



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	<i>Перший (бакалаврський)</i>
Галузь знань	<i>12 «Інформаційні технології»</i>
Спеціальність	<i>121 «Інженерія програмного забезпечення»</i>
Освітня програма	<i>Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці</i>
Статус дисципліни	<i>Вибіркова</i>
Форма навчання	<i>Заочна</i>
Рік підготовки, семестр	<i>3 курс, осінній семестр</i>
Обсяг дисципліни	<i>120 годин / 4 кредити ЕКТС (лекції – 6 год., практичні заняття – 4 год., самостійна робота – 110 год.)</i>
Семестровий контроль/ контрольні заходи	<i>Залік / захист практичних робіт, модульна контрольна робота (МКР), захист міні-проєкту</i>
Розклад занять	<i>http://rozklad.kpi.ua</i>
Мова викладання	<i>Українська</i>
Інформація про керівника курсу / викладачів	Лектор: к.т.н., доцент Гуменний Дмитро Олександрович, e-mail: dmytro.humenni@edu.kpi.ua , тел. +380958978032 Практичні заняття: к.т.н., доцент Гуменний Дмитро Олександрович
Розміщення курсу	<i>https://campus.kpi.ua</i>

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчення 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 перевітками, 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; проектування; 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); проектувати архітектуру firmware з урахуванням тестованості, портованості та обмежень платформи; організувати керування проектом 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; об'єктно-орієнтоване програмування; алгоритми та структури даних; компоненти програмної інженерії; основи комп'ютерних систем і мереж. **Постреквізити:** курсові/проектні 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. Налаштування 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: 20.08.2026).
2. White E. *Making Embedded Systems: Design Patterns for Great Software*. 2nd ed. Sebastopol : O'Reilly Media, 2024. 428 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*. Nuneaton : MIRA Limited, 2013. 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: 20.08.2026).
6. GitHub Docs: *Planning and tracking with Projects*. GitHub, Inc. URL: <https://docs.github.com/en/issues/planning-and-tracking-with-projects> (дата звернення: 20.08.2026).

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. Процеси розробки <i>embedded</i>-систем, системна кооперація, вимоги та моделювання. Основні питання: особливості життєвого циклу вбудованого ПЗ; ресурсні й часові обмеження; артефакти життєвого циклу; межі системи та зацікавлені сторони; кооперація SW/HW/Mech; <i>allocation</i> вимог; інтерфейсні контракти (ICD); функціональні, часові, ресурсні та інтерфейсні вимоги; критерії приймання; трасованість «вимога → дизайн → тест → реліз»; керування змінами; контекстна і компонентна моделі, діаграми взаємодій та <i>state machine</i>; моделювання обмежень

	target-платформи.
2	Лекція 2. Архітектура <i>firmware</i>, керування проєктом, <i>toolchain</i>, конфігурація та стратегія перевірок. Основні питання: <i>GitHub Projects</i> і <i>Jira</i> (<i>backlog</i>, <i>issues</i>, <i>epic/story/task/bug</i>, дошки, спринти, <i>milestones</i>, зв'язок з <i>commits/PR</i>); шари <i>BSP/HAL/драйвери/сервіси/логіка</i>; модульність, тестованість, портованість та інтерфейсні контракти; крос-компіляція, <i>GCC ARM</i>, <i>CMake/Make</i>, <i>Debug/Release</i>, <i>ELF/HEX/BIN</i> і тар-файл; <i>Git workflow</i>, <i>code review</i>, тегування, семантичне версіонування та підтримка ревізій плат; <i>RTOS</i> і конкурентність (огляд); <i>host-based unit tests</i>, <i>on-target</i> перевірки, <i>HIL/SIL</i>.
3	Лекція 3. Якість, <i>CI</i>, налагодження, реліз і супровід <i>embedded-ПЗ</i>. Основні питання: <i>MISRA C / CERT C</i>; <i>cppcheck</i>, <i>clang-tidy</i>, <i>warning-as-error</i>; дефект-менеджмент; <i>GitHub Actions / Jenkins</i> для <i>embedded</i>; <i>quality gates</i> та артефакти <i>CI</i>; <i>JTAG/SWD</i>, логування, <i>crash</i>-аналіз і <i>watchdog</i>; взаємодія з <i>HW-інженером</i> при діагностиці; сумісність <i>FW</i> з апаратною ревізією; <i>release notes</i>, <i>CHANGELOG</i>, контрольні суми, інструкції <i>build/flash</i>, мінімальний набір технічної документації.

Практичні заняття

№ з/п	Назва теми заняття та перелік основних питань
1	Практичне заняття №1. Системний контекст, вимоги, трасованість і моделювання <i>target</i>. Основні питання: побудова контекстної діаграми <i>embedded-системи</i>; визначення зовнішніх інтерфейсів та відповідальності <i>SW/HW/Mech</i>; формулювання вимог і критеріїв приймання; базова матриця трасування; компонентна модель та <i>state machine</i> ключового модуля; створення структури проєкту в <i>GitHub Projects</i> або <i>Jira</i>.
2	Практичне заняття №2. Відтворюваний процес розробки та реліз <i>embedded-ПЗ</i>. Основні питання: <i>Git workflow</i> і <i>PR/code review</i>; відтворювана крос-збірка; <i>unit-тести</i> на хості та <i>on-target</i> перевірки; статичний аналіз; базовий <i>CI</i> та <i>quality gates</i>; оформлення дефектів; формування реліз-пакету (версія/тег, <i>CHANGELOG</i>, контрольні суми, інструкції <i>build/flash</i>) та перевірка трасованості артефактів.

6. Самостійна робота студента

На самостійну роботу студента відведено 110 годин.

№ з/п	Вид самостійної роботи	Кількість годин СРС

1	Опрацювання тем, винесених на самостійне вивчення	50
2	Підготовка та виконання практичної роботи №1	14
3	Підготовка та виконання практичної роботи №2	14
4	Виконання міні-проєкту (процес + артефакти + репозиторій + project board + CI + тести)	22
5	Підготовка до модульної контрольної роботи (МКР)	6
6	Підготовка до семестрового контролю (залік)	4

Теми, винесені на самостійне опрацювання

Тема 1. Огляд стандартів і моделей життєвого циклу embedded-ПЗ; артефакти процесу та системні review.

Тема 2. Розподіл вимог між SW/HW/Mech, ICD, керування змінами та поглиблена трасованість.

Тема 3. Моделювання target-платформи: state machines, переривання, таймери та верифікація моделей.

Тема 4. GitHub Projects і Jira: автоматизації, JQL/фільтри, dashboards, зв'язок задач з commits/PR.

Тема 5. Часова поведінка firmware: RTOS, пріоритети, черги/семафори, дедлайни та WCET (ознайомчо).

Тема 6. Toolchain і конфігурація: крос-компіляція, мар-файл, оптимізації, Git workflow та ревізії апаратури.

Тема 7. Верифікація і якість: host/on-target/HIL/SIL, MISRA C/CERT C, статичний аналіз та quality gates.

Тема 8. Налагодження, спостережуваність, CI, реліз, супровід і документування embedded-продукту.

7. Контрольні заходи

Поточний контроль: виконання та захист двох практичних робіт; перевірка репозиторію та project board (GitHub Projects / Jira); модульна контрольна робота (МКР), що виконується за методикою домашньої контрольної роботи; перевірка артефактів міні-проєкту.

Семестровий контроль: залік за результатами накопиченого рейтингу. Студенти, які мають рейтинг менше 60 балів або бажають підвищити оцінку, виконують залікову контрольну роботу.

Політика та контроль

8. 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 PCO 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 PCO 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: терміни подання практичних робіт, МКР і міні-проєкту визначаються календарем курсу. Якщо студент не виконав МКР у встановлений термін без поважної причини, результат оцінюється у 0 балів. Можливість повторного виконання контрольних заходів визначається правилами університету та рішенням викладача в межах чинних положень.

- 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.

9. Види контролю та рейтингова система оцінювання результатів навчання (PCO)

Поточний контроль: виконання та захист практичних робіт; перевірка репозиторію та project board; МКР; захист міні-проєкту.

Календарний контроль для заочної форми навчання не проводиться; стан виконання передбачених силябусом робіт контролюється під час настановної та екзаменаційної сесій.

Семестровий контроль: залік.

Умови допуску до семестрового контролю: семестровий рейтинг не менше 40 балів; виконані обидві практичні роботи та МКР; представлений міні-проєкт.

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 40 або не виконані умови допуску	Не допущено

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

- виконання та захист двох практичних робіт;
- виконання модульної контрольної роботи (МКР);
- виконання та захист міні-проєкту.

Практичні роботи (2×25)	МКР	Міні-проєкт
50	20	30

Практичні роботи

Ваговий бал 25. Максимальна кількість балів за практичні роботи – 25 балів × 2 роботи = 50 балів.

Практичні роботи здаються з демонстрацією відтворюваності: репозиторій + build + запуск перевірок + project board (GitHub Projects або Jira). Артефакти (вимоги, моделі, дизайн-ноти, тести, релізні матеріали) мають бути версіоновані у репозиторії.

Критерії оцінювання однієї практичної роботи

- роботу виконано повністю, результат відтворюється, артефакти узгоджені та захищені – 25 балів;
- роботу виконано з незначними зауваженнями – 20–24 бали;
- роботу виконано з суттєвими зауваженнями – 15–19 балів;
- роботу виконано частково – 10–14 балів;
- результат не відповідає мінімальним вимогам або роботу не подано – 0–9 балів.

Модульна контрольна робота (МКР)

Максимальна кількість балів за МКР – 20. МКР виконується за методикою домашньої контрольної роботи після опрацювання матеріалу дисципліни та перевіряє розуміння процесів, вимог, моделювання, конфігурації, V&V, CI, якості та релізу embedded-ПЗ.

Критерії оцінювання

- повна і правильна відповідь / виконання – 18–20 балів;
- відповідь з незначними помилками – 14–17 балів;
- відповідь з суттєвими помилками – 10–13 балів;
- відповідь переважно неправильна або неповна – 1–9 балів;
- відповідь відсутня – 0 балів.

Міні-проєкт

Максимальна кількість балів за міні-проєкт – 30.

Міні-проект демонструє повний процес розробки: вимоги → моделі → архітектура → реалізація/збірка → тести → CI → реліз. Оцінюються повнота й узгодженість артефактів, трасованість, відтворюваність build/test, робота project board та якість захисту. Повне виконання – 30 балів; незначні зауваження – 24–29; часткове виконання – 18–23; суттєво неповний або невідтворюваний результат – 0–17 балів. Міні-проект виконується в межах самостійної роботи студента (22 год) і не є окремою формою звітності за навчальним планом.

Максимальна сума балів складає 100. Для отримання заліку «автоматом» потрібно мати рейтинг не менше 60 балів і виконати умови допуску. Студенти з рейтингом менше 60 балів, а також ті, хто бажає підвищити оцінку, виконують залікову контрольну роботу за матеріалом усього курсу. У такому разі попередньо набрані бали анулюються, а оцінка за залікову контрольну роботу є остаточною.

10. Додаткова інформація з дисципліни (освітнього компонента)

Перелік питань, які виносяться на семестровий контроль (залікову контрольну роботу):

1. Особливості життєвого циклу вбудованого програмного забезпечення; відмінності від desktop/web-розробки.
2. Ресурсні та часові обмеження embedded-систем і їх вплив на процес розробки.
3. Артефакти життєвого циклу вбудованого ПЗ; системні рев'ю.
4. Межі системи та зацікавлені сторони; use-cases в embedded-проекті.
5. Кооперація SW/HW/Mech; розподіл (allocation) вимог між підсистемами.
6. Інтерфейсні контракти (ICD): призначення, склад, супровід.
7. Функціональні, часові, ресурсні та інтерфейсні вимоги; критерії приймання.
8. Трасованість «вимога → дизайн → тест → реліз»; матриця трасування.
9. Керування змінами вимог; вплив ревізій апаратури на програмне забезпечення.
10. Роль моделей у embedded-розробці; контекстна діаграма системи.
11. Компонентна модель firmware та діаграми взаємодій.
12. Модель станів (state machine) модуля; верифікація моделі проти вимог.
13. GitHub Projects: backlog, issues, milestones, дошки, автоматизації.
14. Jira: epic/story/task/bug, спринти, JQL, dashboards.
15. Порівняння GitHub Projects і Jira для embedded-команд.
16. Шари архітектури firmware: BSP, HAL, драйвери, сервіси, прикладна логіка.
17. Модульність, тестованість і портованість firmware; інтерфейсні контракти в коді.
18. RTOS: задачі, пріоритети, черги, семафори; дедлайни та WCET (оглядово).
19. Крос-компіляція: GCC ARM, CMake/Make, профілі Debug/Release.
20. ELF/HEX/BIN та тар-файл; контроль розміру образу.
21. Відтворювана збірка: складові та практики забезпечення.
22. Git workflow для embedded; гілкування, тегування, семантичне версіонування.

23. *Code review: процес, чекліст, типові дефекти.*
 24. *Стратегія перевірок: host-based unit tests, on-target перевірки, HIL/SIL.*
 25. *Моки апаратних залежностей у модульних тестах.*
 26. *MISRA C та CERT C: призначення, застосування, обмеження.*
 27. *Статичний аналіз (cppcheck, clang-tidy); практика warning-as-error.*
 28. *CI для embedded: GitHub Actions/Jenkins, матриця збірок, артефакти, quality gates.*
 29. *Налагодження: JTAG/SWD, логуювання, RTT, трасування.*
 30. *Crash-аналіз, fault handlers, watchdog; діагностика спільно з HW-інженером.*
 31. *Реліз-пакет: версія/тег, CHANGELOG, контрольні суми, інструкції build/flash.*
 32. *Супровід: hotfix-процес, сумісність FW з ревізією плати, OTA-оновлення (оглядово).*
 33. *Мінімальний набір технічної документації embedded-проєкту.*
- Здобувачі можуть звертатися до викладача за консультацією щодо виконання практичних робіт, міні-проєкту та підготовки до МКР у робочий час; комунікація – через засоби, узгоджені на настановній сесії.*
- Результати навчання, набуті у неформальній та інформальній освіті (профільні курси з Git, CI/CD, embedded-розробки, підтверджені сертифікатами), можуть бути зараховані за рішенням викладача у рахунок окремих видів робіт з дисципліни відповідно до чинного положення КПІ ім. Ігоря Сікорського про визнання таких результатів.*

Робочу програму навчальної дисципліни (силабус):

Складено Гуменним Дмитром Олександровичем, к.т.н., доц.

Ухвалено кафедрою: _____ (протокол № __ від __.__.20__)

Погоджено методичною комісією: _____ (протокол № __ від __.__.20__)