УДК 004.415.5:004.9
DOI https://doi.org/10.32782/tnv-tech.2025.2.9

ОЦІНЮВАННЯ ВПЛИВУ ВІДКЛЮЧЕННЯ ЛОГУВАННЯ НА ПРОДУКТИВНІСТЬ МОБІЛЬНИХ ЗАСТОСУНКІВ, СТВОРЕНИХ НА UNITY

Клєпцов А. А. *– магістр інженерії програмного забезпечення,*
головний інженер програміст
28Software
ORCID ID: 0009-0008-3147-0618

Гусєва-Божаткіна В. А. *– старший викладач кафедри програмного забезпечення автоматизованих систем*
Національного університету кораблебудування імені адмірала Макарова
ORCID ID: 0000-0002-1117-3391

У процесі розробки програмного забезпечення, зокрема ігор на рушії Unity [1], *досягти ідеальної реалізації з першої спроби практично неможливо. Незалежно від масштабу проєкту – чи то велика команда, чи індивідуальний розробник – програмісти постійно перевіряють правильність роботи свого коду. Для цього існує чимало інструментів: юніт-тестування, системи налагодження (debugger), встановлення точок зупину (breakpoints), а також спеціалізовані фреймворки, такі як Selenium (фреймворк для автоматичного тестування). Проте найпростішим і найбільш поширеним способом оперативного аналізу поведінки коду залишається логування. У середовищі Unity це реалізується через клас Debug і його метод Log, який дозволяє виводити повідомлення в консоль* [2]. *Попри простоту використання, логування може мати серйозний вплив на продуктивність, особливо у випадках великої кількості викликів – наприклад, при логуванні в кожному кадрі гри на мобільному пристрої. У великих проєктах, над якими одночасно працює значна кількість розробників, обсяг логування часто істотно перевищує умовну межу в один логаритмічний запис на кадр. Мобільні ігри або застосунки можуть містити сотні, а іноді й тисячі модулів, у кожному з яких реалізовано логування з метою верифікації коректності роботи або для виведення діагностичної інформації під час розробки.*

У даній роботі проаналізовано вплив активного логування на продуктивність Unity-додатків, орієнтованих на мобільні платформи. Автори ставлять за мету кількісно оцінити продуктивність додатку за умов активного та відключеного логування. Для цього буде застосовано низку базових статистичних методів, зокрема: обчислення середнього значення частоти кадрів (FPS), розрахунок відсоткової зміни продуктивності, а також використання t-критерію Стьюдента для оцінки статистичної значущості результатів. Запропонований підхід дозволить на практиці оцінити доцільність збереження або видалення викликів Debug.Log у фінальних білдах мобільних ігор, а також сформулювати практичні рекомендації для Unity-розробників.

Ключові слова: *Unity, продуктивність, мобільні платформи, логування, Debug.Log, t-критерій Стьюдента, FPS, оптимізація.*

Klieptsov A. A., Guseva-Bozhatkina V. A. Weaknesses of gaming anti-cheat systems: an analysis of contactless automation of gameplay using external devices

Software developers – particularly those working on Unity-based [1] *game projects – rarely produce perfect implementations on the first attempt. Regardless of project scale, whether working in large teams or individually, they continuously check their code for correctness. Developers rely on many tools to support this process, including unit tests, debuggers, breakpoints, and specialized frameworks such as Selenium for automated testing. Among these, logging offers the most direct and widely adopted method for quickly analyzing code behavior. In Unity, developers use the Debug class and its Log method to output messages to the console* [2]. *Despite its apparent simplicity, this method is critical in rapid iteration and issue tracking during development. Although easy to implement, frequent logging can reduce performance, especially in mobile applications that generate log entries every frame. In large projects with*

many contributors, developers often introduce logging statements in hundreds or thousands of modules to validate code logic or monitor runtime behavior. These statements frequently exceed the practical limit of one log call per frame, which can degrade the user experience on resource-constrained mobile platforms. This study examines how active logging affects the performance of Unity applications on mobile devices. The research compares two conditions: logging enabled and logging disabled. The analysis uses several fundamental statistical techniques, including calculations of average frames per second (FPS), percentage differences in performance, and Student's t-test to evaluate the statistical significance of the results. The methodology provides a practical foundation for evaluating whether Unity developers should retain or eliminate Debug. Log calls in final production builds for mobile games.

Key words: *Unity, performance, mobile platforms, logging, Debug.Log, Student's t-test, FPS, optimization.*

Постановка проблеми. У процесі розробки мобільних ігор на рушії Unity розробники широко застосовують системи логування для діагностики, моніторингу внутрішніх процесів і верифікації коректності виконання коду. Найпоширенішим інструментом для цього є метод Debug.Log, що дозволяє швидко отримувати інформацію про виконання певних ділянок коду без необхідності запуску складних механізмів трасування чи підключення зовнішніх засобів тестування. Незважаючи на зручність, численні виклики до Debug.Log можуть створювати значне навантаження на систему, особливо в мобільному середовищі з обмеженими ресурсами обчислення, пам'яті та енергоспоживання [3]. Проблема посилюється у великих проєктах, де логування розміщенно у великій кількості модулів, часто без централізованого контролю або вимкнення логів на етапі продакшену [4].

Через це постає необхідність у системному дослідженні того, наскільки активне логування впливає на продуктивність мобільних застосунків, розроблених на Unity. У той час як існуючі дослідження зосереджуються переважно на оптимізації графіки, навантаженні на GPU або використанні AssetBundle/ Addressables, вплив системного логування як фактора деградації продуктивності залишається недостатньо вивченим. Наявність об'єктивних експериментальних даних, що показують різницю у продуктивності з активним і вимкненим логуванням, дозволить розробникам ухвалювати обгрунтовані рішення щодо залишення чи видалення логів у фінальних збірках додатків.

Аналіз останніх досліджень і публікацій. Проблематика продуктивності мобільних застосунків, зокрема створених на рушії Unity, активно висвітлюється у сучасній літературі як прикладного, так і дослідницького характеру. Основну увагу автори приділяють оптимізації графічного рендерингу, менеджменту ресурсів, впливу фізичних компонентів та загальному зниженню обчислювального навантаження на мобільні пристрої. У виданні «*Unity Game Optimization*», автором якого є Кріс Дікінсон, окремий акцент зроблено на важливості раціонального використання логування. Автор звертає увагу на те, що часті виклики Debug.Log можуть спричинити суттєве падіння продуктивності, особливо у випадках, коли логування відбувається кожного кадру в реальному часі. Це, у свою чергу, може впливати як на стабільність виконання коду, так і на користувацький досвід загалом [5].

У збірнику «*Pro Unity Game Development with C#*», представлено окремий розділ, присвячений профілюванню та моніторингу продуктивності Unity-додатків. Автори зазначають, що налагоджувальні засоби, зокрема логування, необхідно контролювати й обмежувати на етапі компіляції релізних білдів. Постійна генерація логів у таких версіях призводить до непрогнозованих витрат ресурсів та може негативно позначитися на роботі застосунку на мобільних пристроях [6].

Таким чином, хоча в наведених джерелах містяться практичні рекомендації щодо уникнення надмірного логування, кількісного порівняльного аналізу його

впливу на продуктивність, з використанням статистичних методів, наразі недостатньо. Це підтверджує доцільність подальших експериментальних досліджень у цьому напрямку.

Мета статті. Метою даної статті є експериментальне дослідження впливу активного логування, реалізованого за допомогою методу Debug.Log у середовищі Unity, на продуктивність мобільних застосунків. Дослідження орієнтоване на кількісне порівняння ключових метрик продуктивності в умовах активного та відключеного логування.

У межах дослідження автори мають на меті:

- оцінити зміну середньої частоти кадрів (FPS) при різних сценаріях логування;
- визначити відсоткову різницю в продуктивності між конфігураціями з логуванням і без нього;
- перевірити статистичну значущість виявлених відмінностей за допомогою t-критерію Стьюдента.

Експериментальне тестування буде проводитися на готовому білді застосунку, запущеному на мобільній платформі Android. Результати дослідження дозволять зробити практичні висновки щодо доцільності залишення логів у фінальних білдах Unity-застосунків, орієнтованих на мобільні пристрої, та сформулювати рекомендації для розробників щодо ефективного використання логування з урахуванням впливу на продуктивність.

Виклад основного матеріалу. Для проведення дослідження було створено окремий проєкт у середовищі розробки Unity під назвою *DebugLogFpsTest*. У якості платформи використовувалась версія рушія Unity 6000.0.32f LTS, що забезпечує сумісність із сучасними мобільними пристроями (Рис. 1).

На початковому етапі було зібрано спрощену сцену, яка містить усі необхідні елементи інтерфейсу для відображення ключових параметрів експерименту. Зокрема, на екран виводиться інформація про середнє значення FPS, поточне

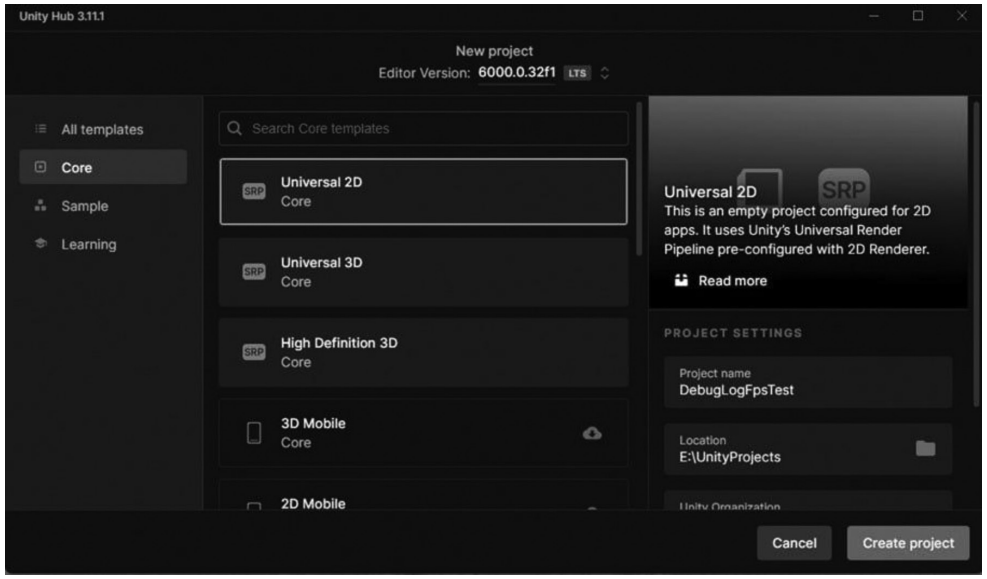

Рис. 1. Вікно створення відповідного проекту

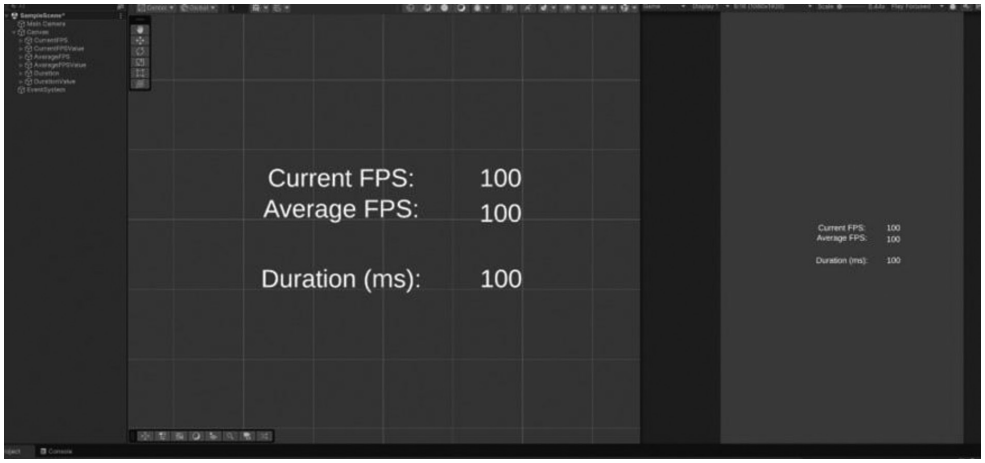

Рис. 2. Створена сцена

значення FPS у реальному часі, а також тривалість виконання логіки вимірювання продуктивності (Рис. 2).

Наступним кроком стало створення спеціального скрипта з назвою PerformanceLoggingTest [7], реалізованого на мові програмування C#. Основна мета скрипта – ініціювати логування за допомогою Debug.Log, обчислювати FPS, а також виводити агреговану інформацію на екран. Скрипт містить два конфігураційні параметри:

- logsPerFrame – кількість логів, які генеруються протягом одного кадру;
- durationSeconds – тривалість експериментального запуску в секундах.

Функціональність скрипта реалізовано у вигляді компонента, розміщеного на об'єкті типу Canvas. Для верифікації коректності реалізованої логіки було проведено два тестових проходи на настільному ПК. Під час першого запуску експерименту параметри logsPerFrame і durationSeconds було встановлено відповідно на значення 0 і 10, що відповідає виконанню логіки тривалістю 10 секунд без активного логування. Приклад результатів наведено на рисунку (Рис. 3).

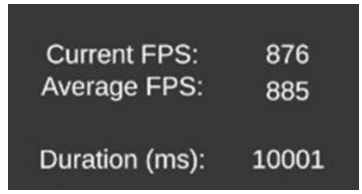

Рис. 3. Результативні значення FPC без логування на ПК

У другому тестовому проході значення параметрів було змінено: logsPerFrame = = 1000, durationSeconds = 10. Як показано на рисунку (Рис. 4), середнє значення FPS у цьому випадку істотно знизилося порівняно з конфігурацією без логування. Така динаміка підтверджує гіпотезу про те, що інтенсивне використання методів Debug.Log справляє значний негативний вплив на продуктивність застосунку.

Отримані результати є достатніми для продовження експериментального дослідження вже на мобільній платформі, де обмеження обчислювальних ресурсів

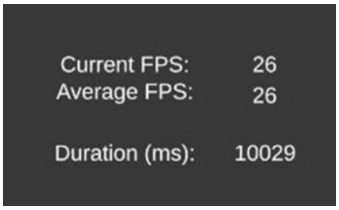

Рис. 4. Результативні значення FPC з логування на ПК

можуть ще більш виразно продемонструвати вплив логування на продуктивність системи.

З метою забезпечення можливості кількісної оцінки впливу активного логування на продуктивність мобільного застосунку було реалізовано вдосконалений інструмент збору статистичних даних безпосередньо під час виконання програми на цільовій платформі Android [8]. Основою виступає модифікований скрипт, реалізований на C# у середовищі Unity, який проводить автоматичне вимірювання частоти кадрів (FPS) на кожному кадрі протягом заданого періоду часу.

Користувач має змогу налаштувати два основні параметри:

- кількість логів, що виводяться на кожному кадрі (logsPerFrame),
- тривалість експерименту в секундах (durationSeconds).

Під час експерименту скрипт фіксує значення FPS для кожного кадру, накопичує дані у вигляді масиву, а після завершення тестування обчислює узагальнені метрики, необхідні для подальшого аналізу. Ці значення автоматично виводяться у графічний інтерфейс користувача через TextMeshPro-елементи. Розрахунок середнього FPS:

Для обчислення середнього значення частоти кадрів $\bar{x}$ використовується стандартна формула:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{1}$$

де: x_i – значення FPS на i-му кадрі,

n – загальна кількість кадрів, оброблених протягом експерименту.

Розрахунок дисперсії:

Дисперсія σ^2 дозволяє оцінити відхилення значень FPS від середнього:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2. \tag{2}$$

Розрахунок стандартного відхилення:

Стандартне відхилення σ обчислюється як квадратний корінь із дисперсії:

$$\sigma = \sqrt{\sigma^2}. \tag{3}$$

Цей показник дозволяє оцінити стабільність продуктивності впродовж усього тесту.

Підготовка до порівняльного аналізу (t-критерій Стьюдента):

У подальших етапах дослідження передбачається порівняння двох вибірок – із активним логуванням та з вимкненим – з використанням t-критерію Стьюдента. Формула для розрахунку t-статистики для двох незалежних вибірок наступна:

$$t = \frac{\overline{x}_1 - \overline{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \quad (4)$$

де: $\overline{x}_1, \overline{x}_2$ – середні FPS у двох експериментах (з логами та без),

s_1^2, s_2^2 – дисперсії відповідних вибірок,

n_1, n_2 – кількість кадрів у кожному експерименті.

Результат t-тесту дозволяє оцінити статистичну значущість різниці між двома режимами та визначити, чи є вплив логування на продуктивність системно достовірним.

Для проведення експериментального дослідження було зібрано два білди мобільного застосунку, розробленого в середовищі Unity, під платформу Android. Основна відмінність між ними полягала у кількості логів, які генеруються кожен кадр за допомогою методу Debug.Log.

У межах дослідження було проведено дві окремі серії вимірювань:

- Перша серія – білд із повністю вимкненим логуванням (logsPerFrame = 0);
- Друга серія – білд із активним логуванням, що генерує 1000 записів на кожному кадрі (logsPerFrame = 1000).

Для кожної з конфігурацій було виконано по 5 повторень тривалістю 10 секунд кожне. Це дозволило отримати стабільні вибірки даних для подальшого статистичного аналізу.

Усі вимірювання проводились на одному й тому ж фізичному пристрої з однаковими умовами виконання, з метою забезпечення порівнюваності результатів. Після завершення кожного запуску фіксувались такі метрики, як середнє значення FPS, його мінімальні та максимальні значення, дисперсія та стандартне відхилення частоти кадрів. Отримані результати будуть використані для порівняльного аналізу ефективності виконання застосунку з активним та відключеним логуванням, включно із застосуванням t-критерію Стьюдента. На основі зібраних експериментальних даних було сформовано порівняльну таблицю (Таблиця 1), що відображає ключові статистичні характеристики продуктивності застосунку при різних режимах логування. Узагальнені метрики свідчать про суттєву різницю між конфігураціями.

Зокрема, середнє значення FPS без логів склало 60.36, тоді як при активному логуванні цей показник знизився до 24.14. Відносне зниження продуктивності

Таблиця 1

Порівняльна таблиця

Конфігурація	Середній FPS	Std. Відхилення
Без логів	60.28	11.03
Без логів	60.64	14.49
Без логів	60.05	7.45
Без логів	59.91	6.57
Без логів	60.93	17.95
1000 логів	24.17	3.7
1000 логів	24.09	3.66
1000 логів	24.15	3.81
1000 логів	24.27	3.74
1000 логів	24.01	4.29

становить – 60.01 %. Для перевірки статистичної значущості різниці було використано t-критерій Стьюдента для незалежних вибірок. Отримане значення t-статистики дорівнює 187.6161, а p-значення становить < 0.0001, що свідчить про високий рівень статистичної достовірності результатів (p – рівень статистичної значущості).

На основі проведеного дослідження та отриманих результатів експерименту можна сформулювати низку практичних рекомендацій для розробників Unity-застосунків, орієнтованих на мобільні платформи:

1. Уникати активного логування у фінальних білдах. Систематичне використання Debug.Log у режимі виконання, особливо з високою частотою (100+ викликів на кадр), призводить до суттєвого зниження продуктивності, що може негативно впливати на досвід користувача.

2. Використовувати умовну компіляцію для логування. Варто застосовувати директиви компіляції (#if UNITY_EDITOR, #if DEBUG) для автоматичного вимкнення логування у фінальній збірці.

3. Глобально керувати логуванням через Debug.logger.logEnabled. У випадках, коли необхідно залишити виклики Debug.Log у коді, рекомендується централізовано керувати їхньою активністю за допомогою властивості Debug.logger.logEnabled. Наприклад: Debug.logger.logEnabled = false; Це дозволяє зберегти логи в коді без впливу на продуктивність у релізних білдах.

4. Застосовувати централізовану систему логування. Доцільно винести логіку виводу повідомлень в окремий сервіс або менеджер, який дозволить гнучко керувати рівнями логування та режимами (debug/release).

5. Використовувати профілювання на ранніх етапах. Регулярне використання Unity Profiler або Android Profiler дозволяє своєчасно виявити зайве логування до того, як воно перетвориться на критичну проблему продуктивності.

6. Зберігати діагностичні логи у файл лише за необхідності. У модулях, де потрібна діагностика у production-середовищі, рекомендовано виводити логи у файл або в буфер, а не безпосередньо в консоль.

Висновки. Проведене дослідження підтверджує, що інтенсивне використання логування методом Debug.Log у мобільних Unity-застосунках має істотний негативний вплив на продуктивність системи. У результаті серії експериментів було встановлено, що середній FPS у конфігурації з 1000 логів на кадр знижується більш ніж на 60 % порівняно з конфігурацією без логування. Застосування t-критерію Стьюдента дозволило підтвердити статистичну значущість виявленої різниці ($p < 0.0001$).

Отримані результати дозволяють зробити висновок про недоцільність використання активного логування у фінальних білдах мобільних ігор. Натомість, розробникам рекомендується впроваджувати практики умовного логування, централізації логів та раннього профілювання. Запропонований підхід до вимірювання впливу логування на продуктивність може бути розширений для інших платформ та типів застосунків, що відкриває перспективи для подальших досліджень.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. Unity Technologies. Unity Manual [Електронний ресурс]. Режим доступу: https://docs.unity3d.com/Manual/index.html – Unity 6 User Manual. Дата звернення: 21.04.2025.

2. Unity Technologies. Debug.Log [Електронний ресурс] // Unity Scripting API. Режим доступу: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Debug.Log.htmlНазва з екрану. Дата звернення: 21.04.2025.

3. Unity Technologies. Tools for profiling and debugging [Електронний ресурс]. Режим доступу: https://unity.com/how-to/profiling-and-debugging-tools – Tools for profiling and debugging. Дата звернення: 21.04.2025.

4. Julian, M. Practical Monitoring [Текст] / Mike Julian. 1st ed. Sebastopol: O'Reilly Media, Inc., 2017. 173 с. ISBN 978-1-491-95731-8. С. 84.

5. Dickinson, C. Unity Game Optimization. Third Edition [Текст] / Chris Dickinson. – Birmingham: Packt Publishing, 2017. 270 с. Режим доступу: https://books.google.com.ua/books/about/Unity_Game_Optimization_Third_Edition.html?id=3KBRzQEACAAJ&redir_esc=y Дата звернення: 21.04.2025.

6. Thorn, A. Pro Unity Game Development with C# [Електронний ресурс] / Alan Thorn. Apress, 2014. Режим доступу: https://www.academia.edu/11201938/Pro_Unity_Game_Development_with_C_ Назва з екрану. Дата звернення: 21.04.2025.

7. Klieptsov, A. Performance Logging Test script for Unity v1.0 [Електронний ресурс]. Zenodo, 2025. DOI: https://doi.org/10.5281/zenodo.15254575 Дата звернення: 21.04.2025.

8. Klieptsov, A. Performance Logging Test script for Unity v2.0 [Електронний ресурс]. Zenodo, 2025. DOI: https://doi.org/10.5281/zenodo.15254641 Дата звернення: 21.04.2025.

REFERENCES:

1. Unity Technologies. (2025). *Unity Manual*. Retrieved April 21, 2025, from https://docs.unity3d.com/Manual/index.html

2. Unity Technologies. (2025). *Debug.Log*. Unity Scripting API. Retrieved April 21, 2025, from https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Debug.Log.html

3. Unity Technologies. (2025). *Tools for profiling and debugging*. Retrieved April 21, 2025, from https://unity.com/how-to/profiling-and-debugging-tools

4. Julian, M. (2017). *Practical monitoring: Effective strategies for the real world* (1st ed.). O'Reilly Media.

5. Dickinson, C. (2017). *Unity game optimization* (3rd ed.). Packt Publishing. Retrieved April 21, 2025, from https://books.google.com.ua/books/about/Unity_Game_Optimization_Third_Edition.html?id=3KBRzQEACAAJ

6. Thorn, A. (2014). *Pro Unity game development with C#*. Apress. Retrieved April 21, 2025, from https://www.academia.edu/11201938/Pro_Unity_Game_Development_with_C_

7. Klieptsov, A. (2025). *Performance Logging Test script for Unity v1.0* [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.15254575

8. Klieptsov, A. (2025). *Performance Logging Test script for Unity v2.0* [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.15254641