Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра інженерії програмного забезпечення в енергетиці (ІПЗЕ)

ПРОЦЕСИ РОЗРОБКИ ВБУДОВАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни

Рівень вищої освіти	***Перший (бакалаврський)***
Галузь знань	*12 «Інформаційні технології»*
Спеціальність	*121 «Інженерія програмного забезпечення»*
Освітня програма	*Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці*
Статус дисципліни	*Вибіркова*
Форма навчання	*Очна (денна)*
Рік підготовки, семестр	*III курс, осінній семестр*
Обсяг дисципліни	*120 годин / 4 кредити ЄКТС (лекції – 36 год., практичні заняття – 18 год., самостійна робота – 66 год.)*
Семестровий контроль/ контрольні заходи	*Залік / захист практичних занять, модульні тести, підсумковий тест або захист міні-проєкту*
Розклад занять	*http://rozklad.kpi.ua*
Мова викладання	*Українська*
Інформація про керівника курсу / викладачів	Лектор: к.т.н., доцент Гуменний Дмитро Олександрович, Практичні заняття: Гуменний Дмитро Олександрович
Розміщення курсу	*(LMS кафедри / репозиторій — заповнюється кафедрою)*

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчання та результати навчання

Дисципліна розглядає процеси створення програмного забезпечення для вбудованих систем: програм, що виконуються на мікроконтролерах/SoC, взаємодіють з апаратними інтерфейсами та працюють під обмеженнями пам'яті, продуктивності, енергоспоживання і часу реакції. Курс орієнтований на інженерний підхід: робота з вимогами, системна декомпозиція, моделювання процесів та поведінки цільової платформи (target), архітектура, керування змінами, кооперація з HW/Mech-інженерами на системному рівні, верифікація/валідація та підготовка релізу. Окрема увага приділяється менеджменту проєкту за допомогою інструментів GitHub Projects та Jira.

Метою навчальної дисципліни є формування та закріплення у студентів наступних компетентностей: *(ЗК01) Здатність до абстрактного мислення, аналізу та синтезу; (ЗК02) Здатність застосовувати знання у практичних ситуаціях; (ЗК05) Здатність вчитися і оволодівати сучасними знаннями; (ЗК06) Здатність до пошуку, оброблення та аналізу інформації з різних джерел; (ЗК07) Здатність працювати в команді; (ФК01) Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;*

(ФК02) Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування; (ФК03) Здатність розробляти архітектури, модулі та компоненти програмних систем; (ФК04) Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням та стандартами; (ФК05) Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу; (ФК11) Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення; (ФК12) Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення; (ФК13) Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення.

Зокрема, дисципліна спрямована на формування навичок організації повного контуру розробки embedded-продукту: *від потреб і вимог до архітектури та реалізації, від відтворюваної збірки й CI до тестування (на хості та на цільовому обладнанні — target), релізу та супроводу. Особлива увага — трасованості артефактів, якості, що підтверджується перевірками, ефективній кооперації SW/HW/Mech на системному рівні та практичному використанню інструментів керування проєктом (GitHub Projects, Jira).*

Предмет навчальної дисципліни — процеси створення програмного забезпечення для вбудованих систем: життєвий цикл вбудованого ПЗ; вимоги і трасованість; моделювання процесів та поведінки target-платформи; проєктування; системна кооперація SW-HW-Mech; реалізація; керування версіями, конфігурацією та проєктом (GitHub Projects, Jira); збірка; тестування і інтеграція; реліз; супровід; документація.

Програмні результати навчання, на формування та покращення яких спрямована дисципліна: *(ПРН03) Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення; (ПРН04) Знати і застосовувати професійні стандарти і інші нормативно-правові документи в галузі інженерії програмного забезпечення; (ПРН09) Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення; (ПРН12) Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення; (ПРН14) Застосовувати на практиці інструментальні програмні засоби доменного аналізу, проектування, тестування, візуалізації, вимірювань та документування програмного забезпечення; (ПРН16) Мати навички командної розробки, погодження, оформлення і випуску всіх видів програмної документації; (ПРН19) Знати та вміти застосовувати методи верифікації та валідації програмного забезпечення; (ПРН20) Знати підходи щодо оцінки та забезпечення якості програмного забезпечення; (ПРН22) Знати та вміти застосовувати методи та засоби управління проектами; (ПРН23) Вміти документувати та презентувати результати розробки програмного забезпечення.*

Після завершення дисципліни студент уміє*: виявляти потреби та формулювати вимоги для вбудованих систем (функціональні, часові, ресурсні, інтерфейсні) і визначати критерії приймання; виконувати системну декомпозицію та описувати інтерфейси між SW/HW/Mech-підсистемами з урахуванням запобігання системним помилкам; створювати моделі поведінки target-платформи: контекстну діаграму, компонентну модель, діаграми взаємодій, модель станів (state machine); проєктувати архітектуру firmware з урахуванням тестованості, портованості та обмежень платформи; організувати керування проєктом за допомогою GitHub Projects та Jira (backlog, спринти, дошки, автоматизації); організувати керування конфігурацією і версіями (Git workflow, релізи, тегування);*

налаштувати крос-компіляцію і відтворювану збірку; будувати стратегію перевірок: host-based unit tests, on-target перевірки, інтеграційні сценарії; використовувати практики якості: code review, статичний аналіз, warning-as-error; підготувати реліз-пакет: версія, CHANGELOG, контрольні суми, інструкції build/flash

2. **Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)**

Пререквізити: основи програмування; компоненти програмної інженерії; основи комп'ютерних систем і мереж. Постреквізити: курсові/проєктні роботи з мікроконтролерами, робототехнікою, IoT, промисловими та енергетичними системами; дисципліни з архітектури ПЗ, тестування, DevOps/CI/CD, надійності та безпеки; практика/стажування в embedded/firmware командах.

3. **Зміст навчальної дисципліни**

Розділ 1. Процеси, вимоги та моделювання

Тема 1.1. Контекст embedded-систем та артефакти життєвого циклу
Тема 1.2. Системна кооперація SW/HW/Mech та розподіл вимог
Тема 1.3. Вимоги, критерії приймання та трасованість
Тема 1.4. Моделювання поведінки target-платформи
Тема 1.5. Менеджмент проєкту: GitHub Projects та Jira

Розділ 2. Архітектура, реалізація та якість

Тема 2.1. Архітектура firmware та інтерфейсні контракти
Тема 2.2. Часова поведінка та конкурентність (RTOS)
Тема 2.3. Toolchain, крос-компіляція та аналіз збірки
Тема 2.4. Керування конфігурацією та версіями (Git)
Тема 2.5. Верифікація та валідація: host, on-target, HIL/SIL
Тема 2.6. Якість коду та статичний аналіз
Тема 2.7. CI для embedded та quality gates
Тема 2.8. Налагодження та спостережуваність
Тема 2.9. Реліз, супровід та документація

4. **Навчальні матеріали та ресурси**

Основна література

1. Chacon S., Straub B. Pro Git. 2nd ed. New York : Apress, 2014. 456 p. URL: https://git-scm.com/book/en/v2 (дата звернення: 01.03.2026).

2. White E. Making Embedded Systems: Design Patterns for Great Software. 2nd ed. Sebastopol : O'Reilly Media, 2024. 425 p.

3. Grenning J. W. Test-Driven Development for Embedded C. Raleigh : Pragmatic Bookshelf, 2011. 356 p.

4. MISRA C:2012. Guidelines for the Use of the C Language in Critical Systems. 3rd ed. Nuneaton : MIRA Ltd, 2019. 236 p.

5. SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems. Pittsburgh : Carnegie Mellon University, 2016. URL: https://wiki.sei.cmu.edu/confluence/display/c (дата звернення: 01.03.2026).

6. GitHub Docs: Planning and tracking with Projects. GitHub, Inc. URL: https://docs.github.com/en/issues/planning-and-tracking-with-projects (дата звернення: 01.03.2026).

Навчальний контент

5. Методика опанування навчальної дисципліни(освітнього компонента)

Лекційні заняття

№ з/п	*Назва теми лекції та перелік основних питань (перелік дидактичних засобів, посилання на інформаційні джерела)*
1	***Лекція 1.*** *Тема 1.1. Контекст embedded-систем та артефакти життєвого циклу. Основні питання: місце вбудованих систем у промисловості та побуті; обмеження ресурсів (пам'ять, CPU, енергія, час реакції); артефакти життєвого циклу; відмінності від desktop/web-розробки; огляд стандартів (IEC 61508, ISO 26262 – ознайомчо).*
2	***Лекція 2.*** *Тема 1.2. Системна кооперація SW/HW/Mech та розподіл вимог. Основні питання: визначення меж системи; зацікавлені сторони; use-cases; розподіл вимог (allocation) між SW, HW та Mech; інтерфейсні контракти (ICD); ревізії плат і вплив на SW; спільні рев'ю; запобігання системним помилкам на ранніх етапах.*
3	***Лекція 3.*** *Тема 1.3. Вимоги, критерії приймання та трасованість. Основні питання: формулювання вимог (функціональних, часових, ресурсних); трасованість «вимога → дизайн → тест → реліз»; керування змінами вимог; зв'язок з HW-вимогами.*
4	***Лекція 4.*** *Тема 1.4. Моделювання поведінки target-платформи. Частина 1. Основні питання: роль моделей у embedded; контекстна діаграма; компонентна модель; діаграми взаємодій; моделювання апаратних обмежень target.*
5	***Лекція 5.*** *Тема 1.4. Моделювання поведінки target-платформи. Частина 2. Основні питання: state machines як основний інструмент моделювання embedded; моделювання переривань і таймерів; інструменти (PlantUML, draw.io, Simulink – огляд); верифікація моделей.*
6	***Лекція 6.*** *Тема 1.5. Менеджмент проєкту: GitHub Projects. Основні питання: організація backlog; issues та milestones; project boards (Kanban, таблиці); автоматизації (workflows); зв'язок issues з commits/PR; labels і пріоритети; відстеження прогресу embedded-проєкту.*
7	***Лекція 7.*** *Тема 1.5. Менеджмент проєкту: Jira. Основні питання: проєкти, epic/story/task/bug; Scrum- та Kanban-дошки; спринти та планування; JQL-запити; інтеграція з Git (Bitbucket/GitHub); dashboards і звітність; порівняння GitHub Projects vs Jira для embedded-команд.*
8	***Лекція 8.*** *Тема 2.1. Архітектура firmware та інтерфейсні контракти. Основні питання: шари абстракції; BSP/HAL/драйвери/сервіси/логіка; принципи модульності; Dependency Injection для тестованості; портованість між target-платформами.*
9	***Лекція 9.*** *Тема 2.2. Часова поведінка та конкурентність (RTOS). Основні питання: пріоритети переривань; огляд RTOS (FreeRTOS); задачі, семафори, черги; вимірювання та аналіз таймінгів; дедлайни; WCET (ознайомчо).*
10	***Лекція 10.*** *Тема 2.3. Toolchain, крос-компіляція та аналіз збірки. Основні питання: GCC ARM toolchain; CMake/Make для embedded; map-файл та контроль розміру; профілі Debug/Release; оптимізації; аналіз артефактів збірки.*
11	***Лекція 11.*** *Тема 2.4. Керування конфігурацією та версіями (Git). Основні питання: Git workflow для embedded (trunk-based, Git Flow); code review процес; release branches; тегування; семантичні версії; підтримка ревізій плат у репозиторії.*
12	***Лекція 12.*** *Тема 2.5. Верифікація та валідація: host, on-target, HIL/SIL. Основні питання: піраміда тестів; host-based unit tests з моками HAL; on-target перевірки;*

	інтеграція з апаратурою; HIL/SIL (огляд); планування тестових кампаній у Jira/GitHub.
13	***Лекція 13.** Тема 2.6. Якість коду та статичний аналіз. Основні питання: огляд MISRA C / CERT C; статичний аналіз (cppcheck, clang-tidy); warning-as-error; метрики якості; дефект-менеджмент; відстеження дефектів у Jira/GitHub Issues.*
14	***Лекція 14.** Тема 2.7. CI для embedded та quality gates. Основні питання: GitHub Actions / Jenkins для embedded; матриця збірок (multi-target); автоматизація unit-тестів; артефакти CI; quality gates; мінімальні практики для апаратних тестів; зв'язок CI з project board.*
15	***Лекція 15.** Тема 2.8. Налагодження та спостережуваність. Частина 1. Основні питання: JTAG/SWD; printf- та RTT-логування; трасування (ETM – огляд); crash-аналіз (fault handlers, core dumps).*
16	***Лекція 16.** Тема 2.8. Налагодження та спостережуваність. Частина 2. Основні питання: watchdog; fault-injection (огляд); взаємодія з HW-інженером при діагностиці; практичні сценарії пошуку дефектів у embedded-системах.*
17	***Лекція 17.** Тема 2.9. Реліз, супровід та документація. Частина 1. Основні питання: прошивання (JTAG/UART/USB); OTA-оновлення (огляд); сумісність FW з апаратною ревізією; release notes; hotfix-процес; реліз-пакет (версія, CHANGELOG, контрольні суми, інструкції build/flash).*
18	***Лекція 18.** Тема 2.9. Реліз, супровід та документація. Частина 2. Основні питання: мінімальний набір документів (README, ICD, дизайн-ноти, тест-репорти, реліз-пакет); шаблони у репозиторії; аудит процесу; типові антипатерни; підсумки курсу. Залік: оголошення кінцевої оцінки, залікова контрольна для бажаючих підвищити оцінку.*

Практичні заняття

№ з/п	*Назва теми заняття та перелік основних питань*
1	***Практичне заняття №1.** Системний контекст та кооперація SW/HW/Mech. Основні питання: побудова контекстної діаграми embedded-системи; перелік зовнішніх інтерфейсів (ICD-чернетка); визначення відповідальності SW vs HW vs Mech; ідентифікація потенційних системних помилок на стику дисциплін.*
2	***Практичне заняття №2.** Вимоги, трасованість та налаштування project board. Основні питання: написання вимог із критеріями приймання; створення базової матриці трасування; налаштування проєкту в GitHub Projects (issues, labels, milestones, project board); створення аналогічної структури в Jira (epic, stories, sprint board).*
3	***Практичне заняття №3.** Моделювання поведінки target-платформи. Основні питання: побудова компонентної моделі firmware; створення state machine ключового модуля (PlantUML/draw.io); моделювання взаємодії з апаратними інтерфейсами target-платформи; верифікація моделі проти вимог.*
4	***Практичне заняття №4.** Архітектура firmware та інтерфейсні контракти. Основні питання: виділення шарів (HAL/драйвери/логіка); проєктування інтерфейсних контрактів; ADR для архітектурних рішень; врахування обмежень target та ревізій плат.*
5	***Практичне заняття №5.** Git workflow та ведення спринту. Основні питання: Git workflow; шаблон PR та чекліст code review; правила комітів; зв'язок commits/PR з issues у GitHub Projects; оновлення статусу задач у Jira; ведення спринту.*

6	***Практичне заняття №6.*** *Крос-компіляція та аналіз збірки. Основні питання: налаштування крос-компіляції для target; профілі Debug/Release; формування ELF/HEX/BIN; аналіз map-файлу; контроль розміру; відтворювана збірка.*
7	***Практичне заняття №7.*** *Unit-тести та on-target перевірки. Основні питання: unit-тести на хості з моками апаратних залежностей; запуск у CI; on-target мінімальний тест-харнес; логування; відтворювані сценарії; реєстрація результатів у project board.*
8	***Практичне заняття №8.*** *Статичний аналіз, CI та quality gates. Основні питання: warning-as-error; базова конфігурація cppcheck/clang-tidy; GitHub Actions для embedded; quality gates; відстеження дефектів як issues/bugs у GitHub Projects або Jira.*
9	***Практичне заняття №9.*** *Реліз-пакет та захист міні-проєкту. Основні питання: формування реліз-пакету (версія/тег, CHANGELOG, контрольні суми, інструкції build/flash); тест-звіт; демонстрація повного процесу: вимоги → моделі → архітектура → код → тести → CI → реліз; захист міні-проєкту.*

Самостійна робота студента

№ з/п	*Вид самостійної роботи*	*Кількість годин СРС*
1	*Опрацювання документації MCU/SDK, toolchain, засобів налагодження*	*8*
2	*Вивчення GitHub Projects та Jira: створення проєктів, настройка дошок*	*6*
3	*Побудова моделей поведінки target та інтерфейсних контрактів SW/HW*	*6*
4	*Підготовка вимог, моделей і тестів до практичних занять*	*10*
5	*Виконання міні-проєкту (процес + артефакти + CI + тести + project board)*	*26*
6	*Підготовка до модульних тестів та підсумкового контролю (залік)*	*10*

5. Контрольні заходи

Поточний контроль*: захист практичних занять; перевірка репозиторію та project board; модульні тести за темами.*

Семестровий контроль: *залік у формі підсумкового тесту та/або захисту міні-проєкту (визначається викладачем).*

Політика та контроль

6. Політика навчальної дисципліни (освітнього компонента)

Система вимог, які викладач ставить перед студентом:

- *правила відвідування занять: відповідно до Наказу 1-273 від 14.09.2020 р. заборонено оцінювати присутність або відсутність здобувача на аудиторному занятті, в тому числі нараховувати заохочувальні або штрафні бали. Відповідно до РСО даної дисципліни бали нараховують за відповідні види навчальної активності на лекційних та практичних заняттях.*
- *правила поведінки на заняттях: студент має можливість отримувати бали за відповідні види навчальної активності на лекційних та практичних*

заняттях, передбачені РСО дисципліни. Використання засобів зв'язку для пошуку інформації на гугл-диску викладача, в інтернеті, в дистанційному курсі на платформі Сікорський здійснюється за умови вказівки викладача;

- *політика дедлайнів та перескладань: якщо студент не проходив або не з'явиться на модульний тест (без поважної причини), його результат оцінюється у 0 балів. Перескладання результатів модульних тестів не передбачено.*
- *політика щодо академічної доброчесності: дозволено використання відкритих бібліотек/прикладів лише з коректним посиланням і поясненням власного внеску. Кодекс честі Національного технічного університету України «Київський політехнічний інститут» https://kpi.ua/files/honorcode.pdf.*
- *при використанні цифрових засобів зв'язку з викладачем (мобільний зв'язок, електронна пошта, переписка на форумах та у соцмережах тощо) необхідно дотримуватись загальноприйнятих етичних норм, зокрема бути ввічливим та обмежувати спілкування робочим часом викладача.*

7. Види контролю та рейтингова система оцінювання результатів навчання (РСО)

Поточний контроль: *захист практичних занять; перевірка репозиторію та project board (GitHub Projects / Jira); модульні тести.*
Календарний контроль: *провадиться двічі на семестр як моніторинг поточного стану виконання вимог силабусу. Умовою позитивної атестації є значення поточного рейтингу студента не менше 50 % від максимально можливого на час атестації.*
Семестровий контроль: *залік.*

Умови допуску до семестрового контролю: семестровий рейтинг більше 30 балів.

Таблиця відповідності рейтингових балів оцінкам за університетською шкалою:

Кількість балів	*Оцінка*
95-100	*Відмінно*
85-94	*Дуже добре*
75-84	*Добре*
65-74	*Задовільно*
60-64	*Достатньо*
Менше 60	*Незадовільно*
Менше 30	*Не допущено*

Загальна рейтингова оцінка студента після завершення семестру складається з балів, отриманих за:

- *виконання та захист практичних занять;*
- *виконання модульних тестів (2 модульні тести);*
- *підсумковий контроль*

Практичні заняття (9×4)	*Модульні тести (2×11)*	*Підсумковий контроль (залік)*
36	*22*	*42*

Практичні заняття

Ваговий бал 4. Максимальна кількість балів за практичні заняття – 4 бали × 9 занять = 36 балів.

Практичні заняття здаються з демонстрацією відтворюваності: репозиторій + білд + запуск тестів + project board (GitHub Projects або Jira); для on-target частини – лог/демонстрація на стенді (за наявності). Артефакти (вимоги, моделі, дизайн-ноти, тести, реліз-ноти) мають бути версіоновані у репозиторії.

Критерії оцінювання

- *завдання виконано повністю, вчасно, з відтворюваним результатом – 4 бали;*
- *виконано з незначними зауваженнями – 3 бали;*
- *виконано з суттєвими зауваженнями або із затримкою – 2 бали;*
- *виконано частково – 1 бал;*
- *не виконано – 0 балів.*

Модульні тести

Ваговий бал 11. Максимальна кількість балів – 11 балів × 2 = 22 бали. Модульний тест 1 проводиться після Розділу 1 (теми 1.1–1.5: процеси, вимоги, моделювання, project management). Модульний тест 2 проводиться після Розділу 2 (теми 2.1–2.9: архітектура, тестування, CI, реліз).

Критерії оцінювання

- *повна правильна відповідь – 11 балів;*
- *відповідь з незначними помилками – 8–10 балів;*
- *відповідь з суттєвими помилками – 4–7 балів;*
- *відповідь переважно неправильна – 1–3 бали;*
- *відповідь відсутня або повністю неправильна – 0 балів.*

Підсумковий контроль

***Форма семестрового контролю** – залік. Максимальний бал – 42.*

Підсумковий контроль проводиться у формі залікового тесту або захисту міні-проєкту (визначається викладачем на початку семестру).

Максимальна сума балів складає 100. Для отримання заліку «автоматом» потрібно мати рейтинг не менше 60 балів та виконані умови допуску. Студенти з рейтингом менше 60 балів або ті, хто хоче підвищити оцінку, виконують залікову контрольну роботу (при цьому набрані бали анулюються, оцінка за контрольну є остаточною).

Робочу програму навчальної дисципліни (силабус):

Складено Гуменним Дмитром Олександровичем, к.т.н., доцентом, *Engineering Manager at N-iX Corp.*

Ухвалено кафедрою ІПЗЕ (протокол N 43 від 24 червня 2026 р.)

Погоджено Методичною комісією НН ІАТЕ ім.Ігоря Сікорського (протокол N 9 від 26 червня 2026 р.))