoyejet.v9i4.9.

7. H. Alqourabah, A. Muneer, and S. M. Fati, “A smart fire detection system using iot technology with automatic water sprinkler,” *Int. J. Elect. Comput. Eng. (IJECE)*, vol. 11, no. 4, p. 2994, Aug. 2021. doi: https://doi.org/10.11591/ijece.v11i4.pp2994-3002.

8. S. N. Rao and K. M. kondiah, “Experimental Paper on Microcontroller based Fire Detection Alarm System,” *Int. J. Eng. Trends Technol.*, vol. 67, no. 2, pp. 50–52, Feb. 2019. doi: https://doi.org/10.14445/22315381/ijett-v67i2p211.

9. S. M. Al-Chalabi, A. M. Al-Chalabi, and R. A. Al-Khafaji, “Innovative pre-fire alert smart detection system-based embedded system,” *AIP Adv.*, vol. 14, no. 5, May 2024. doi: https://doi.org/10.1063/5.0195026.

10.B. Liu, B. Sun, P. Cheng, and Y. Huang, “An Embedded Portable Lightweight Platform for Real-Time Early Smoke Detection,” *Sensors*, vol. 22, no. 12, p. 4655, Jun. 2022. doi: https://doi.org/10.3390/s22124655.

11.O. V. Andreiev, O. Dubyna, and T. Nikitchuk, “The combined alarm system based on ESP32CAM”, *ten*, no. 2(90), pp. 131–135, Dec. 2022. doi: https://doi.org/10.26642/ten-2022-2(90)-131-135.

12.Системи пожежної сигналізації / І.Я. Кріса, О.І. Воробйов: навч: посібник. Львів: Видавництво Львівської політехніки, 2013. 232 с.

Надійшла до редколегії 10.03.2025

Відомості про авторів:

Остапишин Назарій Андрійович – бакалавр кафедри комп’ютерної інженерії та електроніки, Прикарпатський національний університет імені Василя Стефаника / Vasyl Stefanyk Precarpathian National University, Україна; email: nazarii.ostapyshyn.21@pnu.edu.ua

Мандзюк Володимир Ігорович – доктор фізико-математичних наук, професор, професор кафедри комп’ютерної інженерії та електроніки, Прикарпатський національний університет імені Василя Стефаника / Vasyl Stefanyk Precarpathian National University, Україна; email: volodymyr.mandziuk@pnu.edu.ua; ORCID: https://orcid.org/0000-0001-6020-7722

УДК 004.056.5:004.8 **DOI:10.15330/itee.2025.2.02**

І. М. ЛАЗАРОВИЧ, канд. техн. наук, А. Д. КВАСНЮК

ПОРІВНЯЛЬНИЙ АНАЛІЗ ТРАНСФОРМЕРНИХ МОДЕЛЕЙ ТА ГІБРИДНИХ АРХІТЕКТУР У ЗАДАЧАХ ВИЯВЛЕННЯ ФІШИНГОВОГО КОНТЕНТУ

Вступ

Зі стрімким розвитком інформаційно-комунікаційних технологій та широким впровадженням цифрових сервісів фішингові атаки залишаються однією з найбільш поширених і небезпечних загроз у сфері кібербезпеки. Фішинг ґрунтується на методах соціальної інженерії та передбачає створення повідомлень або веб-ресурсів, що імітують легітимні сервіси з метою отримання конфіденційних даних користувачів, зокрема облікових даних, платіжної інформації та персональних відомостей [1]. Постійна еволюція фішингових технік ускладнює їх своєчасне виявлення та знижує ефективність традиційних захисних механізмів.

Класичні підходи до виявлення фішингового контенту, які базуються на сигнатурному аналізі, списках заборонених доменів або ручному формуванні правил, демонструють обмежену ефективність в умовах динамічного розвитку атак. Такі методи не здатні адекватно реагувати на нові варіанти фішингових повідомлень, що відрізняються стилістикою, структурою та мовними особливостями. У зв’язку з цим актуальним напрямом досліджень є застосування методів машинного та глибокого навчання для автоматизованого аналізу текстового та HTML-контенту.

Сучасні методи обробки природної мови (Natural Language Processing, NLP) дозволяють ефективно аналізувати семантичні та контекстуальні особливості текстів. Особливу увагу привертають трансформерні моделі, зокрема BERT, які завдяки механізму самоуваги здатні враховувати глобальні залежності у тексті. Паралельно активно досліджуються гібридні архітектури [2], що поєднують згорткові нейронні мережі (CNN), рекурентні мережі з довготривалою пам’яттю (LSTM) та механізми уваги. Такі моделі дозволяють одночасно аналізувати локальні текстові патерни та довготривалий контекст повідомлень.

Аналіз останніх досліджень та публікацій

У ранніх дослідженнях проблеми виявлення фішингових атак застосовувалися класичні алгоритми машинного навчання, такі як наївний баєсівський класифікатор [3], метод опорних векторів [3] та логістична регресія [3]. Дані підходи вимагали попереднього ручного відбору ознак, зокрема частот слів, n-грам та структурних характеристик веб-сторінок. Хоча такі методи демонстрували прийнятні результати на обмежених наборах даних, вони не забезпечували достатнього рівня узагальнення та погано працювали з новими типами атак.

Подальший розвиток отримали нейромережеві підходи, зокрема згорткові та рекурентні нейронні мережі [4, 5]. CNN зарекомендували себе як ефективний інструмент для виявлення локальних шаблонів у текстах, таких як характерні фрази або словосполучення, притаманні фішинговим повідомленням. LSTM [6], у свою чергу, дозволили моделювати довготривалі залежності між словами та враховувати загальний зміст повідомлень.

Значний прорив у галузі NLP був досягнутий із появою трансформерних моделей. Модель BERT [7] фактично стала загальноприйнятим стандартом для багатьох задач класифікації тексту завдяки двонапрямленому моделюванню контексту та використанню механізму самоуваги. Разом із тим, висока обчислювальна складність трансформерів та

вимоги до великих обсягів навчальних даних стимулювали подальші дослідження гібридних архітектур, які поєднують переваги різних типів нейронних мереж.

Формулювання цілей статті

Основною метою дослідження є експериментальне порівняння ефективності трансформерної моделі BERT та гібридних нейромережевих архітектур CNN та LSTM і CNN, LSTM та Attention у задачі виявлення фішингового контенту. Для досягнення цієї мети було проведено серію експериментів із використанням K-Fold крос-валідації та стандартних метрик оцінювання якості класифікації.

Методологія та основний матеріал досліджень

У дослідженні використано загальнодоступний набір даних [8], що містить HTML-веб-сторінки, класифіковані як фішингові або легітимні. Початково дані були представлені у форматі SQL-архіву та згодом експортовані у формат CSV для подальшої обробки у середовищі Python.

Для аналізу використовувалися дві основні ознаки: htmlContent, що містить HTML-код сторінок, та isPhish, яка визначає клас веб-контенту. Після попередньої обробки набір даних складав 10 373 записи, серед яких переважали фішингові сторінки, що вказує на наявність дисбалансу класів.

Попередня обробка тексту є критично важливою для задач обробки природної мови [9], особливо у контексті виявлення фішингового контенту. У межах роботи застосовувалися очищення HTML-коду, токенізація, лематизація, видалення стоп-слів та формування біграм. Такий підхід дозволив зменшити шум у даних і зберегти значущі лексичні та контекстні ознаки.

Підготовлені текстові дані були використані для формування векторних представлень [10], які подавалися на вхід досліджуваних моделей, забезпечуючи коректні умови для подальшого порівняльного аналізу трансформерних та гібридних архітектур у задачі виявлення фішингового контенту.

У межах дослідження проведено порівняльний аналіз трансформерної моделі BERT та двох гібридних нейромережевих архітектур на основі згорткових і рекурентних шарів. Обрані моделі відрізняються способом вилучення та узагальнення текстових ознак, що дозволяє оцінити їх ефективність у задачі виявлення фішингового контенту.

Трансформерна модель BERT реалізована на основі полегшеної архітектури DistilBERT, яка забезпечує баланс між якістю класифікації та обчислювальною складністю. Модель приймає на вхід токенізований текст та формує контекстуальні векторні представлення за допомогою механізму self-attention. Для задачі класифікації використовується проміжний повнозв’язний шар із нелінійною активацією ReLU та шар Dropout з коефіцієнтом 0.3, після чого вихідний шар із функцією Softmax формує ймовірнісну оцінку належності контенту до відповідного класу.

Базова гібридна архітектура CNN та LSTM поєднує локальний та послідовний аналіз тексту. На першому етапі токени перетворюються у векторні представлення за допомогою попередньо навчених ембеддінгів GloVe. Згортковий шар Conv1D з 128 фільтрами забезпечує вилучення локальних патернів, характерних для фішингових повідомлень, після чого застосовується шар MaxPooling для зменшення розмірності ознак. Отримані послідовності передаються до рекурентного шару LSTM з 128 юнітами та L2-регуляризацією, який моделює довготривалі контекстуальні залежності. Для зменшення ризику перенавчання використовується Dropout, а фінальна класифікація здійснюється за допомогою шару Softmax.

Розширена гібридна архітектура CNN, LSTM та Attention є подальшим розвитком базової моделі та доповнюється механізмами уваги. Після етапу згорткової обробки застосовується self-attention, який дозволяє моделі зосереджуватися на найбільш інформативних фрагментах тексту. Додатковий контекстуальний шар уваги після LSTM

підсилює релевантні ознаки перед фінальною класифікацією. Такий підхід забезпечує більш гнучке узагальнення інформації та підвищує здатність моделі виявляти приховані фішингові патерни.

Таким чином, у дослідженні охоплено три різні підходи до обробки текстового веб-контенту: трансформерний, гібридний згортково-рекурентний та гібридний із механізмом уваги, що створює основу для об'єктивного порівняльного аналізу їх ефективності.

Для забезпечення об'єктивності результатів застосовано K-Fold крос-валідацію [11], що дозволяє оцінити здатність моделей до узагальнення та зменшити вплив випадкового поділу даних.

У цьому дослідженні проведено порівняльний аналіз основних моделй з використанням кількох метрик оцінки для визначення їх ефективності в задачі класифікації веб-контенту. Зокрема, моделі порівнювалися за такими критеріями:

Precision та Recall. Ці метрики використовувалися для оцінки ефективності кожного методу векторизації у правильному виявленні шкідливого контенту. Precision та Recall визначаються як:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}. \tag{1}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}. \tag{2}$$

F1-міра. F1-міра обчислюється за формулою:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{3}$$

Точність (Accuracy). Загальна точність класифікації визначається як:

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Samples}. \tag{4}$$

Окрім зазначених метрик, для більш повної оцінки та порівняння загальної ефективності класифікації моделей використовувалася метрика AUC. Вона відображає здатність моделі розрізняти релевантний (цільовий) та нерелевантний (нецільовий) контент при різних порогах класифікації.

AUC обчислюється як площа під ROC-кривою (Receiver Operating Characteristic), що відображає залежність рівня істинних позитивних від рівня хибнопозитивних спрацьовувань:

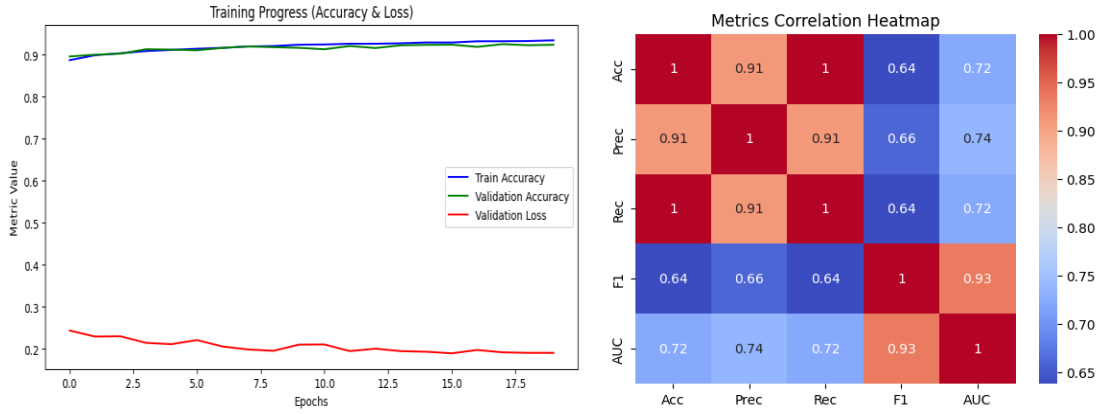
$$AUC = \int_0^1 TRP(t)d(FPR(t)). \tag{5}$$

Результати експериментів для трансформерної моделі BERT наведені на рис. 1 а) та рис. 1 б). Аналіз кривих навчання див. рис. 1 а) демонструє швидку конвергенцію моделі та досягнення високих значень як тренувальної, так і валідаційної точності на рівні близько 0.9. Це свідчить про здатність BERT ефективно відокремлювати фішингові повідомлення від легітимних. Водночас надто висока тренувальна точність може вказувати на потенційний ризик перенавчання, що підтверджується стабілізацією валідаційної втрати без подальшого покращення.

Теплокарта кореляції метрик див. рис. 1 б) показує сильний взаємозв'язок між Accuracy, Precision та Recall, що свідчить про узгоджену поведінку моделі за основними показниками якості. Водночас слабша кореляція між Accuracy та F1-score може вказувати на певну нерівномірність у роботі з різними класами.

Результати для гібридної архітектури CNN та LSTM наведені на рис. 2 а) та рис. 2 б). Криві навчання див. рис. 2 а) демонструють майже максимальну тренувальну точність, що

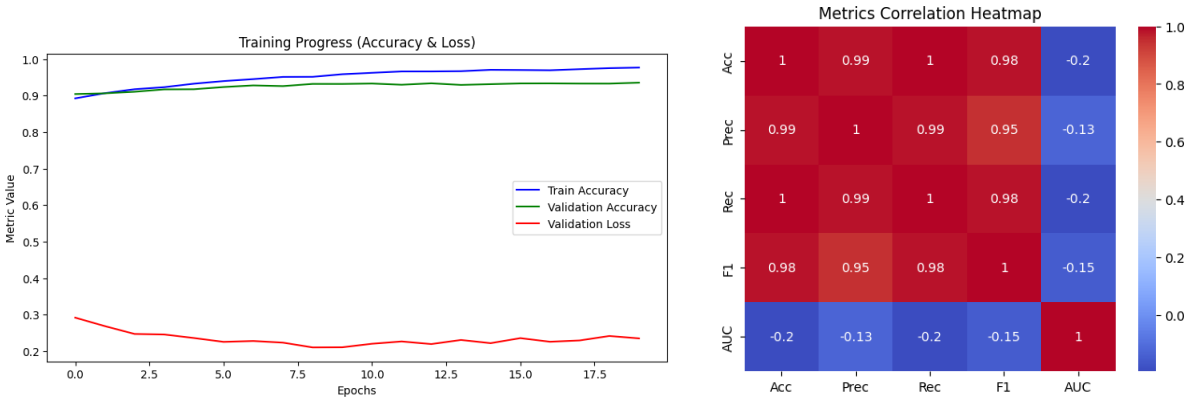
поєднується зі стабілізацією валідаційної точності на рівні близько 93%. Така різниця між тренувальними та валідаційними показниками свідчить про наявність ознак перенавчання. Валідаційна втрата залишається відносно стабільною, проте її значення вказує на можливість подальшої оптимізації архітектури.



а) б)

Рис. 1. а) Процес навчання: зміна точності та втрат для BERT. б) Аналіз кореляції між Accuracy, Precision, Recall, F1 та AUC після проведення проведення K-Fold крос-валідації для BERT

Аналіз кореляції метрик див. рис. 2 б) виявляє слабку або навіть негативну кореляцію AUC з іншими показниками, що означає обмежену здатність моделі ефективно розрізняти класи при зміні порогу класифікації. Це є важливим обмеженням у контексті незбалансованих наборів даних, характерних для задач виявлення фішингового контенту.



а) б)

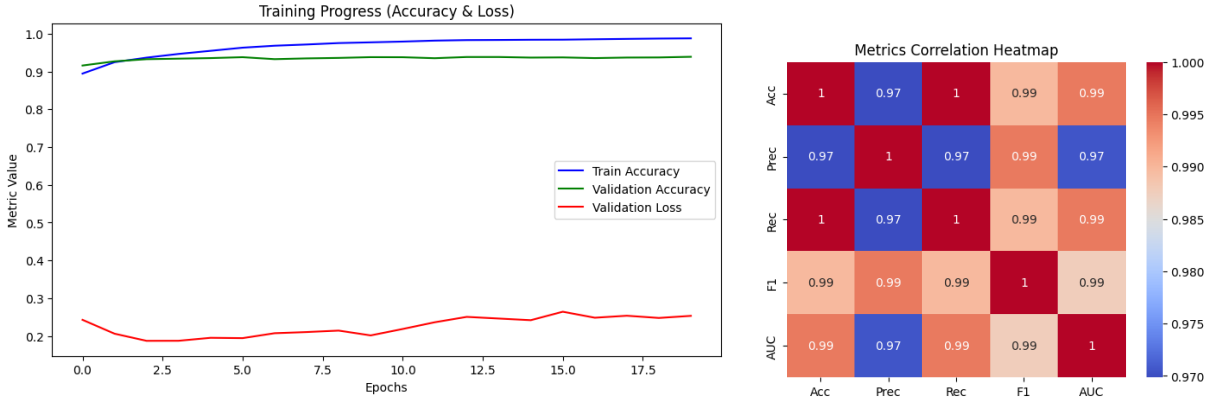
Рис. 2. а) Процес навчання після проведення K-Fold крос-валідації для CNN та LSTM. б) Аналіз кореляції між Accuracy, Precision, Recall, F1 та AUC після проведення K-Fold крос-валідації для CNN та LSTM

Результати для моделі CNN, LSTM та Attention наведені на рис. 3 а) та рис. 3 б). Аналіз процесу навчання див. рис. 3 а) показує зростання тренувальної точності без суттєвого покращення валідаційної, що знову вказує на ризик перенавчання. Водночас використання attention-механізму дозволяє моделі зосереджуватися на найбільш інформативних фрагментах тексту, що позитивно впливає на основні метрики якості.

Кореляційний аналіз див. рис. 3 б) демонструє більш узгоджену поведінку метрик порівняно з базовою моделлю CNN та LSTM.

Результати, наведені в таблиці 1, відображають ефективність трансформерної та гібридних нейромережевих моделей у задачі виявлення фішингового контенту. Трансформерна модель BERT демонструє стабільні результати з показником Accuracy = 0.95, а також близькими значеннями Precision (0.91), Recall (0.87) та F1-score (0.88). Це свідчить

про здатність моделі ефективно враховувати контекст текстових повідомлень, однак її продуктивність поступається гібридним підходам у межах проведеного експерименту.



а) б)

Рис. 3. а) Процес навчання після проведення K-Fold крос-валідації для CNN, LSTM та Attention-механізм. б) Аналіз кореляції між Accuracy, Precision, Recall, F1 та AUC після проведення K-Fold крос-валідації для CNN, LSTM та Attention-механізм

Гібридна архітектура CNN та LSTM забезпечує покращення всіх основних метрик якості класифікації, досягаючи Accuracy = 0.97 та F1-score = 0.91. Отримані результати підтверджують ефективність поєднання локального аналізу тексту за допомогою згорткових шарів із моделюванням послідовного контексту.

Найвищі показники продемонструвала модель CNN, LSTM та Attention, яка досягла максимального значення Accuracy (0.98), а також найвищих значень Precision (0.94) і Recall (0.92). Це вказує на позитивний вплив механізму уваги, який дозволяє моделі зосереджуватися на найбільш інформативних фрагментах тексту та підвищує ефективність виявлення фішингових повідомлень. Незначна різниця у значенні F1-score (0.90) порівняно з моделлю CNN + LSTM може бути пов’язана з балансом між точністю та повнотою класифікації.

Таблиця 1

Метрики навчання моделей

Метод	Accuracy	Precision	Recall	F1-score
BERT	0.951	0.912	0.873	0.882
CNN та LSTM	0.974	0.931	0.901	0.915
CNN, LSTM та ATTENTION	0.980	0.942	0.920	0.931

Загалом результати таблиці 1 свідчать, що гібридні моделі перевершують трансформерний підхід BERT за більшістю показників, а використання Attention-механізму забезпечує додаткове підвищення якості класифікації у задачі виявлення фішингового контенту.

Висновки

Отримані результати дослідження свідчать про високу ефективність гібридних нейромережевих архітектур у задачі виявлення фішингового контенту порівняно з трансформерною моделлю BERT. Проведений аналіз показав, що BERT демонструє стабільні значення Accuracy, Precision, Recall та F1-score, що свідчить про здатність моделі враховувати контекст тексту, проте її продуктивність поступається гібридним підходам, особливо у врахуванні локальних патернів і довготривалих залежностей.

Модель CNN та LSTM забезпечує покращення всіх основних метрик, підтверджуючи ефективність поєднання згорткових шарів для виділення локальних ознак тексту та

рекурентних шарів для моделювання послідовного контексту. Найвищі показники продемонструвала модель CNN, LSTM та Attention, де додатковий механізм уваги дозволяє моделі зосереджуватися на найбільш інформативних фрагментах тексту, підвищуючи точність та повноту виявлення фішингових повідомлень. Водночас експериментальні дані показують ознаки перенавчання та коливання валідаційної втрати, що вказує на потребу додаткової регуляризації та оптимізації гіперпараметрів.

Загалом результати дослідження підтверджують, що гібридні архітектури перевершують трансформерний підхід BERT за більшістю показників якості, а поєднання CNN, LSTM та Attention є перспективним напрямом для побудови практичних систем кібербезпеки. Подальші дослідження доцільно спрямувати на інтеграцію трансформерних моделей із гібридними підходами, застосування методів балансування даних та адаптацію моделей до динамічних фішингових атак.

Список літератури:

1. M. Vijayalakshmi, S. Mercy Shalinie, M. H. Yang, and R. M. U., “Web phishing detection techniques: a survey on the state-of-the-art, taxonomy and future directions,” *IET Netw.*, vol. 9, no. 5, pp. 235–246, Sep. 2020. doi: https://doi.org/10.1049/iet-net.2020.0078.
2. M. A. Adebowale, K. T. Lwin, and M. A. Hossain, “Deep Learning with Convolutional Neural Network and Long Short-Term Memory for Phishing Detection,” in *2019 13th Int. Conf. Softw., Knowl., Inf. Manage. Appl. (SKIMA)*, Island of Ulkulhas, Maldives, Aug. 26–28, 2019. IEEE, 2019. doi: https://doi.org/10.1109/skima47702.2019.8982427.
3. S. Priya, D. Gutema, and S. Singh, “A Comprehensive Survey of Recent Phishing Attacks Detection Techniques,” in *2024 5th Int. Conf. Innovative Trends Inf. Technol. (ICITIIT)*, Kottayam, India, Mar. 15–16, 2024. IEEE, 2024. doi: https://doi.org/10.1109/icitiit61487.2024.10580446.
4. S. Atawneh and H. Aljehani, “Phishing Email Detection Model Using Deep Learning,” *Electronics*, vol. 12, no. 20, p. 4261, Oct. 2023. doi: https://doi.org/10.3390/electronics12204261.
5. U. Daniel, E. Bartholomew, and F. Egbono, “Phishing URL Attack Detection using Logistic Regression and Convolutional Neural Network,” *Int. J. Comput. Appl.*, vol. 187, no. 1, pp. 8–14, May 2025. doi: https://doi.org/10.5120/ijca2025924611.
6. B. Vishnupriya, B. Vikas, and M. D. Choudhry, “Hybrid Deep Learning Framework for ECG-Based Arrhythmia Detection Using CNN, Bi-LSTM, and Transformer,” in *2025 IEEE 4th World Conf. Appl. Intell. Comput. (AIC)*, GB Nagar, Gwalior, India, Jul. 26–27, 2025. IEEE, 2025, pp. 628–632. doi: https://doi.org/10.1109/aic66080.2025.11212074.
7. A. A. Tawil, L. Almazaydeh, D. Qawasmeh, B. Qawasmeh, M. Alshinwan, and K. Elleithy, “Comparative Analysis of Machine Learning Algorithms for Email Phishing Detection Using TF-IDF, Word2Vec, and BERT,” *Comput., Mater. & Continua*, pp. 1–10, 2024. doi: https://doi.org/10.32604/cmc.2024.057279.
8. M. E. Maurer, "Phishload," URL: https://www.medien.ifi.lmu.de/team/max.maurer/files/phishload .
9. S. Luo, Y. Gu, X. Yao, and W. Fan, “Research on Text Sentiment Analysis Based on Neural Network and Ensemble Learning,” *Revue d'Intell. Artificielle*, vol. 35, no. 1, pp. 63–70, Feb. 2021. doi: https://doi.org/10.18280/ria.350107.
10. K. Zhou, Y. Zhou, W. X. Zhao, and J.-R. Wen, “Learning to Perturb for Contrastive Learning of Unsupervised Sentence Representations,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, pp. 1–10, 2023. doi: https://doi.org/10.1109/taslp.2023.3304485.
11. M. Abdar *et al.*, “A review of uncertainty quantification in deep learning: Techniques, applications and challenges,” *Inf. Fusion*, vol. 76, pp. 243–297, Dec. 2021. doi: https://doi.org/10.1016/j.inffus.2021.05.008.

Надійшла до редколегії 30.05.2025

Відомості про авторів:

Лазарович Ігор Миколайович – кандидат технічних наук, доцент, доцент кафедри інформаційних технологій, Прикарпатський національний університет імені Василя Стефаника / Vasyl Stefanyk Precarpathian National University, Україна; email: ihor.lazarovych@pnu.edu.ua; ORCID: https://orcid.org/0000-0001-5219-4714

Кваснюк Андрій Дмитрович – магістрант кафедри інформаційних технологій, Прикарпатський національний університет імені Василя Стефаника / Vasyl Stefanyk Precarpathian National University, Україна; email: andrii.kvasniuk.20@pnu.edu.ua; ORCID: https://orcid.org/0009-0001-0653-3217

УДК 004.9:005.5:640.41 **DOI:10.15330/itee.2025.2.03**

L. B. PETRYSHYN, D.Sc., M. L. PETRYSHYN, Ph.D.

CREATION OF AN IDEF0 FUNCTIONAL MODEL FOR THE INFORMATION SYSTEM OF THE INTERNATIONAL STUDENT YOUTH MEETING CENTER

Introduction

Project Goal and Rationale. The International Student Youth Meeting Center (IMC) is a unique, highly complex environment designed to foster cultural, educational, scientific, and innovative cooperation between student youth from Ukraine and the Republic of Poland. Managing such a multifaceted facility — which integrates accommodation, catering, modern scientific laboratories, virtual reality halls, and large-scale youth projects — is impossible without a robust, modern IT infrastructure. The lack of a unified, automated Management Information System (MIS) can lead to inefficiencies in resource allocation, communication breakdowns, and a reduced capacity to support joint start-ups and international dialogue.

The primary goal of this project is to justify the necessity of implementing advanced Information Systems and Technologies at the IMC and to develop a clear functional model of its business processes. By utilizing the IDEF0 methodology, this study maps the interactions between various center departments, demonstrating how an integrated software environment can optimize operations, ensure security, and support the center's core mission of facilitating international research and innovation projects.

IT Infrastructure and Core Processes. To design an effective Information System, it is crucial to understand and structure the core functional areas of the IMC. The facility operates not only as a research organization but also provides comprehensive accommodation and event services. Instead of viewing these as isolated hotel departments, the system architecture models them as interconnected data-processing nodes:

- administrative and financial modules: requires ERP (enterprise resource planning) solutions to manage HR, accounting, and compliance with national regulations;
- property management system (PMS) / front desk: the software module designed to perform room reservation functions must operate in real-time within a general reservation system. It handles guest registration, financial settlements, and tracks room availability dynamically;
- housekeeping and catering logistics: relies on inventory tracking algorithms and scheduling software to maintain facility standards and manage food and beverage operations, ensuring continuous supply chains without manual tracking;
- security integration: the IT system must synchronize with access control, video surveillance, and automated fire safety networks to ensure the safety of youth participants, especially during mass events;
- IT Service Core: the central backbone ensuring the operability of computer systems, local networks, telecommunications, and access to international educational databases.

The problems that will be solved as a result of the investment project implementation include:

- the lack of an active, stable dialogue between student youth from Ukraine and the Republic of Poland as a foundation for the stability of developing relations between the countries;
- the lack of awareness regarding opportunities for effective cooperation between students of both countries in developing educational activities in higher education within strategically important directions of Ukraine's development;

• the lack of awareness regarding opportunities for effective cooperation between student youth of both countries in undertaking and developing project-based, cultural, innovative, and research activities;

• the lack of practice in creating joint start-ups, implementing innovative ideas, and conducting joint research between students of both countries.

The activities of the International Student Youth Meeting Center aim to provide the following services:

• organization of international youth scientific and practical conferences, seminars, symposia, exhibitions, and forums;

• organization of thematic summer schools in the fields of history, politics, and culture (economic, political, customs);

• assistance in organizing various youth events for students from our country and other countries;

• organization of open lectures and seminars on history, politics, culture, and domestic studies for school and university students from Ukraine and other countries, especially EU countries;

• searching for and facilitating the establishment of new contacts to create partnerships between youth from different countries in the implementation of youth projects and programs;

• searching for sources of co-financing for socio-cultural, innovative projects, and youth exchange programs;

• organizational assistance for the stay of young people from other countries in Ukraine (at the IMC);

• maintaining tourist flows for the Bukovel tourist complex during the busiest periods (from December 1 to March 1) [1].

Business process analysis for IT integration

To analyze and reorganize these business processes for IT integration, the high-level CASE tool AllFusion Process Modeler (BPwin) was used, supporting the IDEF0 functional modeling methodology. The main advantages of this methodology are its simplicity, accuracy, and intuitive graphical representation of complex IT interactionsIn this project, the International Student Youth Meeting Center of Ukraine and Poland is considered a research organization. However, the Center provides the following services:

• provision of accommodation (rooms),
• room service,
• provision of food/catering,
• provision of conference rooms,
• provision of computer laboratories,
• organization of leisure and recreation,
• organization of the enterprise's accounting and financial work,
• organization of the security service work in the hotel.

Hotel Services and Departments. Hotel services and departments can be classified according to various operating methods. According to one method, hotel services and departments are divided as follows:

• revenue centers: these are units that generate direct profit by directly selling services and goods to guests; they include rooms, restaurants, bars, and others;

• support centers: these include the economic department, accounting, planning department, personnel department (HR), engineering, and technical service; i.e., units that generate indirect profits.

A widely used method is the classification of hotel services into front of the house and back of the house, which is based on the degree of contact between employees of a given department and the customers. The reception and restaurant belong to the front of the house, while the kitchen and accounting belong to the back of the house. Employees of the latter group do not have direct contact

with customers. Training for employees of these two groups is conducted according to different programs, and their offices are clearly separated. Even their uniforms are different [2].

Administrative and Management Services. The administrative service is responsible for the organization and management of all services of the hotel complex. It resolves financial problems and personnel issues, creates appropriate working conditions for hotel staff, and monitors compliance with relevant standards and rules regarding occupational safety, fire safety, and environmental protection. Most often, this service includes: a secretary, financial service, personnel service, an ecologist, and fire safety inspectors.

Reception and Accommodation Service. The Reception and Accommodation Service (RAS) is called the "face of the hotel" because the first impression a guest receives of the hotel complex depends on this service. The service includes: administrators, receptionists, cashiers, doormen, porters, bellhops, and concierges. The main task of the RAS is to ensure the smooth operation of customer service from the hotel door to the room door. The main functions of the RAS include: meeting and greeting the guest; guest registration; processing settlement documents; primary calculations; luggage delivery and escorting the customer to the room; assistance in moving to the room; tracking and accounting for changes in the number of rooms.

The function of client registration and check-in is performed by the reception. An administrator, receptionist, and cashier, or a person performing these functions, work here. Currently, two basic principles of organizing the reception service work are common:

- the American principle: based on the interchangeability of employees. If necessary, any employee can replace a colleague and perform their functions. This allows the registration process not to slow down in the absence of one of the employees;
- the European principle: based on a clear division of functions between employees and forbids the replacement of one employee by another [3].

Reservation Service. The customer service process begins with reservation, which means ordering places and rooms. Reservation functions are performed by hotel reservation managers or directly by the Reception and Accommodation Service. As a rule, a tourist or businessman who does not want to deal with the difficulties of renting temporary housing will necessarily contact such a service and submit a request for booking a place or room.

The software module designed to perform hotel room reservation functions operates in a "confirmation/rejection" mode regarding time within the general hotel chain reservation system or autonomous operation. The functions of the reservation service include:

- accepting applications and processing them;
- preparation of necessary documentation – arrival schedule for each day (week, month, quarter, year), room status charts;
- applications are accepted by phone, fax, mail, and using computer reservation systems.

Every order must contain the following information: date and time of arrival; approximate date and time of departure; number of guests:

- room category (suite, class);
- room amenities (bath, shower, TV, fridge, safe, mini-bar, etc.);
- catering services (breakfast only, half board, full board);
- price (when determining the price, it must be precisely specified what the guest pays for – for the entire stay, for one day, for each resident, accommodation only, accommodation and board, accommodation and breakfast, etc.);
- name and initials of the person paying the bill (or name of the organization);
- type of payment (cash, cashless, credit card);
- special requests (advance booking of a restaurant table, transfer, possibility of keeping an animal in the room, etc.).

The organization applying for the reservation also provides its details (name, address, telephone, fax, bank account number, etc.). In the event that the hotel can provide the requested accommodation services, the organization must receive a confirmation of the application.

Otherwise, it must send a refusal. Confirmation of the application is a special notification that the guest will receive accommodation at the hotel. The message usually contains the confirmation number, the date of the guest's expected arrival and departure, the category of the reserved room, the number of guests, the number of beds, and other specially agreed requirements. To clarify all accommodation details and exclude disputes, it is desirable for the guest to have the message upon arrival at the hotel.

Every reservation request and cancellation must be registered. If a cancelled order is not registered on time, there is a high probability that the room will remain unsold. One of the characteristics of the hotel product as a service is the inability to store it. If a room remains unsold, potential revenue from such a service is lost. In their operations, hotel companies often resort to guaranteed confirmation of applications. This means they confirm the reservation only after receiving appropriate payment guarantees from the client in case the client is late or does not arrive at all. Such guarantees are primarily a prepayment of 50 or 100% of the daily accommodation cost or accommodation for the entire period, as well as information about the client's credit card number [4].

Hotel Support and Housekeeping Service. The housekeeping service can be called the most important unit of the reception and accommodation service. It is the part of the hotel team that creates the atmosphere of hospitality in the hotel. This unit employs the maximum number of employees compared to other services (up to 50% of the staff). These employees service various premises (rooms, public areas, special purpose rooms: entertainment centers, fitness centers, etc.), ensuring appropriate sanitary and hygienic standards, and providing laundry and dry cleaning services. This unit may include a repair department, laundry, etc.

The head of the economic support service (Executive Housekeeper) reports to the General Manager or Chief Engineer. The manager is responsible for the effective work of their unit, i.e., teaching, ensuring motivation, and controlling employees. They must have appropriate training and be able to organize the unit's work, recruit staff, and control costs and orders. The deputy head develops the staff work schedule, prepares a report on the status of rooms, and is responsible for cleaning and the condition of the room fund [5].

Room Service (Room Attendants). The most important function of room service is maintaining the required level of comfort and sanitary conditions in hotel rooms, as well as in public spaces (halls, lobbies, arcades, corridors). The room operations service is led by a manager; subordinates, floor maids, supervisors, stewards, and some other categories of employees report to him.

Maids perform cleaning. This is their main duty. Rooms are cleaned regardless of whether they are occupied or vacant. Rooms are cleaned daily, and after a guest's departure – general (deep) cleaning is performed. Every day, the maid performs current and (if necessary) intermediate cleaning of rooms. When cleaning a room after a guest's departure, in addition to standard functions, the maid's duties include checking the room, changing bed linen and towels, and replacing information materials available in the room. General cleaning of rooms and the entire residential part of the hotel takes place at least once every 10 days. Depending on the type of hotel, each maid cleans and arranges from 16 to 20 rooms a day. The time spent on cleaning depends on the proportion of vacant and occupied rooms (cleaning vacated rooms takes more time).

If a suite consists of several rooms, the cleaning process always starts in the bedroom, then continues in the living room and other rooms. Work ends with cleaning the bathroom. Daily intermediate cleaning in rooms is done if necessary and if conditions for cleaning exist. When cleaning a room after a guest's departure, additional functions of the maid include: checking the room, changing bed linen and towels, and replacing information available in the room [6].

Catering Services (Food and Beverage). This unit is an integral part of the hotel business because without a table there is no hospitality. Hotel restaurants represent not only the prestige and face of the hotel but also a main source of income (about one-third of the hotel complex's income). A hotel without a restaurant is just a "bed"; a person must first eat well and only then sleep. When organizing services in restaurants (cafes) of hotel complexes, the following meal plans are usually

offered: full board (three meals a day – breakfast, lunch, and dinner); half board (two meals a day – breakfast plus lunch or dinner); breakfast only (one meal).

Security Service. The security service ensures the maintenance of order and the safety of hotel customers. In recent years, the problem of security has become very relevant, especially in the hospitality industry. International conflicts, the wave of crime and terrorism, illegal arms and explosives trade – all these factors can affect the safety of guests and hotel staff in all countries.

The hotel complex is characterized by threats of a natural, man-made, and environmental nature, as well as terrorism and crime. Currently, the most dangerous is the threat of a terrorist act, which can lead to a large number of victims, creating an atmosphere of fear, disrupting the normal functioning of the hotel, and causing the loss of the positive tourist image of the hotel or the region as a whole. Particular attention should be paid to the professionalism of security personnel, as well as technical security measures on site. To have a constant influx of tourists and run a successful business, it is necessary to improve the hotel's security system, i.e., regularly carry out a range of organizational, methodological, technical, and other activities that ensure the hotel's full autonomy in solving security problems, including terrorist threats.

The daily work of security officers includes a thorough inspection of the protected area (every 2 hours), constant contact with all regular services of the hotel complex, exchange of information about suspicious persons and objects, etc. It is also important to establish active cooperation with territorial law enforcement agencies. When conducting mass events, an inspection of the premises by a cynologist with a dog trained to search for explosives is necessarily carried out.

It is necessary to constantly improve technical security means. It is desirable that video surveillance be organized in the central lobby as well as on the hotel floors. Instructions regarding fire safety measures should be developed. All employees should be allowed to work only after passing fire safety training (which is marked by their signature in a special log). Equipped smoking areas, the procedure for disconnecting electrical appliances, people evacuation plans, fire alarm systems, etc. – all these are elements of the fire safety system. It is also necessary to know the features of fire safety in facilities with a large number of people. In particular, 50 or more people cannot be at one emergency exit at the same time. It is forbidden to block escape routes and exits.

Instructions on how to act on duty when receiving a fire signal and failure of fire automation systems should be placed in the hotel control rooms. Recently, requirements have been introduced for the installation of thermal sensors reacting to temperature increase (up to 30-35 °C) and smoke. Any kind of production and storage rooms containing explosives and flammable materials can be organized in hotel rooms. All hotel customers must know the fire regulations. The security system in the hotel will be effective only if all employees take part in this work, and the specific features of the enterprise are taken into account [7].

IT Service. The IT service ensures the operability of computer systems that support the work of all hotel departments, including reception and accommodation, catering, food and material storage, the business center, boutiques, accounting, the secretariat, the telephone exchange, the marketing department, etc. The IT service typically deals with the maintenance of the computer system that ensures the operation of the hotel's engineering and technological equipment. It should be noted that modern computer systems cover all points of sale and enable immediate settlement for the client upon their departure from the hotel. The hotel's computer network should have access to other computer systems, including international ones for hotel reservations, transport, excursions, theater tickets, etc. Currently, the computer system serves as an operational form of telecommunication for the hotel's internal operations and interaction with the external environment [8].

Creation of an IDEF0 functional model for the information system

To analyze and reorganize business processes, the high-level CASE tool AllFusion Process Modeler (BPwin) was used, which supports the following methodologies:

- IDEF0 (functional model);

- DFD (Data Flow Diagram);
- IDEF3 (Workflow Diagram).

AS-IS Model. IDEF0 Diagram. The functional model aims to describe the existing business processes within the enterprise (the so-called AS-IS Model) as well as the ideal state—the target to be achieved (the TO-BE Model). The main advantages of this methodology are its simplicity, accuracy, and intuitive graphical representation. In IDEF0, the modeled system is represented as a set of interdependent activities (tasks or functions) that interact with the environment as a whole. A model in IDEF0 notation is a system of hierarchically organized and interconnected diagrams, comprising a context diagram and child (decomposition) diagrams at various levels. The context diagram provides a general description of the system and its links to the external environment; child diagrams describe fragments of the system (from general to specific).

The process of providing IMC services is quite diverse, encompassing a set of basic accommodation and catering services for guests staying at the hotel, as well as various additional services, such as modern projects, leisure, and recreation. The scope of additional services depends on the specifics of the projects to be implemented at the IMC, its own conceptual approach to service provision, and the requirements established by the Vasyl Stefanyk Precarpathian National University. Local conditions allow for organizing activities on the outskirts of the center, and the available infrastructure facilitates travel and hiking in the high mountains, as Ukraine's largest mountain ranges, the Gorgany and the Chornohora ridge, are located nearby.

Today, basic services include hotel security, which satisfies a crucial guest need: peace of mind and safety regarding both their person and their belongings. Hotel operations, like those of any company, are impossible without well-established accounting and financial work. At the same time, for the operation of a hotel enterprise to be stable under market conditions, its results must be competitive; therefore, market research activities regarding hotel services are essential.

Building the information system model begins with a description of the enterprise's (system's) functioning as a whole in the form of a context diagram. The activities of the IMC are regulated by a range of legislative, executive, regulatory, statutory, organizational, and methodological documentation, as well as the Statute of the Vasyl Stefanyk Precarpathian National University. The following are used as controls for the "Provision of IMC Services" process (Fig. 1) (control arrows):

1. Legislative and executive regulations of Ukraine (legislation, standards, and rules applicable to the country's tourism industry);
2. Orders and regulations of administrative bodies reflecting the regional specifics of the hotel industry;
3. Standards of ethics, aesthetics, and psychology: IMC employees must adhere to the rules of ethics, aesthetics, and psychology when interacting with the center's clients;
4. Internal rules and methods of IMC work: specific to particular hotel services, the so-called corporate culture;
5. Orders and regulations of the Vasyl Stefanyk Precarpathian National University.

To carry out service activities, the following are necessary (process inputs):

1. Consumers (Clients): those wishing to stay at the hotel;
2. Payment methods for provided services, ensuring the profitability (profit) of the operation – the goal and essence of the activity;
3. Information on the service market, allowing for the maintenance of a specific level of service quality and operational improvement;
4. Logistics providers;
5. Ideas for project implementation.

Service activities in the hotel are realized by (process mechanisms):

1. Existing infrastructure (a set of facilities, premises, management systems, communications, etc., supporting the activity);
2. Materials and technical resources (technological equipment, furniture, bed linen, tablecloths, household and technical appliances, office equipment, etc.);

3. Personnel of various structural units (services) of the hotel;
4. Information-technical complex (computer labs, electronic library).

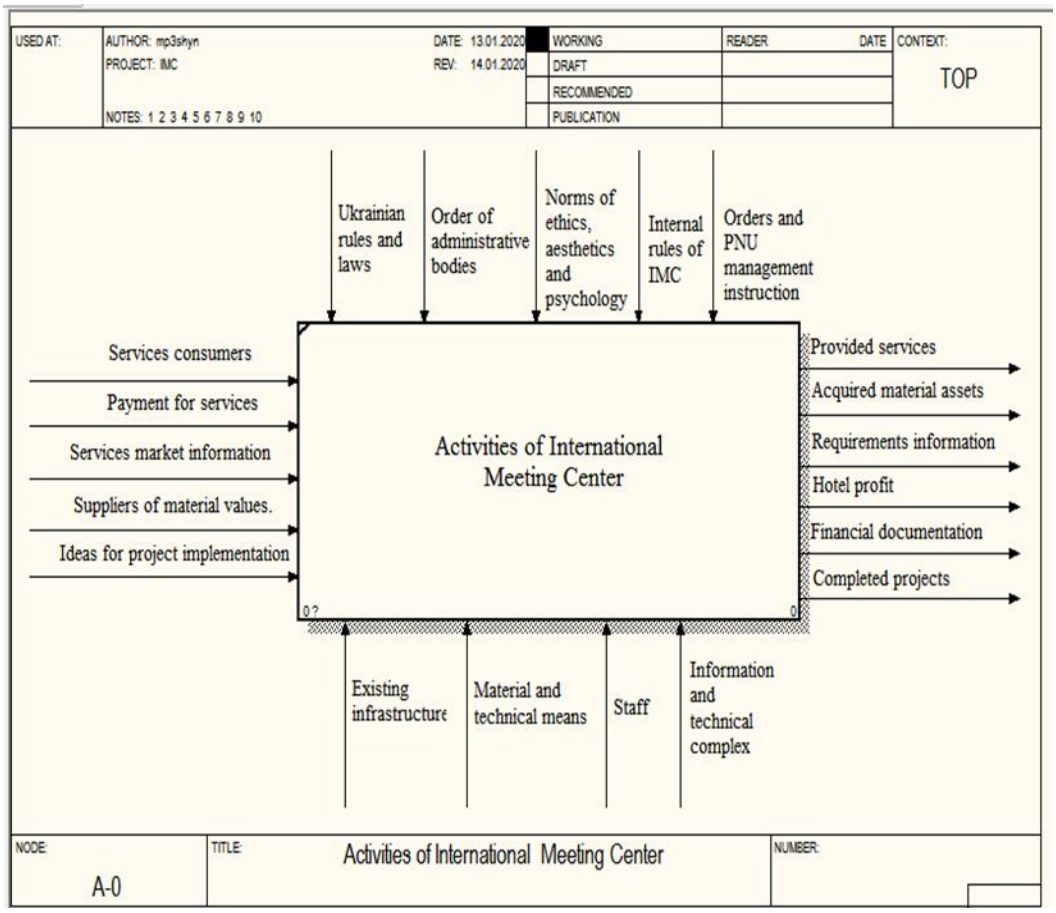


Fig. 1. IDEF0 Context Diagram. Functioning of the IMC

In this case, the results or outputs of the process of providing hotel services at the IMC will be:
1. Provided services;
2. Acquired material assets;
3. Information on consumer and market requirements regarding hotel services;
4. Hotel income (Profit);
5. Financial documentation (invoices);
6. Implemented projects.

Fig. 2 presents the A0 decomposition diagram covering the same modeling domain. In this diagram, each function is represented by corresponding rectangular blocks arranged in order of priority (from the top-left corner to the bottom-right corner). Each block (performed function) possesses appropriate interfaces, consisting of boundary elements presented in the context diagram and internal arrows connecting specific functions. As can be observed, functions (blocks) 1, 2, and 3 are interconnected; that is, the output of the previous function feeds into the subsequent function. Functions (blocks) 4-7 are independent of blocks 1-3; that is, they can be executed in an autonomous mode. At this stage, their inputs and outputs are described only by boundary arrows that separate the modeled object from the environment.

The input for the first block "Provide Rooms" (Fig. 3) is the "client's need for this service." As a result of executing this function, the client is assigned a room for which they pay; that is, they transition from the status of seeking accommodation to being settled in a specific hotel room. The execution of this function is carried out using the following mechanisms: personnel, material and technical resources, and available infrastructure.

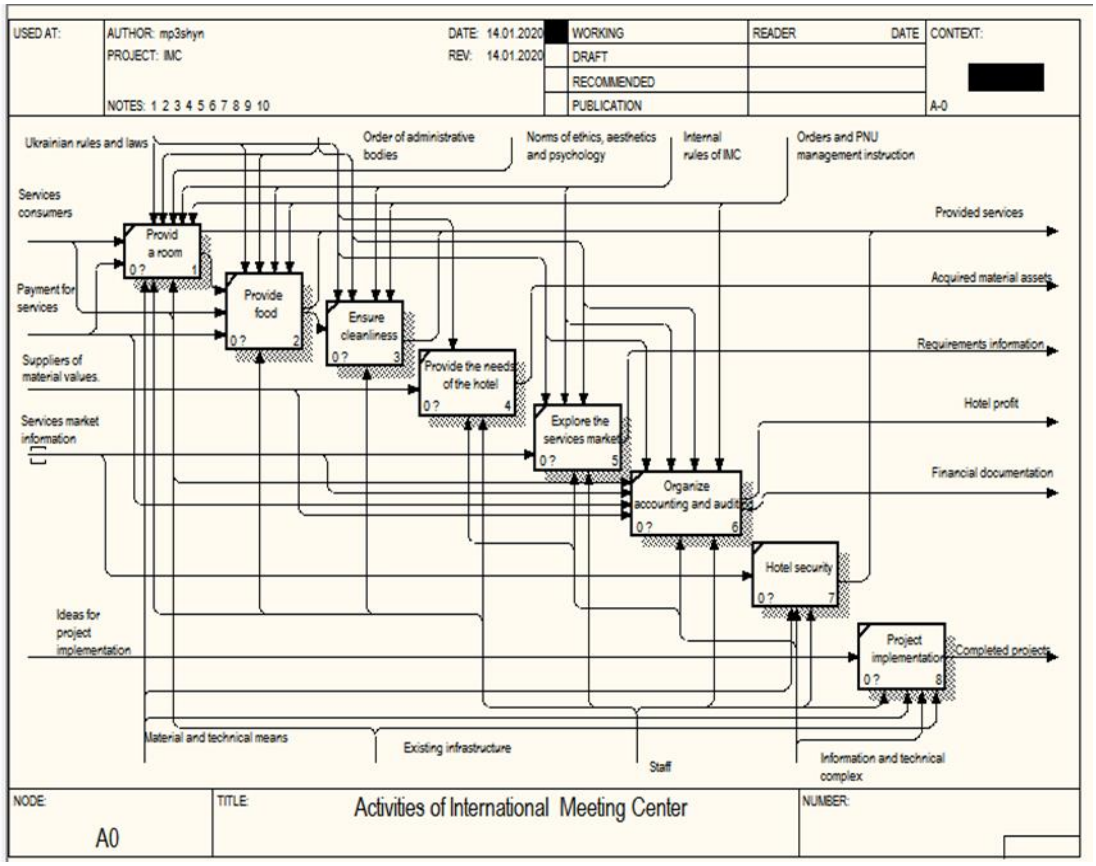


Fig. 2. Extended diagram of the "Provision of Hotel Services" business process

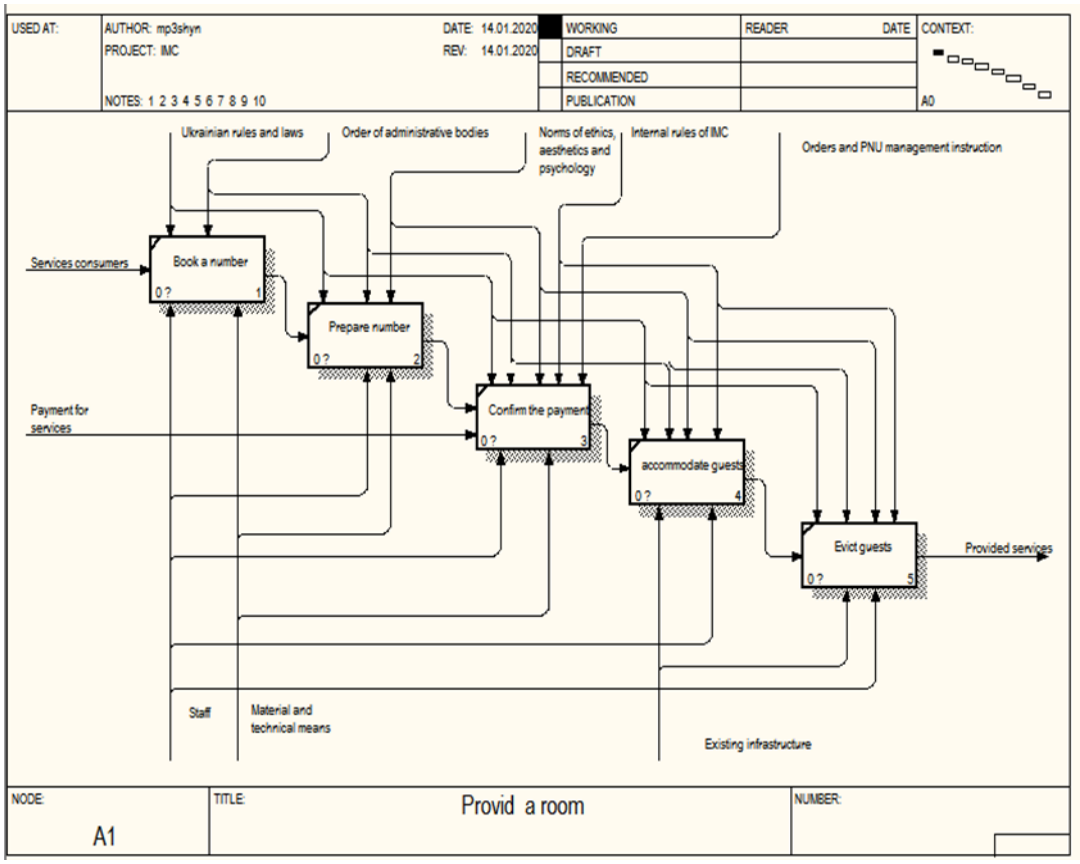


Fig. 3. Extended diagram of the "Room Reservation" business process

A detailed description of the processing process is shown in Figure 3.

The room assignment process can be accomplished by performing the following activities:

1. Room reservation;
2. Room preparation;
3. Payment confirmation;
4. Guest settlement (Check-in);
5. Guest departure (Check-out);
6. The "Food Service Provision" process can be presented as the realization of five functions: 1 – "Purchase raw materials"; 2 – "Prepare the dish"; 3 – "Accept the order"; 4 – "Prepare the order"; 5 – "Accept payment";
7. All these processes are interconnected and carried out in the sequence shown in the diagram (Fig. 4), as the result of one process is necessary for the realization of the next (the output of the previous one is the input for the next).

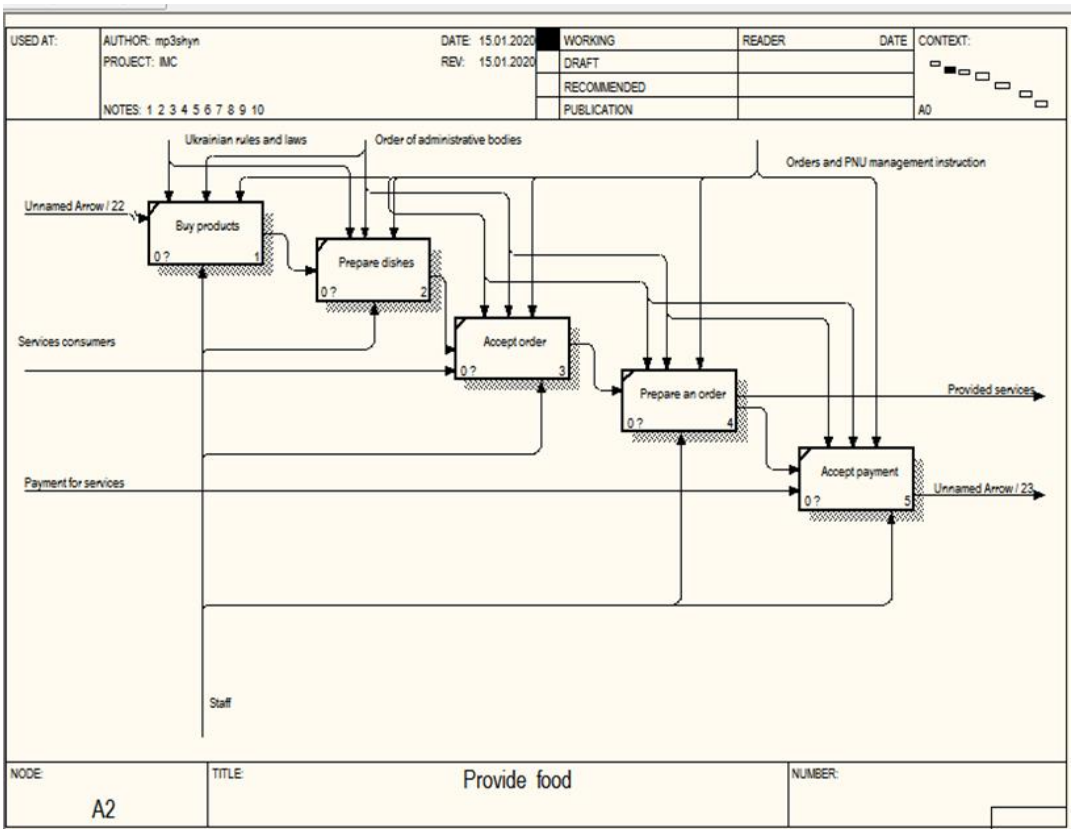


Fig. 4. Extended diagram of the "Food Service Provision" business process

Fig. 5 presents the child diagram of parent unit 3 (from diagram A0), the process "Maintaining Cleanliness of Premises." It includes three functional blocks:

1. Maintaining cleanliness in public areas – lobby, corridors, etc.;
2. Inside the premises – hotel rooms;
3. Maintaining order and cleanliness on the hotel complex grounds – access roads, parking, recreation zone, and other adjacent areas that require systematic cleaning of both debris left by tourists staying at the hotel and debris resulting from natural processes (fallen leaves, snow, etc.).

Cleaning of all facilities is carried out in accordance with the orders and instructions of the hotel management, regulations, methods, and recommendations developed in the hotel, and in compliance with ethics, aesthetics, and ergonomics.

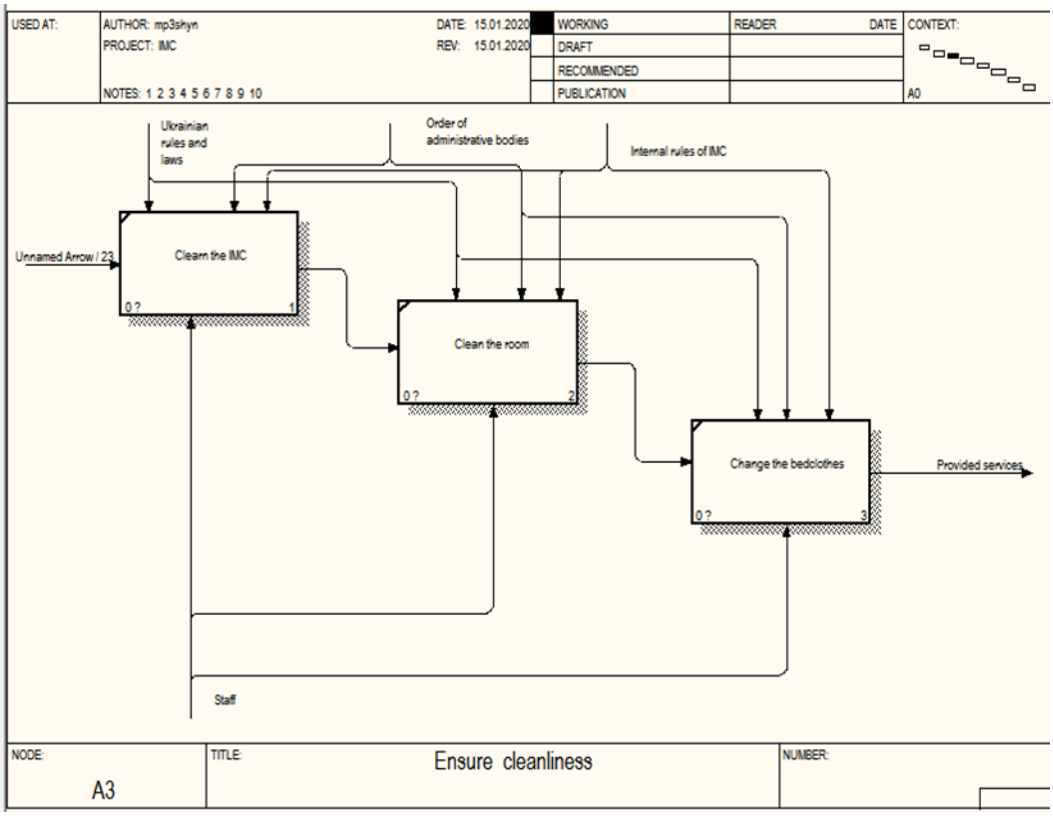


Fig. 5. Functional block diagram "Ensure Cleanliness"

Fig. 6 presents the child diagram of the fourth function of the business process (Fig. 2, diagram A0) – "Ensure Hotel Supplies" (Node A4).

This function is divided into four sub-functions:

1. "Check stock availability";
2. "Create a list of necessary goods and services";
3. "Pay bills for services and goods";
4. "Warehousing".

The function "Organize Accounting and Financial Work" (Fig. 7) in the hotel is performed by the accounting staff. The input to this function involves various types of payments for services provided, as well as purchased materials and technical assets. The result of successful activity is income ensuring financial profit.

The function "Organize Hotel Security" (Fig. 8) is performed by the security service personnel. The inputs to the block are: the need to ensure the guest's peace of mind and the safety of their person and belongings, as well as the property and the entire accommodation infrastructure; and information on scientific and technological progress in this area of activity within the hotel services market. Of particular importance for the performance of this function as a control mechanism is the ethics of interaction with consumers, as was typical for functions 1-3, especially in emergency situations.

The last and most important function is "Project Implementation," since the main idea of the IMC is establishing contacts between the youth of neighboring countries. Therefore, information and technical support, namely 2 modern computer laboratories, an electronic library, and virtual reality halls, provide the opportunity to implement various interesting international youth projects: sports competitions, e-sports championships, etc.

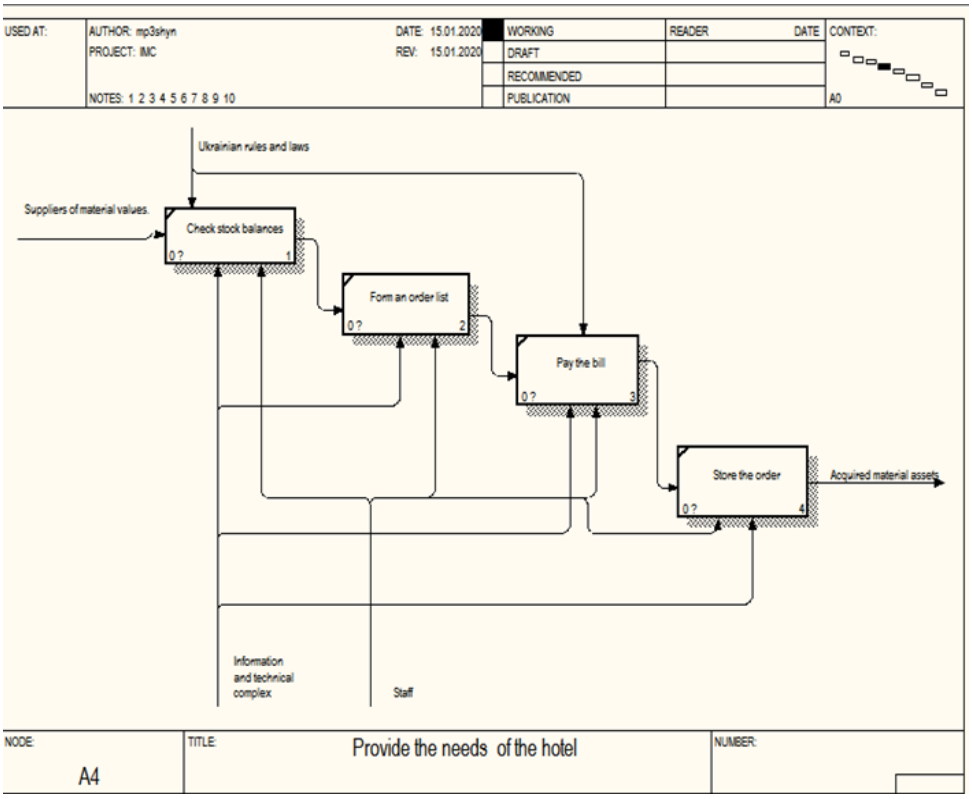


Fig. 6. Functional block diagram "Ensure Hotel Supply Delivery"

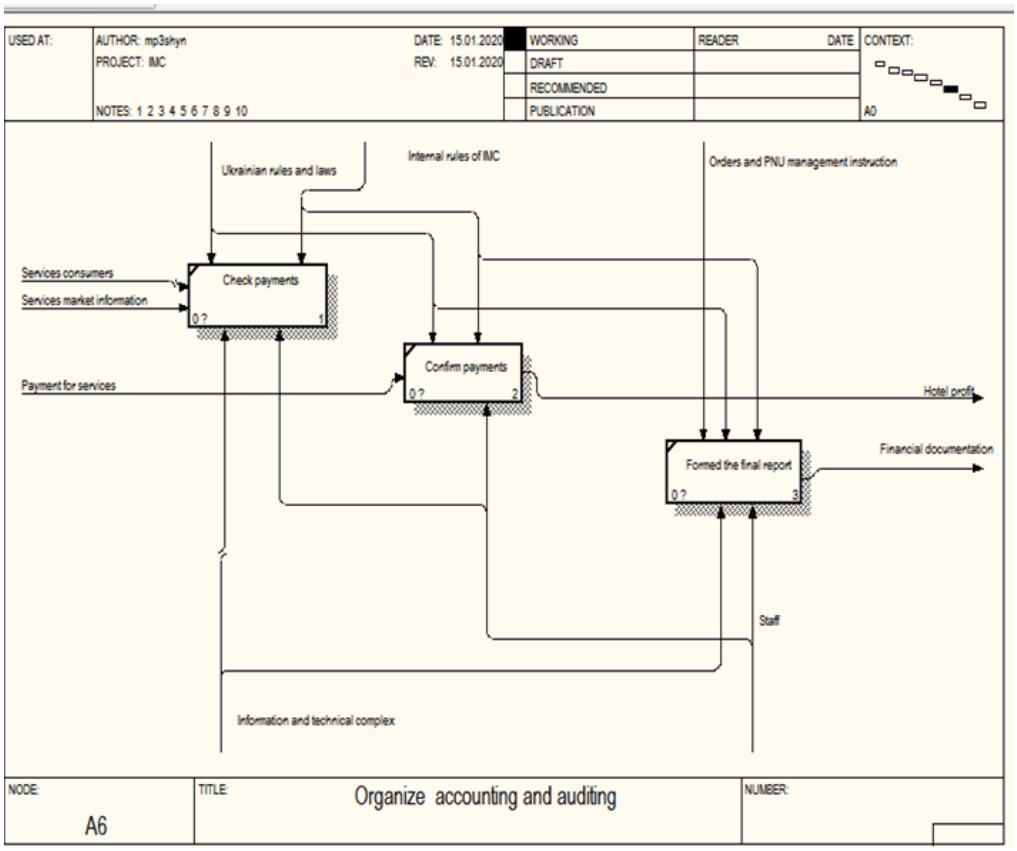


Fig. 7. Functional block diagram of Accounting and Audit

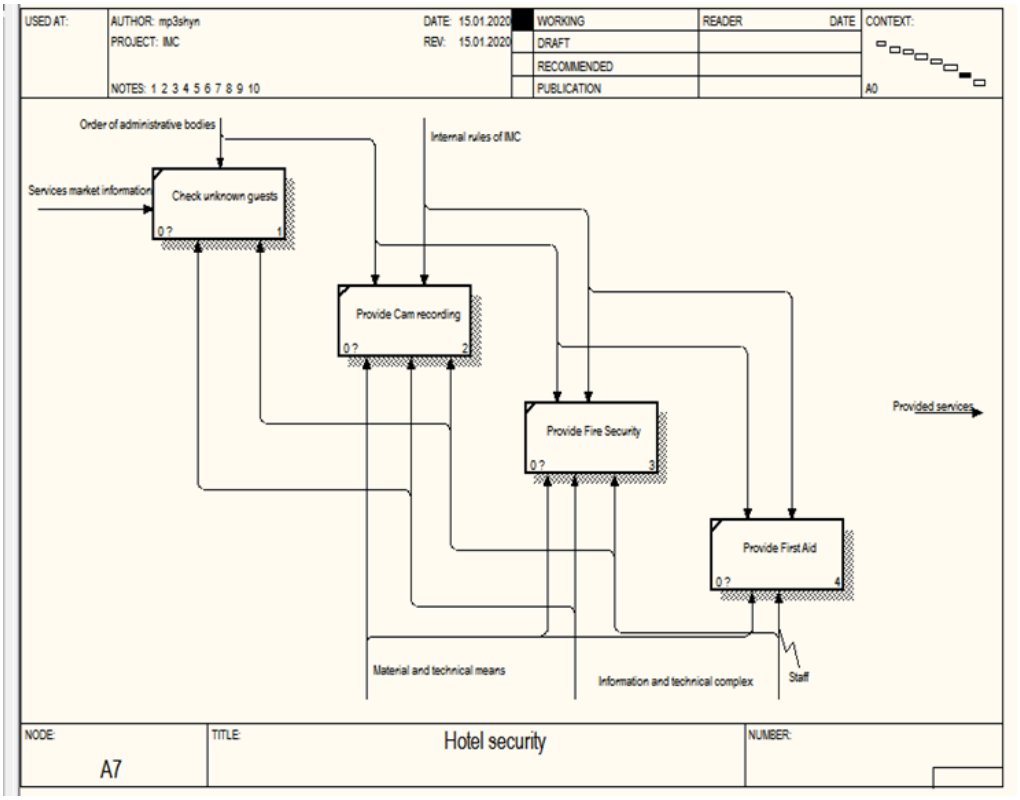


Fig. 8. Functional block diagram of Hotel Security

Conclusions

The study presented the conceptual framework and functional modeling of the International Student Youth Meeting Center (IMC), a strategic project designed to strengthen cooperation between the youth of Ukraine and the Republic of Poland. The IMC is not merely a hospitality facility but a multifunctional environment fostering educational, cultural, and innovative development.

The application of the IDEF0 methodology allowed for a detailed analysis of the center's business processes. By decomposing the system into key functional blocks—ranging from standard hotel operations (accommodation, catering, housekeeping) to complex administrative and security functions – we established a clear structural model for the center's management. The distinction between front-of-house and back-of-house operations, along with the detailed mapping of inputs, outputs, controls, and mechanisms, provides a solid foundation for the efficient organization of the enterprise.

A critical component of the IMC highlighted in the model is the "Project Implementation" function. Unlike standard hotel complexes, the IMC integrates advanced IT infrastructure, including computer laboratories and virtual reality halls. These facilities are essential for the center's core mission: facilitating international youth projects, startups, and e-sports competitions.

In summary, the creation of the IMC, supported by a robust information system and clearly defined business processes, will significantly contribute to:

1. cross-border integration: establishing a stable dialogue between Ukrainian and Polish students;

2. regional development: supporting the tourism potential of the Bukovel region during peak and off-peak seasons;

3. innovation and Education: providing the technical resources necessary for modern scientific and creative collaboration.

References:

1. M. D. Godlevskyi, D. L. Orlovskyi, and A. M. Kopp, “Structural Analysis and optimization of IDEF0 functional business process models,” *Radio Electron., Comput. Sci., Control*, no. 3, Dec. 2018. doi: https://doi.org/10.15588/1607-3274-2018-3-6.

2. “Hotel Basics - Departments In A Hotel.” SetupMyHotel. [Online]. URL: https://setupmyhotel.com/train-my-hotel-staff/securityandloss/765-departments-that-make-a-hotel.html.

3. N. Klada, A. Stroumpoulis, S. Varelas, and N. Georgopoulos, “Information Systems in Sustainable Hospitality and the Creation of Business Value,” *ENTRENOVA - ENTerprise Res. Innov.*, vol. 9, no. 1, pp. 290–295, Dec. 2023. doi: https://doi.org/10.54820/entrenova-2023-0027.

4. A. A. Turki and W. F. Faris, “Modelling manufacturing process using a modified IDEF0 framework: a case study of a car door manufacturing plant,” *Int. J. Eng. Syst. Modelling Simul.*, vol. 2, no. 4, p. 234, 2010. doi: https://doi.org/10.1504/ijesms.2010.038143.

5. E. Rozario, "Functions and divisions in a hotel and how they relate to the room divisions," *WordPress*. [Online]. URL: https://elenrozario.wordpress.com/blog-2-functions-and-divisions-in-a-hotel-and-how-they-relate-to-the-room-divisions/

6. "Hotel room cleaning: 2020 guide," *Hotel Cleaning Services*. [Online]. URL: https://hotelcleaningservices.com/hotel-cleaning-2020-guide/.

7. "Hotel security services," *Cambridge Security Services*. [Online]. URL: https://cambridgesecurityservices.com/hotel-security/.

8. R. Law, R. Leung, and D. Buhalis, “ Information Technology Applications in Hospitality and Tourism: A Review of Publications 2005 to 2007,” *J. Travel & Tourism Marketing*, vol. 26, no. 5-6, pp. 599–623, Sep. 2009. doi: https://doi.org/10.1080/10548400903163160.

Надійшла до редколегії 04.03.2025

Information about the authors:

Petryshyn Lubomyr – doctor of technical sciences, professor, professor in the department of enterprise management, AGH University of Science and Technology / Університет науки і технологій AGH, Poland; professor in the department computer science, Vasyl Stefanyk Precarpathian National University / Прикарпатський національний університет імені Василя Стефаника, Ukraine; email: lpetr@agh.edu.pl; ORCID: https://orcid.org/0000-0003-4168-3891

Petryshyn Mykhailo – PhD, associate professor in the department computer science, Vasyl Stefanyk Precarpathian National University / Прикарпатський національний університет імені Василя Стефаника, Ukraine; email: m.l.petryshyn@pnu.edu.ua; ORCID: https://orcid.org/0000-0001-6319-3768

УДК 681.325+004.4 **DOI:10.15330/itee.2025.2.04**

В. І. ГОЛОТА, канд. техн. наук, В. М. ГРИГА, канд. техн. наук,
Т. Г. БЕНЬКО, А. В. МОРГУН

ВИКОРИСТАННЯ ВЕКТОРНИХ ІНСТРУКЦІЙ МІКРОПРОЦЕСОРА X86-64 В ЗАДАЧАХ ЛІНІЙНОЇ АЛГЕБРИ

Вступ

Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням обсягів даних, що підлягають обробці в режимі реального часу. Сфери застосування високопродуктивних обчислень значно розширилися: від класичного наукового моделювання фізичних процесів та прогнозування погоди до завдань штучного інтелекту, глибокого машинного навчання, обробки потокового відео надвисокої роздільної здатності та криптографічного захисту інформації. У всіх згаданих галузях основна частка обчислювального навантаження припадає на операції над векторами та матрицями.

Тривалий час підвищення продуктивності програмного забезпечення досягалося екстенсивним шляхом – за рахунок зростання тактової частоти центральних процесорів. Сучасна парадигма підвищення швидкодії змістилася в бік архітектурного паралелізму. Однією з ключових технологій, що дозволяє ефективно використовувати ресурси центрального процесора є принцип «одна інструкція – множина даних» (Single Instruction, Multiple Data (SIMD)).

Розширення архітектура x86-64 Advanced Vector Extensions (AVX), надало розробникам для використання 256-бітові регістри YMM, що дозволяє виконувати арифметичні операції над вісьмома числами одинарної точності (float) або чотирма числами подвійної точності (double) за один машинний такт [1, 2]. Це теоретично дозволяє збільшити швидкодію програм у 4–8 разів.

Попри те, що сучасні компілятори мов високого рівня (C++, Rust) мають механізми автовекторизації, вони не завжди генерують оптимальний машинний код через складність аналізу залежностей даних та управління пам'яттю [3, 4]. Тому для створення критично важливих бібліотек, драйверів та ядер обчислювальних систем використовується мова асемблера [6, 7]. Безпосередня оптимізація на рівні асемблерних команд дозволяє розробнику повністю контролювати розподіл регістрів, планування конвеєра команд та роботу з кеш-пам'яттю [8].

Основними високопродуктивними мікропроцесорами у настільних і портативних комп'ютерах є серії Core i3/i5/i7/i9, Ultra від Intel та серії Ryzen 3/5/7/9 від AMD. Основними асемблерами для цих мікроконтролерів є MASM, NASM, FASM та GAS. В освітньому середовищі активно використовується NASM асемблер [9-11]. Це зумовлено безоплатними ліцензіями "Simplified BSD License" і "FreeBSD License", підтримкою різних платформ Windows, DOS, Linux, BSD, MAC OS та найновіших векторних інструкцій AVX, AVX2, AVX-512.

Таким чином, на даний час актуальним є використання векторних інструкцій AVX мікропроцесорів x86-64, що дозволить використати апаратний паралелізм в обробці даних та підвищити продуктивність програм, особливо для задач лінійної алгебри.

1 Аналіз сучасного стану та перспективи розвитку векторних обчислень

1.1 Еволюція архітектури процесорів: від SISD до SIMD

Класична архітектура фон Неймана передбачає послідовне виконання команд за принципом «одна інструкція - одиничні дані» (Single Instruction, Single Data, (SISD)). У такій моделі для додавання двох масивів з N елементів процесор змушений виконати N операцій завантаження, N операцій додавання та N операцій збереження результату. Це створює значне навантаження на блок декодування інструкцій та шину даних.

Для подолання цього бар'єру було розроблено технологію SIMD. Суть технології полягає в тому, що одна інструкція ініціює виконання однієї й тієї ж арифметичної дії над цілим вектором значень одночасно. Це дозволяє реалізувати паралелізм на рівні даних, а не на рівні операційної системи, що суттєво зменшує накладні витрати на перемикання контексту [12]. Приклади архітектур SISD та SIMD показано на на рис 1.

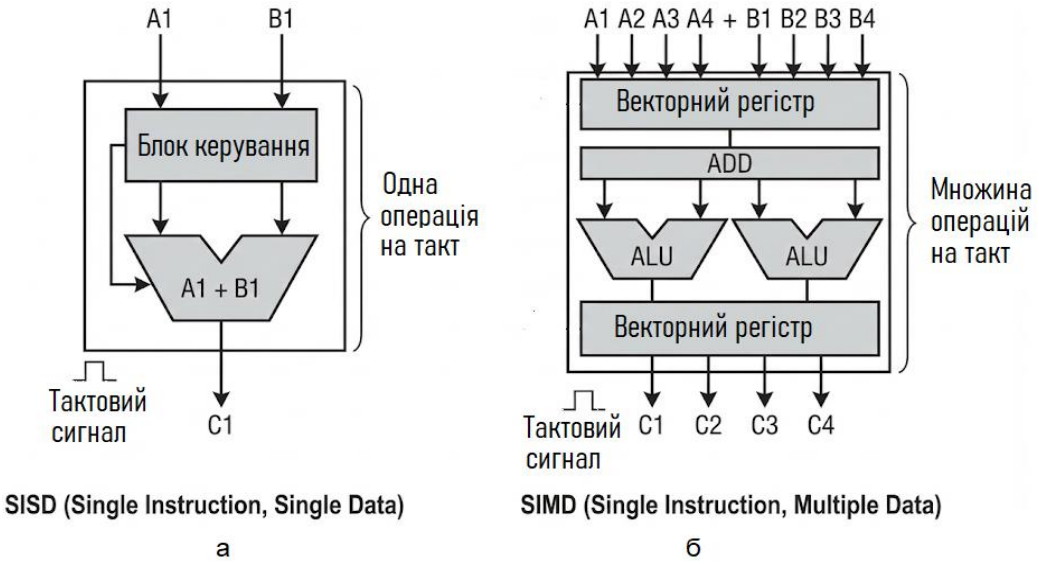


Рис. 1. Архітектури SISD та SIMD: а) додавання операндів (A1)+(B1);
б) додавання векторів (A1,A2,A3,A4) + (B1,B2,B3,B4)

1.2 Огляд розширень системи команд Intel x86-64

Еволюція векторних розширень архітектури x86-64 відбувалася поетапно, кожне нове покоління збільшувало розрядність регістрів та розширювало набір доступних інструкцій.

1. MMX (MultiMedia Extensions, 1997). Перша спроба впровадження SIMD. В MMX використовувалися 64-бітові регістри, які фізично накладалися на регістри співпроцесора (FPU), що унеможливлювало одночасну роботу з дійсними числами та MMX-командами.

2. SSE (Streaming SIMD Extensions, 1999). Наступний крок, що додав 128-бітові регістри XMM (XMM0–XMM15 у 64-бітному режимі). Це дозволило обробляти чотири 32-бітові числа з плаваючою крапкою (float) одночасно. Подальші версії (SSE2, SSE3, SSE4.1, SSE4.2) додали підтримку чисел подвійної точності (double) та розширили функціонал для роботи з цілими числами.

3. AVX (Advanced Vector Extensions, 2011). В AVX розширено регістри до 256 біт (YMM), що дозволило обробляти вже 8 чисел float або 4 числа double за такт. Важливою особливістю стала зміна синтаксису асемблерних команд з двоопераднього на триопарандний.

4. AVX2 (2013). Розширено можливості першої версії AVX, додавши повноцінну підтримку 256-бітових операцій над цілими числами та інструкції FMA (Fused Multiply-Add), які виконують множення та додавання за один крок без втрати точності.

5. AVX-512. Найсучасніше розширення з 512-бітовими регістрами ZMM, яке доступне переважно у серверних процесорах (Xeon) та високопродуктивних настільних комп'ютерах.

1.3 Архітектурні особливості AVX та регістровий файл YMM

Для виконання векторних операцій у наборі команд AVX використовуються регістри YMM. У 64-бітовому режимі роботи процесора (Long Mode) програмісту доступно 16 регістрів: YMM0 – YMM15.

Кожний регістр YMM (рис. 2) має довжину 256 біт (32 байти). Архітектурно молодші 128 біт регістру YMM відповідають регістру XMM (успадкованому від SSE). Це забезпечує зворотну сумісність: команд, що працюють з SSE, автоматично змінюють молодшу частину YMM, залишаючи старшу частину без змін (у деяких режимах) або обнуляючи її.

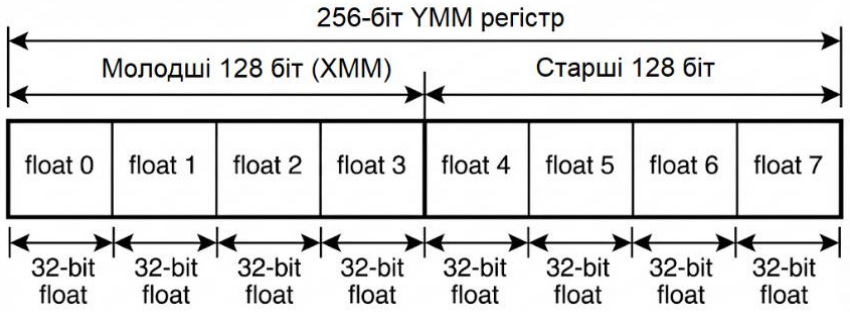


Рис. 2. Структура регістра YMM

YMM не має фіксованого типу, а підтримує запаковані і скалярні формати даних: 32×int8, 16×int16, 8×int32/float, 4×int64/double.

Важливою вимогою при роботі з AVX є вирівнювання даних у пам'яті. Для досягнення максимальної швидкодії адреси початку масивів повинні бути кратними 32 байтам. Використання невирівняних даних (інструкції типу vmovups) може призводити до штрафів продуктивності.

Можливо, найбільш примітним аспектом AVX є використання нового синтаксису команд. Більшість AVX команд використовують формат з двох операндів джерела та одного операнда призначення:

```
InstrMnemonic DesOp, SrcOp1, SrcOp2,
```

де `InstrMnemonic` означає мнемоніку AVX команд, а `DesOp`, `SrcOp1` і `SrcOp2` позначають операнди призначення та джерел відповідно. Майже всі операнди джерела є неруйнівними (тобто не змінюються під час виконання команди), за винятком випадків, коли регістр операнда призначення збігається з одним із регістрів операндів джерела.

Нові команди AVX поділені на наступні підгрупи:

- широкомовна трансляція (Broadcast);
- змішування (Blend);
- переставлення (Permute);
- видобування та вставлення (Extact and Insert);
- пересилання з маскою (Masked move);
- змінний бітовий зсув (Variale bit sift);
- збирання (Gather).

2 Алгоритмічне забезпечення векторних обчислень

2.1 Векторизація базових операцій

Перш ніж переходити до розв’язування задач лінійної алгебри, наприклад систем лінійних рівнянь, необхідно розробити ефективні алгоритми для базових операцій додавання векторів (рис. 3) та множення вектора на скаляр.

Нехай задано два вектори A та B розміром N:

$$A=\{ a_0 , a_1 , \dots , a_{n-1} \},\ B=\{ b_0 , b_1 , \dots , b_{n-1} \}.$$

Операція додавання векторів (C = A + B) у скалярному вигляді (SISD) описується формулою:

$$c_i=a_i+ b_i,\ \text{де } i=0, \dots, N\text{-}1. \quad (1)$$

При використанні розширення AVX з числами одинарної точності (float, 32 біти) у 256-бітовий регістр YMM поміщається 8 елементів. Це дозволяє змінити логіку циклу (1). Індекс i збільшується не на 1, а на 8 за кожний крок.

Тоді алгоритм векторного додавання буде наступним:

1. Завантажити 8 елементів масиву A у регістр `ymm1`.
2. Завантажити 8 елементів масиву B у регістр `ymm2`.
3. Виконати інструкцію векторного додавання `vaddps ymm0, ymm1, ymm2`.
4. Зберегти результат з `ymm0` у пам'ять масиву C.
5. Повторювати для всіх блоків по 8 чисел.

Якщо N не кратна 8, виникає проблема "залишків" масиву. Залишок елементів (від 1 до 7) необхідно обробляти класичним скалярним методом або використовувати маскування.

На рис. 3 показано схему векторного додавання. Вектори A і B розміщені у пам'яті і містять по вісім 32-бітових значень з плаваючою крапкою. Вектори A і B завантажуються у регістри YMM 1 і YMM2 та додаються командою `vaddps ymm0,ymm1,ymm2`. Результат з регістра YMM 0 записується у пам'ять.

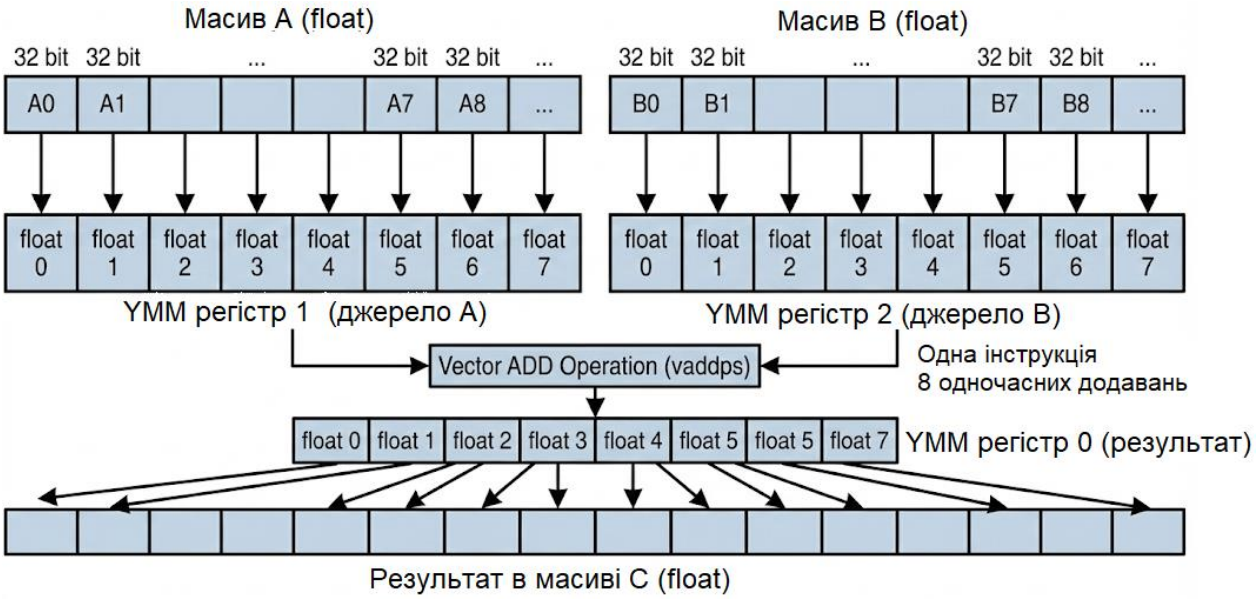


Рис. 3. Схема векторного додавання

2.2 Алгоритмічні особливості методу Жордана-Гауса

Однією з поширених задач лінійної алгебри є розв'язування систем лінійних алгебраїчних рівнянь (СЛАР). Метод Жордана-Гауса – це ефективний прямий алгоритм розв'язування СЛАР, який базується на зведенні розширеної матриці коефіцієнтів до діагонального (одиничного) вигляду за допомогою елементарних перетворень рядків [13-14]. Це вдосконалена версія методу Гаусса, яка відразу дає значення змінних без зворотної підстановки.

Нехай задано СЛАР у матричному вигляді $Ax = B$, де A – матриця коефіцієнтів $N \times N$, B – вектор вільних членів.

Основні етапи методу Жордана-Гауса:

1. Складання розширеної матриці $N\times(N+1)$, шляхом доповнення стовпцем вільних членів стовпці матриці коефіцієнтів.

2. На кожному кроці i вибирається ненульовий елемент a_{ii} (бажано максимальний за модулем), який стає ведучим.

3. Рядок, який містить ведучий елемент, ділиться на цей елемент, щоб зробити ведучий елемент рівним 1.

Алгоритм складається з N ітерацій. На кожному кроці k (де k – номер ведучого рядка) виконуються дві дії:

1.Нормалізація ведучого рядка. Усі елементи k-го рядка діляться на діагональний елемент a_{kk}:

$$a_{kj} \leftarrow \frac{a_{kj}}{a_{kk}}, j = k, \dots, N.$$

2. Обнулення змінних стовпця. Від усіх інших рядків i (де i != k) віднімається ведучий рядок, помножений на відповідний коефіцієнт a_{ik}:

$$a_{ij} \leftarrow a_{ij} - a_{ik} \cdot a_{kj}.$$

Саме друга дія є найбільш ресурсоємною, оскільки вона виконується для всіх рядків матриці. Однак вона ідеально підходить для векторизації. Коефіцієнт a_{ik} є сталим для всього i-го рядка під час однієї операції віднімання.

Послідовність оптимізації обчислень з використання векторних інструкцій AVX:

Для опрацювання i-го рядка:

1. Завантажити елементи поточного i-го рядка (a_{ij}) у вісім комірок регістра YMM.

2. Завантажити коефіцієнт a_{ik} (множник) у вісім комірок іншого регістра YMM інструкцією тиражування Broadcast (рис. 4).

3. Завантажити елементи ведучого рядка (a_{kj}) у вісім комірок іншого регістра YMM.

4. Виконати операція FMA (Fused Multiply-Add) для об’єднаного множення і додавання ($a\cdot b + c$ за один крок) одночасно для всіх 8 елементів поточного рядка.

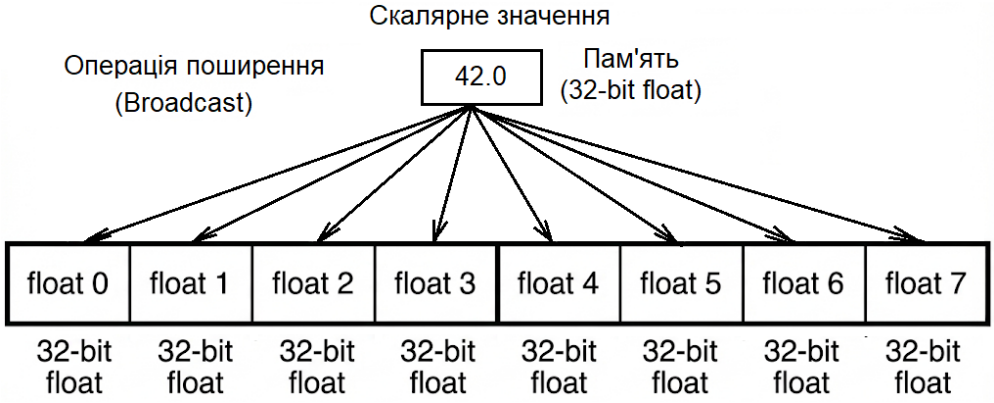


Рис. 4. Принцип роботи команди тиражування (Broadcast)

2.3 Бібліотека на асемблері з використанням інструкцій AVX/AVX2

Для реалізації алгоритму Жордана-Гауса була створена бібліотеки на асемблері NASM з наступними інструкціями процесора. Усі інструкції мають префікс *v*, що дозволяє використовувати три операнди.

1. Інструкції пересилання даних:

• `vmovups ymm1, [mem]` (Move Unaligned Packed Single) – завантаження 256 біт (8 float) з пам'яті в регістр. Використовується версія ups (unaligned), оскільки вона безпечніша, якщо дані не вирівняні на межі 32 байт.

• `vmovups [mem], ymm1` – збереження даних з регістра в пам'ять.

• `vbroadcastss ymm1, [mem]` (Broadcast Scalar Single) – ключова команда для методу Гауса. Вона бере одне 32-бітове число з пам'яті та заповнює ним увесь 256-бітовий регістр.

2. Арифметичні команди:

• `vaddps ymm1, ymm2, ymm3` – додавання запакованих чисел (`ymm1 = ymm2 + ymm3`).

• `vmulps ymm1, ymm2, ymm3` – множення запакованих чисел.

• `vsubps ymm1, ymm2, ymm3` – віднімання запакованих чисел.

• `vdivps ymm1, ymm2, ymm3` – ділення запакованих чисел (використовується при нормалізації рядка).

3. Команди FMA (Fused Multiply-Add) – для AVX2:

• `vfmsub213ps ymm1, ymm2, ymm3` – дія команди `ymm1 = (ymm2 * ymm1) - ymm3`.

Використання FMA дозволяє підвищити точність, оскільки проміжний результат множення не округляється, а зберігається з повною точністю до моменту віднімання.

2.4 Структура даних і алгоритми з використанням векторних операцій

Матриці в пам'яті комп'ютера подані як одновимірний масив, де елементи зберігаються рядками. Доступ до елемента $A[i][j]$ здійснюється за індексом $i\times(N+1)$, де $N+1$ – ширина розширеної матриці.

Алгоритм векторного множення:

1. Початок.
2. Перевірка, чи кількість елементів $N \geq 8$.
3. Якщо так:
   - завантажити 8 чисел з масиву A в YMM0;
   - завантажити 8 чисел з масиву B в YMM1;
   - виконати команду `vmulps ymm0, ymm0, ymm1`;
   - зберегти YMM0;
   - збільшити покажчик на 8 елементів (32 байти);
   - повторити цикл.
4. Якщо ні (або є залишок):
   - обробити елементи по одному класичними командами SSE (скалярно).
5. Кінець.

Блок схема методу Жордана-Гауса показана на рис. 5.

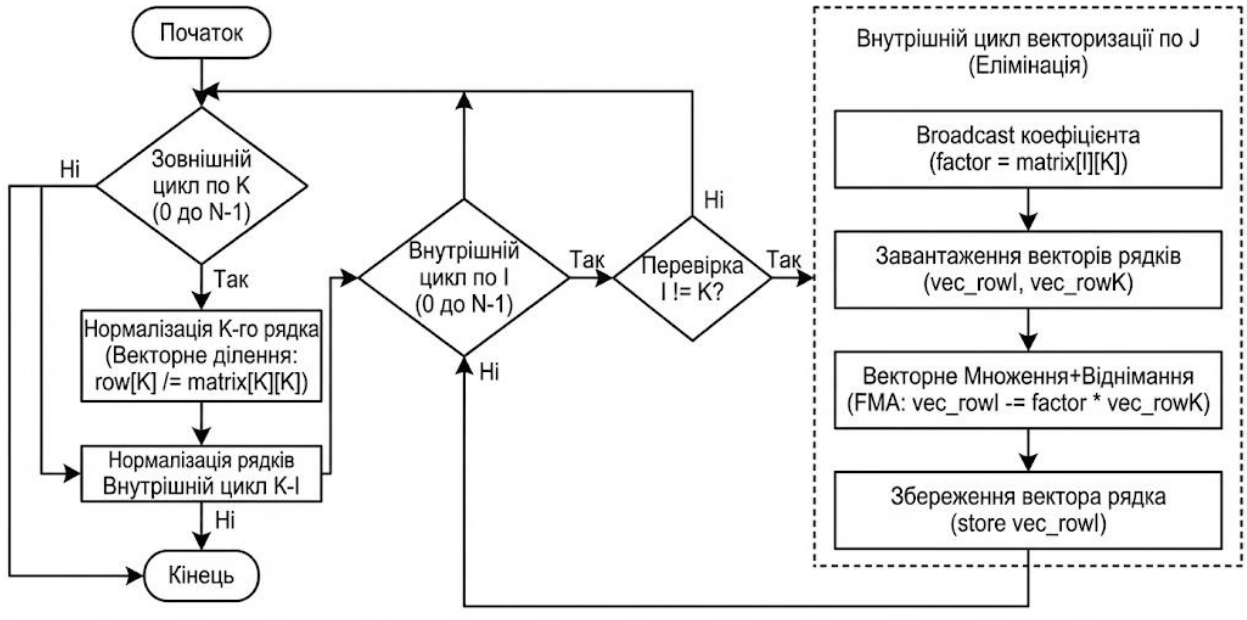


Рис. 5. Блок-схема методу Жордана-Гауса

Алгоритм методу Жордана-Гауса:

1. Зовнішній цикл по *K* (від 0 до *N*-1).
2. Нормалізація *K*-го рядка (векторне ділення).
3. Внутрішній цикл по *I* (від 0 до *N*-1).
4. Перевірка *I* != *K*.
5. Внутрішній цикл векторизації по *J*: Тиражування (Broadcast) коефіцієнта → Завантаження рядка → Множення + Віднімання → Збереження.

3 Реалізація програм та дослідження ефективності бібліотеки

3.1 Структура програм та організація даних

Прийнято векторизовані функції асемблера об'єднувати у бібліотеки, що дозволяє їх використання у мовах програмування високого рівня [15, 16]. Розроблені функції з використанням векторизованих інструкцій AVX об'єднано у бібліотеку на мові асемблера NASM для 64-бітової архітектури Windows (Win64 домовленості викликів). Програмний код структуровано на кілька логічних модулів:

1. Секція даних (`.data` та `.bss`): Відповідає за зберігання глобальних змінних та констант. Ключовим моментом є використання директиви вирівнювання `align 32`. Це критично необхідно для інструкцій AVX, оскільки доступ до невирівняної пам'яті може спричинити виняткову ситуацію (crash) або суттєве падіння продуктивності.
2. Секція коду (`.text`): Містить процедури `VectorAdd`, `VectorMul` та `GaussJordanSolve`.
3. Макроси: Використано макроси препроцесора NASM для стандартизації прологу та епілогу функцій (збереження регістрів `rbp`, `rsp`, `rsi`, `rdi`), що спрощує читальність коду.

Для взаємодії з операційною системою та виведення результатів використано функції Windows API (`WriteConsoleA`, `ExitProcess`) та функції стандартної бібліотеки C (`printf`), підключені через компонувальник.

3.2 Реалізація базових векторних операцій

Нижче показано фрагмент розробленої процедури додавання векторів. Основний цикл обробляє по 8 елементів типу float за ітерацію.

Роздрук 1 – Реалізація векторного додавання (фрагмент)

```
; Вхідні параметри (Win64 convention):
; rcx - покажчик на масив A
; rdx - покажчик на масив B
; r8  - покажчик на масив Result
; r9  - кількість елементів N

global VectorAdd_AVX
VectorAdd_AVX:
    push rbp
    mov rbp, rsp

    ; Основний цикл (обробка по 8 чисел)
    xor rax, rax            ; Обнулення лічильника
.loop_block:
    cmp rax, r9             ; Перевірка кінця масиву
    jge .cleanup            ; Якщо пройшли весь масив - вихід

    ; Завантаження даних (використовується unaligned для безпеки)
    vmovups ymm0, [rcx + rax*4] ; Завантаження 8 float з A
    vmovups ymm1, [rdx + rax*4] ; Завантаження 8 float з B

    ; Векторне додавання
    vaddps ymm0, ymm0, ymm1     ; YMM0 = A + B

    ; Збереження результату
```

```
    vmovups [r8 + rax*4], ymm0

    add rax, 8              ; Крок циклу +8 елементів
    jmp .loop_block

.cleanup:
    ; Тут розміщується код обробки "залишку" масиву (якщо N не кратне 8)
    ; ... (реалізовано через скалярні команди xmm - addss)

    mov rsp, rbp
    pop rbp
    ret
```

3.3 Реалізація ядра методу Жордана-Гауса

Найбільш складним є векторизація внутрішнього циклу, де відбувається віднімання рядків. Для оптимізації використано команду `vbroadcastss`, яка дублює поточний коефіцієнт $K = a_{ik}$ на весь регістр. Це дозволяє уникнути циклічного звернення до пам'яті за одним і тим самим значенням.

Роздрук 2 – Векторизація віднімання рядків

```
; YMM2 містить тиражований коефіцієнт (multiplier)
    ; RDI вказує на поточний рядок (Target Row)
    ; RSI вказує на ведучий рядок  (Pivot Row)

.inner_loop_simd:
    ; Завантаження 8-ми елементів ведучого рядка
    vmovups ymm0, [rsi + rbx*4]

    ; Завантаження 8 елементів рядка, що змінюється
    vmovups ymm1, [rdi + rbx*4]

    ; Обчислення: Row[j] = Row[j] - Multiplier * Pivot[j]
    vmulps ymm0, ymm0, ymm2     ; YMM0 = Pivot * Multiplier
    vsubps ymm1, ymm1, ymm0     ; YMM1 = Target - (Pivot * Mult)

    ; Збереження результату назад у пам'ять
    vmovups [rdi + rbx*4], ymm1

    add rbx, 8                  ; Наступні 8 стовпців
    cmp rbx, r10                ; Перевірка на кінець рядка
    jl .inner_loop_simd
```

3.4 Методика тестування та характеристики тестового стенду

Для оцінки ефективності розроблених алгоритмів проведено серію експериментів. Вимірювання часу виконання здійснювалося за допомогою процесорної інструкції `rdtsc` (Read Time-Stamp Counter), яка повертає кількість тактів процесора з моменту скидання. Це забезпечує найвищу можливу точність вимірювань, нівелюючи похибки системного таймера ОС.

Конфігурація апаратного та програмного забезпечення, на якому проводилося тестування, показана в табл. 1.

3.5 Аналіз результатів експериментів

Було проведено порівняльне тестування двох версій алгоритму розв'язування СЛАР:

1. SISD (Scalar): класична реалізація з використанням одного регістра FPU/XMM для одного числа.

2. SIMD (AVX): оптимізована реалізація з використанням 256-бітових регістрів YMM (8 чисел одночасно).

Тестування проводилося на випадково згенерованих квадратних матрицях розмірністю від 256×256 до 2048×2048. Для кожної розмірності тест запускався 10 разів, після чого

обчислювалося середнє арифметичне значення часу виконання (у мільйонах тактів процесора).

Таблиця 1

Параметр	Значення
Центральний процесор (CPU)	Intel Core i5-11400H @ 2.70GHz
Архітектура	x86-64 (Rocket Lake)
Підтримка інструкцій	MMX, SSE4.2, AVX, AVX2, FMA3
Оперативна пам'ять (RAM)	16 GB DDR4-3200
Операційна система	Windows 11 Home (64-bit)
Середовище розробки	VS Code + NASM 2.15
Режим компіляції	Release (без налагоджувальної інформації)

Результати вимірювань (у тактах процесора) показано в табл. 2.

Таблиця 2

Розмірність матриці (N)	Час виконання SISD (10^6 тактів)	Час виконання AVX, (10^6 тактів)	Прискорення, разів
256	45.2	8.8	5.14
512	385.6	68.9	5.60
1024	3120.5	510.4	6.11
2048	25890.1	4050.8	6.39

Як видно з табл. 2, використання векторних інструкцій забезпечує суттєве прискорення обчислень. Теоретичний максимум прискорення для AVX (обробка 8 чисел float) становить 8 разів. На практиці отримано коефіцієнт у межах 5.1 – 6.4 разів.

Відхилення від теоретичного максимуму показано на рис. 6 і пояснюється такими факторами:

1. Пропускна здатність пам'яті: процесор обчислює дані швидше, ніж встигає отримувати їх з оперативної пам'яті.

2. Накладні витрати: частина часу витрачається на підготовку циклів, завантаження коефіцієнтів та обробку "залишків" масивів, які виконуються скалярно.

3. Кеш-промахи: при великих розмірностях матриць (2048 і більше) дані перестають вміщатися у L2/L3 кеш процесора, що збільшує затримки доступу до даних.

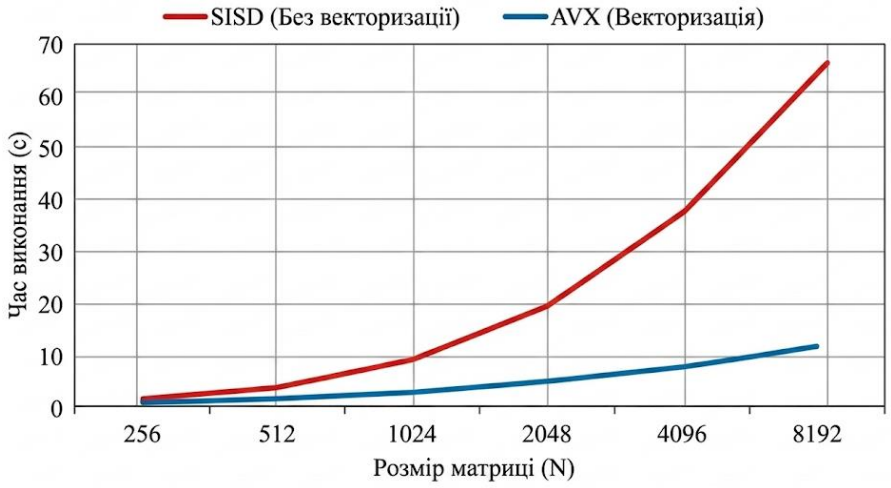


Рис. 6. Залежності часу виконання від розмірів матриць

Висновки

У статті розглянуто актуальну науково-практичну задачу розв'язування систем лінійних алгебраїчних рівнянь з використанням мови асемблера і векторних інструкцій AVX. Показано, що скалярна модель обчислень SISD вичерпала можливості екстенсивного росту продуктивності. Перехід до векторної моделі обчислень SIMD є ключовим фактором для прискорення обробки великих масивів даних. Аналіз системи інструкцій AVX показав, що використання триоперандного синтаксису дозволяє прискорити виконання арифметичних операцій і оптимізувати використання регістрового файлу.

Розроблено алгоритми для базових векторних операцій (додавання, множення) та векторизованого методу Жордана-Гауса. Створено бібліотеку функцій на мові асемблера NASM з використанням інструкцій AVX. Реалізовано метод Жордана-Гауса з використанням розроблених бібліотечних функцій.

Проведено експериментальне дослідження на тестових матрицях розмірністю від 256×256 до 2048×2048. Отримано прискорення обчислень від 5 до 6 разів порівняно з класичною реалізацією. Отримані результати показують доцільність використання асемблерних функцій з векторними інструкціями AVX/AVX2 у критичних до швидкодії ділянках коду.

Список літератури:

1. C. Xie, H. Wu, and J. Zhou, “Vectorization Programming Based on HR DSP Using SIMD,” *Electronics*, vol. 12, no. 13, p. 2922, Jul. 2023. doi: https://doi.org/10.3390/electronics12132922.
2. Intel® Intrinsics Guide: Instruction set. URL: https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html?utm_source=chatgpt.com#.
3. Kusswurm D. Modern Parallel Programming with C++ and Assembly Language: X86 SIMD Development Using AVX, AVX2, and AVX-512 / D. Kusswurm. New York : Apress, 2022. 580 p. URL: https://www.oreilly.com/library/view/modern-parallel-programming/9781484279182/.
4. M. Costanzo, E. Rucci, M. Naiouf, and A. D. Giusti, “Performance vs Programming Effort between Rust and C on Multicore Architectures: Case Study in N-Body,” in *2021 XLVII Latin Amer. Comput. Conf. (CLEI)*, Cartago, Costa Rica, Oct. 25–29, 2021. IEEE, 2021. doi: https://doi.org/10.1109/clei53233.2021.9640225.
5. P. E. de Vilhena, O. Lahav, V. Vafeiadis, and A. Raad, “Extending the C/C++ Memory Model with Inline Assembly,” *Proc. ACM Program. Lang.*, vol. 8, OOPSLA2, pp. 1081–1107, Oct. 2024. doi: https://doi.org/10.1145/3689749.
6. M. Rigger, S. Marr, S. Kell, D. Leopoldseder, and H. Mössenböck, “An Analysis of x86-64 Inline Assembly in C Programs,” in *VEE '18: 14th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ.*, Williamsburg VA USA. New York, NY, USA: ACM, 2018. doi: https://doi.org/10.1145/3186411.3186418.
7. K. Papagiannopoulos, “Low Randomness Masking and Shuffling: An Evaluation Using Mutual Information,” *IACR Trans. Cryptographic Hardware Embedded Syst.*, pp. 524–546, Aug. 2018. doi: https://doi.org/10.46586/tches.v2018.i3.524-546.
8. D. Salomon and I. Levi, “MaskSIMD-lib: on the performance gap of a generic C optimized assembly and wide vector extensions for masked software with an Ascon-p test case,” *J. Cryptographic Eng.*, May 2023. doi: https://doi.org/10.1007/s13389-023-00322-4.
9. NASM - The Netwide Assembler. URL: https://www.nasm.us.
10. “Microsoft Macro Assembler reference.” Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/uk-ua/cpp/assembler/masm/microsoft-macro-assembler-reference?view=msvc-170.
11. GAS - the GNU assembler. URL: https://www.gnu.org/software/binutils/.
12. B. Adeleye and S. M. Jiddah, “Analysis of Parallel Architectures: SIMD, tightly-coupled MIMD, and loosely-coupled MIMD,” *Int. J. Comput. Trends Technol.*, vol. 53, no. 1, pp. 6–8, Nov. 2017. doi: https://doi.org/10.14445/22312803/ijctt-v53p102.
13. L. Bobrowski and C. Boldak, “Stepwise Inversion of Large Matrices with the Gauss-Jordan Vector Transformation,” *J. Advances Math. Comput. Sci.*, pp. 28–39, Jan. 2022. doi: https://doi.org/10.9734/jamcs/2022/v37i130429.
14. P. S. Stanimirović and M. D. Petković, “Gauss–Jordan elimination method for computing outer inverses,” *Appl. Math. Computation*, vol. 219, no. 9, pp. 4667–4679, Jan. 2013. doi: https://doi.org/10.1016/j.amc.2012.10.081.
15. K. Goto and R. A. v. d. Geijn, “Anatomy of high-performance matrix multiplication,” *ACM Trans. Math. Softw.*, vol. 34, no. 3, pp. 1–25, May 2008. doi: https://doi.org/10.1145/1356052.1356053

16.T. B. Schardl, W. S. Moses, and C. E. Leiserson, “Tapir,” in *PPoPP '17: 22nd ACM SIGPLAN Symp. Princ. Pract. Parallel Program.*, Austin Texas USA. New York, NY, USA: ACM, 2017. doi: https://doi.org/10.1145/3018743.3018758.

Надійшла до редколегії 10.03.2025

Відомості про авторів:

Голота Віктор Іванович - кандидат технічних наук, доцент кафедри комп’ютерної інженерії та електроніки, Прикарпатський національний університети імені Василя Стефаника / Vasyl Stefanyk Precarpathian National University, Україна; email: viktor.holota@pnu.edu.ua, ORCID: https://orcid.org/0000-0003-4605-7507.

Грига Володимир Михайлович – канд. техн. наук, доцент, доцент кафедри комп’ютерної інженерії та електроніки, Прикарпатський національний університети імені Василя Стефаника / Vasyl Stefanyk Precarpathian National University, Україна; email: volodymyr.hryha@pnu.edu.ua; ORCID: https://orcid.org/0000-0001-5458-525X

Бенько Тарас Григорович – асистент кафедри комп’ютерної інженерії та електроніки, Прикарпатський національний університет імені Василя Стефаника, Україна / Vasyl Stefanyk Precarpathian National University, Україна; email: taras.benko@pnu.edu.ua; ORCID: https://orcid.org/0000-0002-6310-8743

Моргун Анатолій Володимирович – аспірант кафедри комп’ютерної інженерії та електроніки, Прикарпатський національний університети імені Василя Стефаника / Vasyl Stefanyk Precarpathian National University, Україна; email: morgun.anatoliy@gmail.com; ORCID: https://orcid.org/0009-0002-4765-2885

УДК 004.43:004.415.5 **DOI:10.15330/itee.2025.2.05**

Р. І. ЗАПУХЛЯК, канд. фіз.-мат., наук, В. В. ДОВГИЙ, канд. техн. наук

БЕЗПЕКА ПАМ'ЯТІ У C++: ІНІЦІАТИВИ WG21 (LIFETIME, CONTRACTS, SAFETY PROFILES) ТА ЕМПІРИЧНИЙ АНАЛІЗ OPEN-SOURCE ПРОЄКТІВ

Вступ

C++ залишається ключовою мовою системного програмування завдяки високій продуктивності, контролю над ресурсами та багатим абстракціям. Водночас саме доступність низькорівневих механізмів (вказівники, арифметика адрес, неініціалізована пам'ять, ручне керування буферами) зумовлює ризики порушення memory safety. Уразливості пам'яті лежать в основі значної частини інцидентів безпеки у великих програмних системах, а undefined behavior (UB) [2] (UB) у C/C++ створює додатковий клас помилок: програма може демонструвати коректну поведінку в одних умовах і бути вразливою або некоректною в інших.

Останніми роками в межах WG21 [2] активізовано роботу над «безпечнішою C++» як поєднанням: (1) профілів та контрактів, що дозволяють підвищувати рівень перевірок; (2) формалізації lifetime-аналізу (зокрема на основі Core Guidelines Lifetime profile [3]); (3) практик та інструментів (компіляторні попередження, статичний аналіз, санітайзери [1]). Мета цієї роботи — надати огляд ініціатив WG21 [2] та показати відтворюваний експериментальний зріз для трьох репрезентативних open-source бібліотек.

Завдання дослідження:

1) систематизувати напрями Lifetime Safety [3, 8], Contracts [5], Safety Profiles [4, 6, 7];

2) розробити протокол експерименту (g++ + санітайзери [1]) та отримати вимірювані результати;

3) інтерпретувати результати у контексті ініціатив WG21 [2] та обмежень експерименту.

Дослідження має характер відтворюваного case study та не претендує на статистично репрезентативну оцінку всієї екосистеми C++

Класи помилок і роль undefined behavior (UB)

Під memory safety у контексті C++ зазвичай розуміють запобігання або виявлення принаймні таких класів помилок: out-of-bounds доступи (читання/запис поза межами буфера), use-after-free (доступ після звільнення), double free, dangling references/iterators, некоректне перетворення типів та порушення вирівнювання. Значна частина таких ситуацій класифікується стандартом як UB, що надає компілятору свободу оптимізацій. Як наслідок, UB може мати неочевидні прояви, включно з видаленням «захисних» перевірок під час оптимізації.

RAII та інваріанти часу життя

RAII (Resource Acquisition Is Initialization) є фундаментальним механізмом C++ для керування ресурсами: ресурс набувається під час ініціалізації об'єкта і звільняється в деструкторі. Однак RAII не усуває потребу у правильному керуванні посиланнями/вказівниками на об'єкти, час життя яких може завершитися раніше за час життя посилання. Саме цей розрив (lifetime mismatch) породжує dangling references та use-after-free. Статичні правила (напр., Core Guidelines Lifetime profile [3]) та контракти можуть формалізувати інваріанти та зробити їх перевірку системнішою.

Інструментальний рівень: санітайзери

Санітайзери - це інструменти, що додають інструментування до коду під час компіляції. AddressSanitizer (ASan) [1] виявляє широке коло помилок пам'яті (out-of-bounds, use-after-free тощо) і широко застосовується в індустрії та open-source. UBSan спрямований на виявлення

виконуваних проявів UB (переповнення, некоректні зсуви, порушення вирівнювання, некоректні перетворення типів тощо). У роботі санітайзери [1] застосовано як практичний критерій «виявлення» проблем у тестових сценаріях.

Ініціативи WG21: Lifetime, Contracts, Safety Profiles

Під Lifetime Safety [3, 8] у рамках WG21 [2]/SG23 часто мають на увазі розвиток підходів, що дозволяють компілятору/аналізатору виявляти типові випадки dangling (use-after-free) на основі локального аналізу, не вимагаючи повноцінного borrow-checker на кшталт Rust. Репрезентативним документом є опис Core Guidelines Lifetime profile [3], де формалізовано набір правил і підхід до діагностики типових помилок часу життя (зокрема через анотації/атрибути та аналіз повернених значень).

Контракти дозволяють задавати преумови, постумови та твердження (assertions) як частину інтерфейсу. Це дає можливість: (1) документувати інваріанти; (2) виконувати перевірки з різними режимами (від вимкнених до жорстких); (3) інтегрувати контракти з профілями безпеки та hardening стандартної бібліотеки. У контексті memory safety контракти корисні для позначення припущень щодо розмірів, непорожності, допустимих діапазонів, валідності вказівників/ітераторів, що зменшує ймовірність UB.

Safety Profiles [4, 6, 7] розглядаються як механізм «конфігурованих» обмежень і перевірок, що можуть застосовуватися до коду без повної зміни мови. Ідея полягає у визначенні профілів (напр., lifetime/type профілі) як сукупності заборон/вимог та способів їх перевірки (компілятор, бібліотека, статичний аналіз, санітайзери [1]). Профілі можуть бути адаптовані до домену (embedded, safety-critical тощо) і співіснувати з існуючим C++ кодом.

Методологія експерименту

Об’єктами дослідження обрано три популярні C++ бібліотеки різного масштабу: fmt (форматування), spdlog (логування) та nlohmann/json (JSON). Мета — отримати відтворювані метрики на реальному коді: (1) кількість попереджень компілятора у конфігурації «warnings-lite»; (2) кількість унікальних спрацювань ASan під час виконання тестів; (3) кількість унікальних спрацювань UBSan (рядки `runtime error:`) під час тестів. Додатково фіксуються commit-ідентифікатори та LOC (cloc).

Середовище виконання

Опис середовища виконання наведено в табл. 1.

Таблиця 1.

Характеристика середовища виконання

Параметр	Значення
OS	Ubuntu (Linux kernel 6.8.0-101-generic, x86_64)
CPU	16 logical cores (nproc=16)
Compiler	g++ (Ubuntu 13.3.0-6ubuntu2~24.04.1) 13.3.0
CMake	3.28.3
Make	GNU Make 4.3

Протокол збірки та тестування (g++)

Для забезпечення відтворюваності застосовано три незалежні конфігурації збірки для кожного проєкту. Усі команди виконувалися в окремих директоріях збірки. Паралелізм складання: `make -j4` (стабільний режим), тести запускалися через `ctest --output-on-failure`.

Конфігурація A: попередження (warnings-lite):

```
cmake -DCMAKE_CXX_COMPILER=g++ -DCMAKE_BUILD_TYPE=Debug \
  -DCMAKE_CXX_FLAGS="-Wall -Wextra -Wpedantic -Wshadow -Wconversion -Wsign-conversion -
Wnull-dereference -Wdouble-promotion -Wformat=2 -Wduplicated-cond -Wduplicated-branches
-Wlogical-op -Wuseless-cast" ..
make -j4 > warnings_build_lite.log 2>&1
grep -E "warning:|warning\(" warnings_build_lite.log | wc -l
Конфігурація B: AddressSanitizer (ASan) [1].
cmake -DCMAKE_CXX_COMPILER=g++ -DCMAKE_BUILD_TYPE=Debug \
  -DCMAKE_CXX_FLAGS="-fsanitize=address -g -fno-omit-frame-pointer" \
  -DCMAKE_EXE_LINKER_FLAGS="-fsanitize=address" ..
make -j4 > asan_build.log 2>&1
ctest --output-on-failure 2>&1 | tee asan_ctest.log
grep -oE "SUMMARY: AddressSanitizer: [A-Za-z0-9_-]+" asan_ctest.log | sort | uniq -c
Конфігурація C: ASan + UBSan.
cmake -DCMAKE_CXX_COMPILER=g++ -DCMAKE_BUILD_TYPE=Debug \
  -DCMAKE_CXX_FLAGS="-fsanitize=address,undefined -g -fno-omit-frame-pointer" \
  -DCMAKE_EXE_LINKER_FLAGS="-fsanitize=address,undefined" ..
make -j4 > asan_ubsan_build.log 2>&1
ctest --output-on-failure 2>&1 | tee asan_ubsan_ctest.log
grep -oE "runtime error: [^:]*" asan_ubsan_ctest.log | sort | uniq -c
```

Для порівняння проєктів різного розміру використано нормалізацію на 10 000 рядків коду (LOC):

Отже, основні результати наведені в табл. 2.

Таблиця 2.

Результати виконання експерименту

Проєкт	Commit	LOC	Попередження	ASan (унік.)	UBSan (унік.)
fmt	82553a7a5b78e446f7f559d7b968f606652ee5d2	18740	18	1	0
spdlog	0f7562a0f9273cfc71fddc6ae52ebff7a490fa04	3636	0	0	0
json (nlohmann/ json)	8167d2f64122c3ef7e5a0d9cab239fdba37aa54c	48347	0	0	0

Нормалізовані метрики експерименту продемонстровано в табл. 3.

Таблиця 3.

Нормалізовані метрики

Проєкт	warnings / 10k LOC	ASan / 10k LOC	UBSan / 10k LOC
fmt	9.61	0.53	0.00
spdlog	0.00	0.00	0.00
json (nlohmann/json)	0.00	0.00	0.00

На рисунках 1-4 продемонстровано діаграми результатів при експериментів та виявлення різних чинників.

Під час запуску тестового набору fmt в конфігурації ASan/ASan+UBSan зафіксовано один унікальний інцидент: `SUMMARY: AddressSanitizer: allocation-size-too-big`. Локалізація вказувала на тест `util_test.format_system_error` (файл `fmt/test/format-test.cc`, рядок 278). ASan перервав виконання, оскільки запитаний розмір виділення пам’яті перевищував максимальний підтримуваний ліміт. Важливо, що це не обов’язково вказує на експлуатовану уразливість у бібліотеці як такій – у тестах можуть бути негативні сценарії та перевірки граничних умов. Однак сам факт виявлення некоректного запиту демонструє корисність санітайзерів як інструменту для раннього виявлення помилок керування ресурсами або некоректної обробки даних.

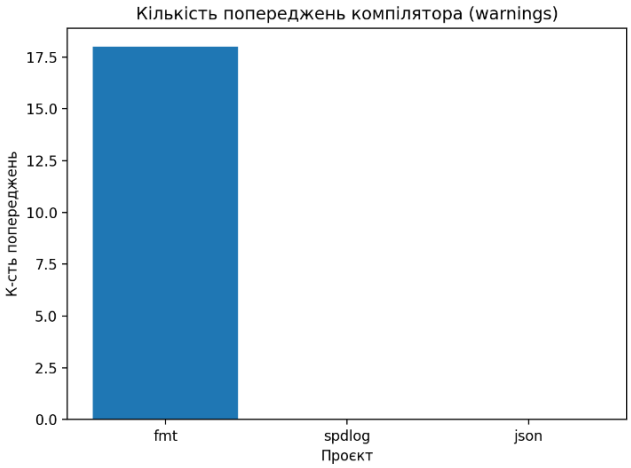


Рис. 1. Кількість попереджень компілятора (warnings)

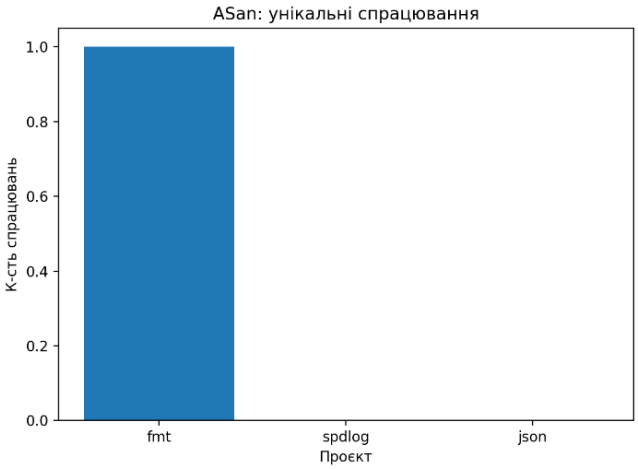


Рис. 2. Унікальні спрацювання ASan

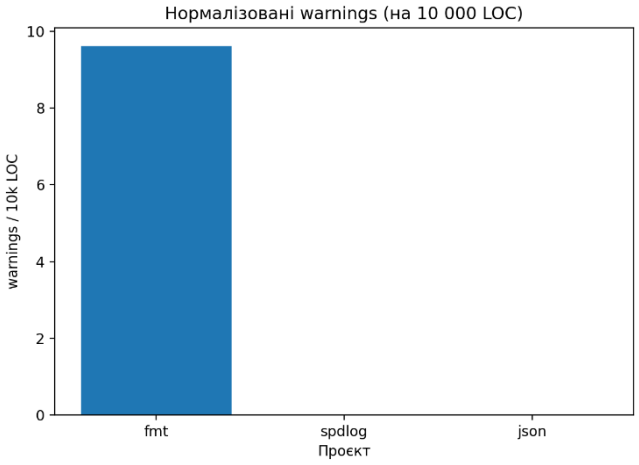


Рис. 3. Нормалізовані warnings (на 10 000 LOC)

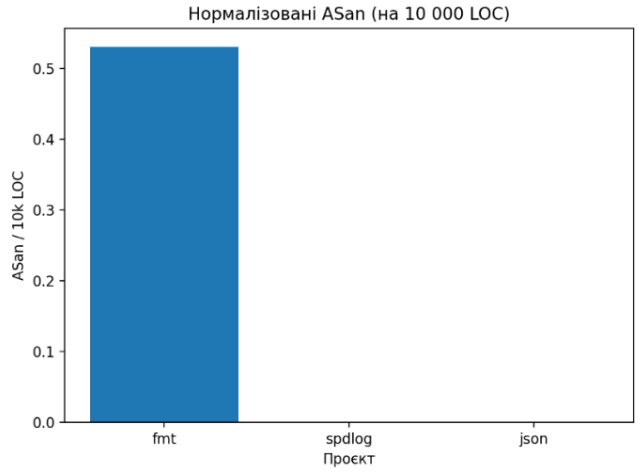


Рис. 4. Нормалізовані warnings (на 10 000 LOC)

Обґрунтування результатів у контексті ініціатив WG21

У конфігурації warnings-lite попередження з'явилися лише в fmt (18), тоді як spdlog та nlohmann/json зібралися без попереджень у заданому наборі прапорів. Це ілюструє, що «чистота» щодо попереджень є властивістю не лише якості коду, а й стилю проєкту, специфіки тестів та компіляторних налаштувань. Нормалізований показник для fmt становить 9,61 warnings/10k LOC.

ASan виявив один унікальний інцидент у fmt та не виявив спрацювань у spdlog і nlohmann/json. UBSan не зафіксував жодного `runtime error:` у тестових сценаріях усіх трьох проєктів. Це може означати: (1) відсутність UB у покритих тестами шляхах; (2) наявність UB, яке не активувалося в межах тестів; (3) UB, що проявляється лише під іншими компіляторами/прапорцями/платформами. Таким чином, результати слід інтерпретувати як «виявлено за умов конкретного експерименту», а не як абсолютний доказ відсутності UB.

Відповідність класів проблем напрямам WG21

Фіксація інциденту, пов'язаного з некоректним виділенням пам'яті, корелює з мотивацією напрямів WG21 [2]: lifetime/type профілі та контракти націлені на те, щоб або забороняти небезпечні патерни, або робити їх перевірюваними та керованими. Зокрема:

• Lifetime Safety [3, 8] може зменшувати ризики dangling/use-after-free через аналіз походження посилань та повернених значень;

• Contracts [5] дозволяють виражати передумови (наприклад, допустимі розміри, діапазони, валідність аргументів) і керувати семантикою перевірок;

• Safety Profiles [4, 6, 7] надають механізм оголошення та застосування наборів обмежень (наприклад, заборона reinterpret_cast або обмеження сирих вказівників) відповідно до доменних вимог.

Обмеження експерименту:

1) Вибірка з трьох бібліотек не репрезентує всю екосистему C++.

2) Виявлення санітайзерів залежить від тестового покриття та умов виконання.

3) У тестових наборах можуть бути «негативні» тести (наприклад, assert-test у fmt), що навмисно призводять до аварійної зупинки.

4) Результати отримано для g++ 13.3; інші компілятори або налаштування можуть дати інші спрацювання.

5) Показники warnings залежать від набору прапорів і політики попереджень у проєкті.

Порівняння з попередніми емпіричними дослідженнями

Отримані в роботі результати демонструють низьку частоту runtime-виявлень (ASan/UBSan) у трьох зрілих open-source бібліотеках за умов виконання штатних тестових наборів. Такі спостереження узгоджуються з висновками низки попередніх емпіричних досліджень, присвячених проблематиці undefined behavior (UB) та безпеки пам’яті в мовах сімейства C/C++.

Зокрема, у роботі Yang та ін. [11] показано, що навіть за наявності формально коректного коду компіляторні оптимізації можуть індукувати некоректну поведінку за умов UB, що ускладнює його виявлення. Це підкреслює, що відсутність runtime-спрацювань у конкретному експерименті не еквівалентна відсутності потенційно небезпечних конструкцій у кодовій базі. Натомість це свідчить про те, що в межах протестованих сценаріїв такі конструкції або відсутні, або не активуються.

Дослідження, присвячені аналізу вразливостей у прикладах C/C++-коду з відкритих джерел [12], демонструють, що значна частина memory-related помилок виникає у фрагментах з недостатнім тестуванням або у навчальних/демонстраційних прикладах. На цьому тлі низька кількість спрацювань у бібліотеках fmt, spdlog та nlohmann/json може бути пояснена високим рівнем зрілості цих проєктів, наявністю розвинених CI-пайплайнів та регулярного використання інструментів статичного й динамічного аналізу.

У роботах, присвячених Rust та memory-safe мовам [13], підкреслюється роль статичних механізмів перевірки часу життя та власності як способу радикального зменшення класів помилок use-after-free і dangling references. У цьому контексті ініціативи WG21 щодо Lifetime Safety [3, 8] та Safety Profiles [4, 6, 7] можна розглядати як еволюційний підхід до часткової імплементації подібних гарантій у межах існуючої парадигми C++. Отримані експериментальні результати опосередковано підтверджують доцільність такого поетапного підсилення гарантій: навіть без повної зміни мовної моделі поєднання тестування та санітайзерів дозволяє знизити прояви memory-related дефектів у виконуваних сценаріях.

Огляд сучасних технік забезпечення memory integrity для мов без повної статичної безпеки пам’яті [14] також наголошує на багаторівневому характері захисту: поєднанні мовних обмежень, профілів безпеки, компіляторних перевірок та runtime-інструментування. Результати цього дослідження узгоджуються з таким багаторівневим підходом: використання ASan/UBSan у поєднанні з дисципліною розробки та тестуванням у зрілих бібліотеках демонструє практичну ефективність інструментального рівня, тоді як ініціативи WG21 можуть надати додаткові статичні гарантії на рівні мови та стандартної бібліотеки.

Таким чином, результати проведеного case study не суперечать попереднім емпіричним спостереженням, а доповнюють їх у прикладному аспекті, демонструючи реальний стан

memory-safety у конкретних популярних C++-проєктах за умов сучасних інструментів аналізу. Водночас вони підтверджують, що для повнішої картини необхідні ширші вибірки, різні компіляторні конфігурації та додаткові методи аналізу, що визначає напрями подальших досліджень.

Наукова новизна роботи полягає у поєднанні систематизованого аналізу сучасних ініціатив WG21 у сфері підвищення безпеки C++ (Lifetime Safety, Contracts, Safety Profiles) з відтворюваним емпіричним дослідженням реальних open-source проєктів на основі стандартизованого протоколу збірки та тестування з використанням санітайзерів (ASan, UBSan). На відміну від суто концептуальних оглядів або ізольованих експериментальних звітів, у роботі запропоновано інтегрований підхід, що поєднує нормативно-стандартизаційний контекст (WG21) з практичними метриками (кількість попереджень, унікальні runtime-спрацювання, нормалізація на 10 000 LOC). Отримані результати дозволяють конкретизувати реальний рівень проявів memory-related дефектів у зрілих C++ бібліотеках за умов сучасних інструментів аналізу та формують емпіричну основу для оцінки доцільності подальшого розвитку профілів безпеки та механізмів формалізації часу життя в стандарті C++.

Висновки

У роботі систематизовано сучасні ініціативи WG21 [2] у сфері підвищення безпеки C++ та проведено відтворюваний експеримент на трьох open-source бібліотеках. Емпіричні результати показали відсутність UBSan-спрацювань у тестових сценаріях та один унікальний інцидент ASan у fmt, а також відмінності у кількості компіляторних попереджень. Це підтверджує, що поєднання інструментального рівня (санітайзери [1], попередження) з еволюцією мови (контракти, профілі, lifetime-аналіз) є практично обґрунтованою стратегією. Подальша робота може включати розширення вибірки, порівняння компіляторів (Clang/LLVM) та дослідження впливу різних профілів/режимів контрактів на реальні кодові бази.

Список літератури:

1. Serebryany K., Bruening D., Potapenko A., Vyukov D. AddressSanitizer: A Fast Address Sanity Checker. *Proceedings of the 2012 USENIX Annual Technical Conference* (*USENIX ATC '12*). 2012. URL: https://dl.acm.org/doi/10.5555/2342821.2342849
2. ISO/IEC JTC1/SC22/WG21. Working Draft, Standard for Programming Language C++. URL: https://open-std.org
3. Sutter H. Lifetime safety: Preventing common dangling. P1179R1. WG21, 2019. URL: https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1179r1.pdf
4. Stroustrup B., Dos Reis G. Design Alternatives for Type-and-Resource Safe C++. P2687R0. WG21, 2022. URL: https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2022/p2687r0.pdf
5. Doumler T. *at al*. Contracts for C++. P2900R13. WG21, 2025. URL: https://open-std.org/jtc1/sc22/wg21/docs/papers/2025/p2900r13.pdf
6. Sutter H. Core safety profiles for C++26. P3081R2. WG21, 2025. URL: https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2025/p3081r2.pdf
7. Berne J., Lakos J. Prevent Undefined Behavior By Default. P3558R0. WG21, 2025. URL: https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2025/p3558r0.pdf
8. Dos Reis G. Reference checking. P2878R1. WG21, 2023. URL: https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2023/p2878r1.html
9. Sutter H. Making Safe C++ Happen. P3700R0. WG21, 2025. URL: https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2025/p3700r0.html
10. Du J. X. K. L. Z. *at al*. A Study of Compiler-Introduced Security Bugs. 2023. URL: https://nebelwelt.net/files/23SEC4.pdf
11. X. Yang, Y. Chen, E. Eide, and J. Regehr, "Finding and understanding bugs in C compilers," *ACM SIGPLAN Not.*, vol. 46, no. 6, pp. 283–294, Jun. 2011. doi: https://doi.org/10.1145/1993316.1993532
12. Verdi M. *at al*. An Empirical Study of C++ Vulnerabilities in Crowd-Sourced Code Examples. 2019. URL: https://arxiv.org/pdf/1910.01321
13. Xu H. та ін. Memory-Safety Challenge Considered Solved? An In-Depth Study of Rust. 2020. URL: https://arxiv.org/pdf/2003.03296

14.Moghadam V. E. Memory Integrity Techniques for Memory-Unsafe Languages: A Survey. 2024. URL: https://iris.santannapisa.it/retrieve/9ebf4c27-86fb-401c-827b-6fad063f1d08/Memory_Integrity_Techniques_for_Memory-Unsafe_Languages_A_Survey.pdf
15.nlohmann. JSON for Modern C++. GitHub repository. URL: https://github.com/nlohmann/json

Надійшла до редколегії 27.05.2025

Відомості про авторів:
Запухляк Руслан Ігорович – кандидат фізико-математичних наук, професор кафедри комп'ютерної інженерії та електроніки, Прикарпатський національний університети імені Василя Стефаника / Vasyl Stefanyk Precarpathian National University, Україна; email: ruslan.zapukhlyak@pnu.edu.ua, ORCID: https://orcid.org/0000-0002-4204-8235
Довгий Віктор Володимирович – канд. техн. наук, старший викладач кафедри комп'ютерної інженерії та електроніки, Прикарпатський національний університети імені Василя Стефаника / Vasyl Stefanyk Precarpathian National University, Україна; email: viktor.dovhyi@pnu.edu.ua ORCID: https://orcid.org/0009-0009-7158-6938

ABSTRACTS

РЕФЕРАТИ

ELECTRONICS, ELECTRONIC COMMUNICATIONS, INSTRUMENTATION AND RADIO ENGINEERING

ЕЛЕКТРОНІКА, ЕЛЕКТРОННІ КОМУНІКАЦІЇ, ПРИЛАДОБУДУВАННЯ ТА РАДІОТЕХНІКА

UDC 614.842.435

Fire alarm system based on NodeMcu V3 ESP8266 module for domestic use / *N.A. Ostapyshyn, V.I. Mandziuk* // Information Technologies and Engineering Electronics : Scientific journal, 2025, №2, P. 7-13.

The work proposes a prototype of a fire alarm system based on the NodeMCU V3 ESP8266 platform. The element base is selected for the implementation of the device using sensors MQ-2 smoke, MQ-2 gas, DHT22 temperature and humidity, SW-420 vibration, a trip relay and a M590 GSM module for exchanging SMS messages and transmitting data via GPRS. The program code for the device is written in the C++ programming language.

Key words: fire alarm; microcontroller system; physical quantity sensor; structural electrical diagram; program code.

9 fig. Ref: 11 items.

УДК 614.842.435

Протипожежна система на основі модуля NodeMcu V3 ESP8266 для побутового застосування / *Н.А. Остапишин, В.І. Мандзюк* // Інформаційні технології та інженерна електроніка : Науковий журнал, 2025, №2, С. 7-13.

У роботі запропоновано прототип системи пожежної сигналізації на базі платформи платформи NodeMCU V3 ESP8266. Підібрано елементну базу для реалізації роботи пристрою з використанням давачів диму MQ-2, газу MQ-2, температури і вологості DHT22, вібрації SW-420, реле спрацьовування та GSM-модуля M590 для обміну SMS-повідомленнями і передачею даних через GPRS. Написано програмний код для роботи пристрою на мові програмування C++.

Ключові слова: пожежна сигналізація; мікроконтролерна система; давач фізичної величина; структурна електрична схема; програмний код.

Іл. 9. Бібліогр.: 11 назв.

SOFTWARE ENGINEERING

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

UDC 004.056.5:004.8

Comparative analysis of transformer models and hybrid architectures in phishing content detection tasks / *I.M. Lazarovych, A.D. Kvasniuk* // Information Technologies and Engineering Electronics : Scientific journal, 2025, №2, P. 14-19.

The work demonstrates that the use of hybrid neural network architectures based on CNN, LSTM, and the attention mechanism provides higher efficiency in phishing content detection compared to the transformer-based BERT model. The experimental analysis conducted using K-Fold cross-validation and standard evaluation metrics confirmed the advantages of combining local text analysis with long-term context modeling. It was found that the integration of the attention mechanism increases classification accuracy and recall by focusing on the most informative text fragments. The obtained results demonstrate the potential of hybrid approaches for building practical phishing attack detection systems.

Key words: phishing content, phishing detection, natural language processing, transformer models, BERT, hybrid neural networks, CNN, LSTM, attention mechanism, text classification.

6 fig. Ref: 11 items.

УДК 004.056.5:004.8

Порівняльний аналіз трансформерних моделей та гібридних архітектур у задачах виявлення фішингового контенту / *І.М. Лазарович, А.Д. Кваснюк* // Інформаційні технології та інженерна електроніка : Науковий журнал, 2025, №2, С. 14-19.

У роботі показано, що застосування гібридних нейромережевих архітектур на основі CNN, LSTM та механізму уваги забезпечує вищу ефективність виявлення фішингового контенту порівняно з трансформерною моделлю BERT. Проведений експериментальний аналіз із використанням K-Fold крос-валідації та стандартних метрик якості підтвердив переваги поєднання локального аналізу тексту й моделювання довготривалого контексту. Встановлено, що інтеграція механізму Attention підвищує точність і повноту класифікації за рахунок

фокусування на найбільш інформативних фрагментах тексту. Отримані результати свідчать про перспективність гібридних підходів для побудови практичних систем виявлення фішингових атак.

Ключові слова: фішинговий контент, виявлення фішингу, обробка природної мови, трансформерні моделі, BERT, гібридні нейронні мережі, CNN, LSTM, механізм уваги, класифікація тексту.

Іл. 6. Бібліогр.: 11 назв.

COMPUTER SCIENCE

КОМП'ЮТЕРНІ НАУКИ

UDC 004.9:005.5:640.41

Creation of an IDEF0 functional model for the information system of the International Student Youth Meeting Center / *L.B. Petryshyn, M.L. Petryshyn* // Information Technologies and Engineering Electronics : Scientific journal, 2025, №2, P. 20-32.

The paper justifies the necessity of implementing advanced information systems and technologies at the International Student Youth Meeting Center. The main goal of this project is to justify the need to implement advanced information systems and technologies in the International Center for Student Youth Meetings and to develop a clear functional model of its business processes. Using the IDEF0 methodology, a functional model of business processes was developed to optimize operations and support IT infrastructure.

Key words: information system, functional modeling, IDEF0, business processes, IT infrastructure, international center.

8 fig. Ref: 8 items.

УДК 004.9:005.5:640.41

Створення функціональної моделі IDEF0 для інформаційної системи Міжнародного центру зустрічей студентської молоді / *Л.Б. Петришин, М.Л. Петришин* // Інформаційні технології та інженерна електроніка : Науковий журнал, 2025, №2, С. 20-32.

У роботі обґрунтовано необхідність впровадження сучасних інформаційних систем та технологій у Міжнародному центрі зустрічей студентської молоді. Основною метою цього проекту є обґрунтування необхідності впровадження передових інформаційних систем і технологій у Міжнародному центрі зустрічей студентської молоді та розробка чіткої функціональної моделі його бізнес-процесів. За допомогою методології IDEF0 розроблено функціональну модель бізнес-процесів для оптимізації діяльності та підтримки IT-інфраструктури.

Ключові слова: інформаційна система, функціональне моделювання, IDEF0, бізнес-процеси, IT-інфраструктура, міжнародний центр.

Іл. 8. Бібліогр.: 8 назв.

COMPUTER ENGINEERING

КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

UDC 681.325+004.4

Using vector instructions of the x86-64 microprocessor in linear algebra problems / *V.I. Holota, V.M. Hryha, T.G. Benko, A.V. Morgun* // Information Technologies and Engineering Electronics : Scientific journal, 2025, №2, P. 33-43.

The paper shows the use of AVX vector instructions when solving linear algebra problems. Algorithms and a library of functions in the NASM assembly language have been developed for vector arithmetic operations (addition, multiplication) and solving systems of linear equations using the Jordan-Gauss method.

The performance of the developed programs using vector SIMD and scalar SISD technologies when solving systems of linear algebraic equations of different dimensions has been compared. According to the results of the experiments, the calculation speedup is obtained from 5 to 6 times when using AVX vector instructions.

Key words: x86-64 microprocessor; computing technologies; NASM assembler; vector instructions; Jordan-Gauss method.

2 tabl. 6 fig. Ref: 16 items.

УДК 681.325+004.4

Використання векторних інструкцій мікропроцесора x86-64 в задачах лінійної алгебри / *В.І. Голота, В.М. Грига, Т.Г. Бенько, А.В. Моргун* // Інформаційні технології та інженерна електроніка : Науковий журнал, 2025, №2, С. 33-43.

У роботі показано використання векторних інструкцій AVX при розв'язанні задач лінійної алгебри. Розроблено алгоритми і бібліотеку функцій на мові асемблера NASM для векторних арифметичних операцій (додавання, множення) та розв'язування систем лінійних алгебраїчних рівнянь методом Жордана-Гауса.

Порівняно швидкодію розроблених програм із використанням векторної SIMD і скалярної SISD технологій при розв'язування систем лінійних алгебраїчних рівнянь різної розмірності. За результатами експериментів отримно прискорення обчислень від 5 до 6 разів при використанні векторних інструкцій AVX.

Ключові слова: мікропроцесор x86-64; технології обчислень; асемблер NASM; векторні інструкції; метод Жордана-Гауса.

Табл. 2. Іл. 6. Бібліогр.: 16 назв.

UDC 004.43:004.415.5

Memory safety in C++: WG21 initiatives (lifetime, contracts, safety profiles) and empirical analysis of open-source projects / *R.I. Zapukhlyak, V.V. Dovhyi* // Information Technologies and Engineering Electronics : Scientific journal, 2025, №2, P. 44-50.

The study demonstrates memory safety in C++ in the context of ISO WG21 initiatives and an empirical evaluation on real-world open-source code. We summarize typical undefined behavior (UB) and vulnerability classes related to object lifetimes, out-of-bounds accesses, and resource misuse. We review WG21/SG23 directions: Lifetime Safety, Contracts, and Safety Profiles as a staged approach to increase safety guarantees while preserving C++ performance and ecosystem. We conduct a reproducible experiment on three libraries (fmt, spdlog, nlohmann/json) using g++ 13.3 with AddressSanitizer and UndefinedBehaviorSanitizer, provide a build-and-test protocol, normalized metrics, tables, and charts. The data show a low incidence of runtime findings in mature libraries under test coverage, while supporting the relevance of both tooling and standardization efforts to reduce UB-related risks.

Key words: C++, memory safety, undefined behavior, lifetime safety, contracts, safety profiles, sanitizers, ASan, UBSan.

3 tabl. 4 fig. Ref: 15 items.

УДК 004.43:004.415.5

Безпека пам'яті у C++: ініціативи WG21 (lifetime, contracts, safety profiles) та емпіричний аналіз open-source проєктів / *Р.І. Запухляк, В.В. Довгий* // Інформаційні технології та інженерна електроніка : Науковий журнал, 2025, №2, С. 44-50.

У роботі показано, аналіз по підвищенню безпеки пам'яті (memory safety) у C++ у контексті сучасних ініціатив ISO WG21 та практичного аналізу помилок у реальному коді. Узагальнено проблематику undefined behavior (UB) і класів вразливостей, що виникають унаслідок помилок часу життя об'єктів, доступів поза межами буферів та некоректного керування ресурсами. Розглянуто підходи WG21/SG23 до формалізації Lifetime Safety, Contracts та Safety Profiles (профілів безпеки) як механізмів поступового підвищення гарантій без руйнування екосистеми C++. Проведено відтворюваний експеримент на трьох open-source бібліотеках (fmt, spdlog, nlohmann/json) із застосуванням g++ 13.3 та санітайзерів AddressSanitizer (ASan) і UndefinedBehaviorSanitizer (UBSan). Подано протокол збірки, тестування, збору та нормалізації метрик; наведено таблиці результатів і діаграми. Емпіричні дані демонструють низьку частоту runtime-помилок у зрілих бібліотеках за умов покриття тестами, але підтверджують практичну цінність інструментів та напрямів стандартизації для системного зменшення ризиків UB.

Ключові слова: C++, безпека пам'яті, undefined behavior, lifetime safety, contracts, safety profiles, ASan, UBSan.

Табл. 3. Іл. 4. Бібліогр.: 15 назв.

SCIENTIFIC JOURNAL
INFORMATION TECHNOLOGIES AND ENGINEERING ELECTRONICS
ISSN 3083-7618
eISSN 3083-7626
Issue 2
In English and Ukrainian

НАУКОВИЙ ЖУРНАЛ
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА ІНЖЕНЕРНА ЕЛЕКТРОНІКА
ISSN 3083-7618
eISSN 3083-7626
Випуск 2
Англійською та українською мовами

Комп'ютерна верстка І.В. Свид

Підп. до друку 27.06.2025. Формат 60x90/8. Папір офсет. Гарнітура Таймс. Друк. ризограф.
Ум. друк. арк. 6,3. Тираж 100 прим. Ціна договір.

Прикарпатський національний університет імені Василя Стефаника (ПНУ ім. Василя Стефаника)
м. Івано-Франківськ, вул. Шевченка, 57 тел. +38 (0342) 75-23-51, факс +38 (0342) 53-15-74,
e-mail: office@pnu.edu.ua.

Друк: підприємець Голіней О.М.
вул. Галицька, 128, м. Івано-Франківськ, 76008
Тел. +38(0342) 58-04-32