УДК 004.415.2:[004.89:504.064.36]:519.816

Бородкіна Ірина Лаврентіївна
кандидат технічних наук, доцент, доцент кафедри комп'ютерних наук,
Національний університет біоресурсів і природокористування України
ORCID: https://orcid.org/0000-0003-3667-3728
E-mail: i.borodkina@nubip.edu

Бородкін Георгій Олексійович
старший викладач кафедри комп'ютерних наук,
Національний університет біоресурсів і природокористування України
ORCID: https://orcid.org/0000-0002-6488-6512
E-mail: heorhii.borodkin@nubip.edu.ua

Міловідов Юрій Олегович
старший викладач кафедри комп'ютерних наук,
Національний університет біоресурсів і природокористування України
ORCID: https://orcid.org/0009-0000-5705-3590
E-mail: yurii_milovidov@nubip.edu.ua

ВИБІР МОВИ ПРОГРАМУВАННЯ ДЛЯ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ МОНІТОРИНГУ ЛІСІВ І ПРОТИДІЇ ПОЖЕЖАМ МЕТОДОМ АНАЛІЗУ ІЄРАРХІЙ

Анотація. *Робота присвячена розв'язанню актуальної науково-практичної задачі — обґрунтуванню вибору мови програмування для створення інтелектуальних систем екологічного моніторингу в режимі реального часу. У дослідженні наголошується на переході від пасивного спостереження до активного використання алгоритмів штучного інтелекту (комп'ютерного зору, глибокого навчання) для раннього виявлення лісових пожеж. Наукова новизна полягає у застосуванні математичного апарату методу аналізу ієрархій (MAI) для мінімізації суб'єктивізму при виборі технологічного стека. Авторами побудовано ієрархічну модель оцінювання чотирьох альтернатив: Python, C++, Rust та Lisp. Для порівняльного аналізу обрано п'ять ключових критеріїв: швидкість обробки, точність розпізнавання, автономність (енергоефективність), масштабованість та вартість розробки. Розрахунок вагових коефіцієнтів критеріїв продемонстрував пріоритетність точності детекції (43%) та продуктивності роботи системи (28%). За результатами обчислення інтегральних показників виявлено перевагу мови Rust (8,54), яка випередила C++ (7,82) та Python (7,16). Встановлено, що Rust є найбільш збалансованим рішенням для реалізації концепції Edge Computing, оскільки забезпечує безпеку пам'яті та високу швидкість обробки безпосередньо на автономних пристроях. У підсумку запропоновано оптимальну стратегію побудови гібридної архітектури: використання Rust для низькорівневих модулів детекції та Python для хмарної аналітики даних.*

Ключові слова: *метод аналізу ієрархій, інтелектуальні системи моніторингу, моніторинг лісових пожеж, штучний інтелект, мова програмування Rust, мова програмування Python, глибоке навчання, обробка в реальному часі, Edge Computing, багатокритеріальне прийняття рішень, багатокритеріальний аналіз.*

Актуальність. Сучасний етап розвитку систем екологічної безпеки характеризується переходом від пасивного спостереження та фіксації результатів до активного моніторингу із використанням інтелектуальних систем у режимі реального часу.

У цій сфері одним із найскладніших завдань є раннє виявлення різноманітних нестандартних ситуацій, зокрема лісових пожеж та інших екологічно небезпечних явищ, де кожна хвилина затримки реагування може призвести до стрімкого зростання економічних і екологічних збитків.

Для вирішення цих проблем необхідно створювати системи, які здатні здійснювати безперервний моніторинг стану довкілля в реальному часі та оперативно приймати відповідні рішення. Створення таких систем потребує інтеграції складних алгоритмів штучного інтелекту (Computer Vision, Deep Learning) з апаратними засобами, що розбудовані зокрема на базі бездротових технологій і безпілотних літальних апаратів.

DOI: 10.31548/itees.2025.02.028
ISSN 3083-7502

На етапі проєктування та обґрунтування вибору інструментальних засобів розробки інтелектуальних систем моніторингу надзвичайно важливою є проблема вибору мови програмування. Це питання не зводиться до простого порівняння характеристик різних мов, а передбачає розв'язання багатокритеріальної задачі, що потребує балансування між високою швидкістю обробки потоків даних і обмеженою енергоефективністю автономних пристроїв. Крім того, необхідно враховувати гнучкість реалізації сучасних нейромережевих моделей.

На сьогодні відсутня єдина методика оцінювання програмного інструментарію, що часто призводить до застосування суб'єктивних підходів, що в свою чергу, знижує загальну надійність та ефективність інтелектуальних систем моніторингу довкілля.

Аналіз останніх досліджень та публікацій. Теоретичним фундаментом розв'язання задачі багатокритеріального вибору можна вважати роботи Saaty T. L., у яких було запропоновано метод аналізу ієрархій [1], [2]. Цей математичний апарат базується на обчисленні власних значень матриць попарних порівнянь і на сьогодні залишається одним із стандартних підходів у системній інженерії [3].

Сучасні модифікації та адаптації цього методу для цифрових екосистем запропоновано в роботах Goepel K. D., де акцентується увага на важливості узгодженості експертних суджень під час оцінювання програмних засобів [4]. Останні дослідження також підтверджують ефективність використання методу аналізу ієрархій у задачах вибору програмного забезпечення та інтелектуальних систем, зокрема в умовах невизначеності та багатокритеріальності. В роботі [6] розглянуто методологічні засади, які пов'язані з процесами побудови аналітичної ієрархії, зокрема, розглядаються питання, що пов'язані з моделюванням проблем, попарними порівняннями, шкалами оцінювання.

Методи багатокритеріального прийняття рішень (MCDM) можуть використовуватись в різних типах застосувань. В [7] проведено огляд поширених методів багатокритеріального прийняття рішень, розглянуто їх переваги та недоліки. В результаті, цю роботу можна вважати керівництвом, яке допомагає визначити, як методи MCDM слід використовувати в конкретних ситуаціях.

В роботі [8] автори стверджують, що вибір конкретного методу багатокритеріального прийняття рішень та способу нормалізації даних критично впливає на надійність результатів. Вони пропонують оцінювати інженерні рішення через компроміс між 5 вимірами: продуктивністю, екологічним впливом, економічним впливом, соціальним впливом та циклічністю.

Іншим аспектом, на який слід звернути увагу підчас розробки інтелектуальних систем моніторингу є специфіка побудови таких систем залежно від сфери застосування. Так, питання, що пов'язані з виявленням пожеж детально досліджуються в роботах López et al., де автори наголошують на перевагах концепції Edge Computing (прикордонні або периферійні обчислення) для мінімізації часу реакції. Питання точності розпізнавання пожеж на базі глибокого навчання (Deep Learning) є центральними у дослідженнях B.Jahne [9], який вказує на пряму залежність ефективності алгоритмів від обчислювальної архітектури обраної мови.

В роботі [10] автори наголошують на необхідності переходу від класичних алгоритмів до алгоритмів глибокого навчання і зазначають, що інтеграція глибокого навчання в системи моніторингу лісів дозволила значно підвищити точність ідентифікації пожеж, зокрема автори доводять, що саме Deep Learning мінімізує кількість хибних спрацювань. Проте такий підхід висуває нові вимоги до архітектури обчислювальної системи.

Питання обробки відеопотоків в реальному часі розглядаються в [11], де автори наголошують на необхідності високої обчислювальної потужності, що прямо вказує на важливість використання відповідних програмних засобів і, зокрема, мов програмування.

Разом з тим, аналіз літературних джерел свідчить про наявність розриву між теоретичними дослідженнями алгоритмів штучного інтелекту та практичними рекомендаціями щодо вибору системних мов програмування для їх реалізації. Традиційні порівняння мов програмування не завжди враховують специфічні особливості саме систем екологічного моніторингу, такі як автономність та масштабованість у диких природних

умовах. Це зумовлює необхідність проведення комплексного аналізу мов програмування на основі математичних методів прийняття рішень.

Мета дослідження полягає в математичному обґрунтуванні вибору оптимального інструментарію програмування для розробки інтелектуальних систем моніторингу лісових пожеж шляхом застосування методу аналізу ієрархій. Цей підхід дозволить забезпечити баланс між високою точністю розпізнавання екологічно небезпечних ситуацій та продуктивністю обробки даних у реальному часі. Він може стати підгрунтям для переходу від суб'єктивного вибору мови програмування до об'єктивного математичного розрахунку, який базується на критичних вимогах безпеки.

Матеріали і методи дослідження. Вибір мови програмування як інструменту реалізації проєкту – це більша частина складової успіху проєкту. Скористаємось методом багатокритеріального аналізу рішень (MCDA), а саме методом аналізу ієрархій [1]. Цей метод являє собою структурований математичний апарат для прийняття складних рішень. Він розбиває проблему на ієрархію (мета, критерії, альтернативи) і за допомогою парних порівнянь дозволяє визначити найкращий варіант на основі кількісної оцінки суб'єктивних думок експертів. Таке попарне порівняння критеріїв суттєво зменшує суб'єктивну похибку.

Сформулюємо задачу вибору мови програмування як функцію корисності U, яку потрібно максимізувати.

$$U(L) = \sum_{i=1}^{n} w_i * c_i(L), \quad (1)$$

де L – конкретна мова програмування (Piton, C++, Julia, Rust або інша);

w_i – ваговий коефіцієнт *i*-го критерію, який визначає, наскільки цей критерій важливий для оцінювання;

$c_i(L)$ – оцінка мови L за *i*-м критерієм (зазвичай її значення береться від 0 до 1).

Для інтелектуальних систем основними критеріями зазвичай використовують:

c_1 – наявність бібліотек;

c_2 – продуктивність (особливо для систем, що працюють в реальному часі);

c_3 – швидкість розробки (тут слід враховувати лаконічність синтаксису та простоту прототипування);

c_4 – доступність документації та готових рішень;

c_5 – можливість інтеграції у вже існуючі системи.

Покроково алгоритм розрахунку ваги критеріїв має наступний вигляд.

Крок 1. Будуємо матрицю попарних порівнянь. Це квадратна матриця розміром $n * n$, де n - кількість критеріїв. Кожний елемент цієї матриці a_{ij} показує ступінь переваги критерію i над критерієм j за шкалою Сааті (значення 2, 4, 6, 8 є проміжними):

1 – однакова важливість,

3 – помірна перевага одного над іншим,

5 – суттєва перевага,

7 – значна перевага,

9 – абсолютна перевага.

У разі побудови матриці попарних порівнянь працює правило зворотної симетрії: якщо критерій А в 3 рази важливіший критерія В, то критерії В важливіший за А в 1/3 рази. На діагоналі матриці порівнянь завжди стоять одиниці.

Крок 2. Розраховуємо вагу кожного критерію. Для цього розраховуємо суму елементів у кожному стовпці матриці пріоритетів, нормуємо матрицю, розділивши кожний елемент матриці на суму його стовпця, та обчислюємо середнє арифметичне для кожного рядка отриманої на попередньому кроці нормованої матриці.

DOI: 10.31548/itees.2025.02.028
ISSN 3083-7502

$$w_i = \frac{1}{n}\sum_{j=1}^{n}\frac{a_{ij}}{\sum_{k=1}^{n} a_{kj}}.$$

В результаті буде отримано набір значень ваги критеріїв w_i сума яких дорівнює одиниці.

Крок 3. Перевіряємо отримані результати на узгодженість. Скориставшись алгоритмом, наведеним в [12] та он-лайн калькулятором для MAI [5], розраховуємо головне власне число λ_{max} та знаходимо індекс узгодженості CI:

$$CI = \frac{\lambda_{max} - n}{n - 1}. \quad (2)$$

Та відношення узгодженості CR:

$$CR = \frac{CI}{RI} \quad (3)$$

В цьому випадку RI – табличне значення випадкової узгодженості для матриці заданого розміру. Якщо $CR \leq 0{,}1$ (10%) то ваги критеріїв w_i вважаються прийнятними, якщо CR приймає більше значення, то матрицю порівнянь слід переглянути через те, що надані експертом судження є суперечливими.

Результати дослідження та їх обговорення. Побудуємо матрицю порівнянь для вибору мови програмування для інтелектуальної системи моніторингу лісів та боротьби з пожежами. В такій системі є критично важливою швидкість обробки даних в реальному часі (це забезпечить виявлення вогню за лічені секунди) та можливість інтеграції системи з наявним апаратним обладнанням (дрони, супутники, тепловізійні пристрої). Це означає, що критерії, за якими ми будемо обирати оптимальну мову програмування, мають бути такими:

C_1 – швидкість обробки в реальному часі (характеризує здатність мови програмування обробляти відеопотік з мінімальною затримкою).

C_2 – точність розпізнавання (вона забезпечується наявністю в мові програмування оптимізованих бібліотек для нейромережевих обчислень);

C_3 – автономність (в процесі моніторингу необхідно забезпечити мінімальне споживання енергії та ресурсів пам'яті апаратним модулем);

C_4 – вартість розробки (в цьому критерії слід враховувати наявність фахівців та швидкість написання коду).

C_5 – масштабованість (цей критерій відповідає за стабільність роботи системи у разі розширення мережі пристроїв).

Пріоритетність критеріїв була визначена виходячи зі специфіки систем екологічної безпеки, де вирішальними факторами є час виявлення пожежі та надійність ідентифікації пожежі, яка напряму пов'язана з мінімізацією хибних спрацювань. Таким чином, в нашому дослідженні домінуючими є критерії C_1 та C_2.

Для організації процедури порівняння була побудована матриця $A = [a_{ij}]$, де a_{ij} – ступінь переваги і -го критерію над j-м за дев'ятибальною шкалою Сааті. За цих умов матриця попарних порівнянь має вигляд, що наведений у табл. 1.

Для отримання вектору локальних пріоритетів w був використаний метод нормалізації геометричного середнього:

$$w_i = \frac{\sqrt[n]{\prod_{j=1}^{n} a_{ij}}}{\sum_{k=1}^{n}\sqrt[n]{\prod_{j=1}^{n} a_{kj}}}.$$

Таблиця 1 – Матриця попарних порівнянь з розрахованою вагою критеріїв w_i

Критерії	C_1	C_2	C_3	C_4	C_5	Вага (w_i)
C_1 (Швидкість)	1	1/2	3	5	4	**0.28**
C_2 (Точність)	2	1	4	7	5	**0.43**
C_3 (Автономність)	1/3	1/4	1	3	2	**0.13**
C_4 (Вартість)	1/5	1/7	1/3	1	1/2	**0.06**
C_5 (Масштабованість)	1/4	1/5	1/2	2	1	**0.10**

На наступному кроці для верифікації моделі було розраховано головне власне число $\lambda_{max} = 5{,}21$ нашої матриці, що дозволило обчислити індекс узгодженості CI. Для цього була використана формула (2) для n=5, де n – розмірність матриці.

$$CI = \frac{5{,}21 - 5}{5 - 1} = 0{,}0525.$$

Для підтвердження достовірності результатів за формулою (3) було розраховане відношення узгодженості

$$CR = \frac{0{,}0525}{1{,}12} = 0{,}046, \text{ або } 4{,}6\%. \text{ (для } n = 5, \text{значення } RI = 1{,}12).$$

Оскільки $CR < 10\%$, то отримані значення ваги вважаються достовірними.

Тепер можна перейти до порівняльного аналізу мов програмування. Альтернативами, серед яких будемо обирати, розглянемо такі мови програмування: Python, C++, Rust, Lisp. Проведемо оцінювання цих альтернатив за кожним із п'яти критеріїв. Узагальнені результати такого оцінювання за 10-бальною шкалою наведено в табл. 2.

Таблиця 2 – Матриця оцінювання альтернатив за шкалою (1-10)

Критерій	Python	C++	Rust	Lisp
Точність (0.43)	10	7	8	9
Швидкість (0.28)	3	10	10	5
Автономність (0.13)	4	9	10	4
Масштабованість (0.10)	9	6	7	6
Вартість (0.06)	10	4	5	5

Розрахуємо інтегральний показник (1) для кожної з альтернатив:

U(Python) = 10*0,43+3*0,28+4*0,13+9*0,10+10*0,06 = 7,16

U(C++) = 7*0,43+10*0,28+9*0,13+6*0,10+4*0,06 = 7,82

U(Rust) = 8*0,43+10*0,28+10*0,13+7*0,10+5*0,06 = 8,54

U(Lisp) = 9*0,43+5*0,28+4*0,13+6*0,10+5*0,06 = 6,69.

DOI: 10.31548/itees.2025.02.028
ISSN 3083-7502

Згідно з розрахунками, мова Rust отримала найвищий інтегральний показник (8.54), випередивши C++ (7.82) та Python (7.16). Найнижчий показник продемонструвала мова програмування Lisp (6.69), що пояснюється її меншою придатністю для систем реального часу з обмеженими апаратними ресурсами.

Аналіз отриманих інтегральних показників (U) дозволяє глибше зрозуміти ієрархію вибору мови програмування для інтелектуальних систем екологічного моніторингу з критично важливими факторами, як-то виявлення пожеж. Отримані результати підтверджують гіпотезу про те, що для інтелектуальних систем реального часу технічні характеристики виконання програмного коду переважають над зручністю розробки.

Найвищий інтегральний показник мови Rust пояснюється її унікальною архітектурою. В процесі дослідження виявлено, що за критеріями «Швидкість» (C_1) та «Автономність» (C_3) мова програмування Rust не поступається C++, проте значно випереджає її за безпекою. У системах, що працюють в режимі Edge computing (на дронах або на автономних вежах), помилка переповнення буфера в C++ може призвести до повної зупинки пристрою. Rust виключає такі помилки на етапі компіляції, що робить систему більш живучою без втручання людини. Хоча мова програмування Python посіла загальне третє місце з інтегральним показником 7.16, вона отримала найвищий бал за критерієм «Точність» (C_2). Це зумовлено наявністю зрілих бібліотек (PyTorch, TensorFlow). При цьому спостерігається чіткий розрив між зручністю розробки в Python та можливістю його запуску на малопотужному обладнанні. Це обґрунтовує необхідність переписування навчених моделей на мови системного рівня (Rust/C++) для їх реалізації безпосередньо в місці моніторингу.

Критерій «Автономність» (C_3) став вирішальним для відсіювання мови програмування Lisp та зниження рейтингу мови Python. Для систем моніторингу лісів, де живлення здійснюється від сонячних батарей або акумуляторів, використання інтерпретованих мов є неефективним через високе енергоємне навантаження на процесор. Перехід на мови з прямим керуванням пам'яттю дозволяє подовжити термін роботи автономного вузла на 25-40% при тій самій ємності батареї.

Отже, на основі наведених вище міркувань можна запропонувати відійти від концепції використання лише однієї мови програмування для розробки інтелектуальних систем моніторингу лісів та боротьби з пожежами і запропонувати таку стратегію: для детекції (Edge) доцільно використовувати мову Rust, яка забезпечить максимальну швидкість та економію енергії; для обробки підтверджених даних доцільно використовувати мову Python, яка характеризується максимальною гнучкістю алгоритмів.

Висновки і перспективи. В результаті проведеного дослідження було розв’язано науково-практичну задачу обґрунтування вибору мови програмування для інтелектуальних систем моніторингу лісів та боротьби з пожежами на основі методу аналізу ієрархій. Застосування цього методу дозволило зробити нижче зазначені висновки.

Для систем екологічної безпеки критичними критеріями є точність розпізнавання (43%) та швидкість обробки даних у реальному часі (28%). Економічні чинники, такі як вартість розробки (6%), мають вторинне значення в контексті мінімізації ризиків розпізнавання масштабних лісових пожеж. Це означає, що для обирання мови програмування технічні характеристики повинні бути більш пріоритетними за зручність програмування та інші чинники.

Застосування методу ієрархій дозволило сформувати рейтинг програмного інструментарію. За інтегральним показником ефективності перше місце посіла мова програмування Rust з інтегральним показником 8,54. Вона виявилася найбільш збалансованим рішенням, яке поєднує продуктивність системного рівня із сучасними механізмами безпеки використання оперативної пам’яті, що є критичним для автономних пристроїв.

Оцінено роль альтернативних мов програмування. Встановлено, що C++ (інтегральний показник 7.82) залишається потужним конкурентом для низькорівневих модулів, тоді як Python (інтегральний показник 7.16) є оптимальним вибором для верхнього рівня архітектури (хмарна аналітика). Lisp (інтегральний показник 6.69) продемонстрував потенціал для

складної логіки планування, проте наразі є обмеженим у застосуванні парадигми Edge computing, тобто у разі перенесення обробки даних із віддалених серверів (хмари) безпосередньо на пристрій, який ці дані збирає (камеру, дрон або датчик).

Сформульовано рекомендації до системної архітектури. Для досягнення максимальної ефективності рекомендовано впровадження гібридної моделі розробки: використання Rust для прошивки датчиків та дрон-систем первинної детекції, та Python – для централізованої обробки даних і візуалізації результатів моніторингу.

Практичне значення отриманих результатів полягає у можливості використання запропонованої математичної моделі розробниками складних IT-систем для об'єктивного вибору технологічного стека, що дозволяє уникнути архітектурних помилок на етапі проектування.

Список використаних джерел

1. Saaty, T. L. (1980). The analytic hierarchy process. McGraw-Hill.
2. Saaty, T. L. (2003). Decision making with the analytic hierarchy process: Why is the principal eigenvector necessary. European Journal of Operational Research, 145(1), 85–91. https://doi.org/10.1016/S0377-2217(02)00227-8.
3. Saaty, T. L. (2008). Decision making for leaders: The analytic hierarchy process for decisions in a complex world (Rev. ed.). RWS Publications.
4. Goepel, K. D. (2013). Implementing the analytic hierarchy process as a standard method for multi-criteria decision making in corporate enterprises. In Proceedings of the International Symposium on the Analytic Hierarchy Process. BPMSG.
5. Goepel, K. D. (n.d.). Analytic hierarchy process (AHP) tutorial. Business Performance Management Singapore. Retrieved October 24, 2023, from https://bpmsg.com/ahp-introduction/
6. Ishizaka, A., & Labib, A. (2011). Review of the main developments of the analytic hierarchy process. Expert Systems with Applications, 38(11), 14036–14039. https://doi.org/10.1016/j.eswa.2011.04.143.
7. Velasquez, M., & Hester, P. T. (2021). An analysis of multi-criteria decision making methods. International Journal of Operations Research, 10(2), 56–66.
8. Malefaki, S., Markatos, D., Filippatos, A., & Pantelakis, S. G. (2025). A comparative analysis of multi-criteria decision-making methods and normalization techniques in holistic sustainability assessment for engineering applications. Aerospace, 12(2), 100. https://doi.org/10.3390/aerospace12020100.
9. Jähne, B. (2022). Digital image processing: Concepts, algorithms, and scientific applications (3rd ed.). Springer. https://doi.org/10.1007/978-3-662-03174-2.
10. Saleh, A., Zulkifley, M. A., Harun, H. H., Gaudreault, F., Davison, I., & Spraggon, M. (2024). Forest fire surveillance systems: A review of deep learning methods. Heliyon, 10(1), Article e23127. https://doi.org/10.1016/j.heliyon.2023.e23127.
11. Bustamante, A., Belmonte, L. M., Morales, R., Pereira, A., & Fernández-Caballero, A. (2022). Video Processing from a Virtual Unmanned Aerial Vehicle: Comparing Two Approaches to Using OpenCV in Unity. Applied Sciences, 12(12), 5958. https://doi.org/10.3390/app12125958.
12. Vargas, R. V. (2010). Using the analytic hierarchy process (AHP) to select and prioritize projects in a portfolio. In Paper presented at PMI® Global Congress 2010—North America, Washington, DC. Project Management Institute. https://www.pmi.org/learning/library/analytic-hierarchy-process-prioritize-projects-6608.

Borodkina Iryna
Candidate of Engineering Sciences, Associate Professor, Department of Computer Science, National University of Life and Environmental Sciences of Ukraine
ORCID: https://orcid.org/0000-0003-3667-3728
E-mail: i.borodkina@nubip.edu.ua

DOI: 10.31548/itees.2025.02.028
ISSN 3083-7502

Borodkin Heorhii
Senior Lecturer of the Department of Computer Science,
National University of Life and Environmental Sciences of Ukraine
ORCID: https://orcid.org/0000-0002-6488-6512
E-mail: heorhii.borodkin@nubip.edu.ua

Milovidov Yurii
Senior Lecturer of the Department of Computer Science,
National University of Life and Environmental Sciences of Ukraine
ORCID: https://orcid.org/0009-0000-5705-3590
E-mail: yurii_milovidov@nubip.edu.ua

PROGRAMMING LANGUAGE SELECTION FOR INTELLIGENT FOREST MONITORING AND FIRE-FIGHTING SYSTEMS USING THE ANALYTIC HIERARCHY PROCESS

Abstract. *This paper addresses the critical issue of selecting the optimal programming language for developing intelligent environmental monitoring systems operating in real-time. The study emphasizes the transition from passive observation to the active use of artificial intelligence algorithms (Computer Vision, Deep Learning) for the early detection of forest fires. The scientific novelty lies in the application of the mathematical framework of the Analytic Hierarchy Process (AHP) to minimize subjectivity in the selection of the technology stack. The authors constructed a hierarchical evaluation model for four alternatives: Python, C++, Rust, and Lisp.For the comparative analysis, five key criteria were selected: processing speed, recognition accuracy, autonomy (energy efficiency), scalability, and development cost. The calculation of the criteria weight coefficients demonstrated the priority of detection accuracy (43%) and real-time system performance (28%). Based on the integral indicators, the Rust language (8.54) showed clear superiority over C++ (7.82) and Python (7.16). It was established that Rust is the most balanced solution for implementing the Edge Computing concept, as it ensures memory safety and high processing speeds directly on autonomous devices. Consequently, an optimal hybrid architecture strategy is proposed: utilizing Rust for low-level detection modules and Python for cloud data analytics and visualization. This approach allows for a shift from subjective tool selection to objective mathematical justification based on critical safety requirements.*

Keywords*: Analytic Hierarchy Process, Intelligent Monitoring Systems, Forest Fire Monitoring, Artificial Intelligence, programming language Rust, programming language Python, Deep Learning, Real-time Processing, Edge Computing, Multi-Criteria Decision Making.*