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

Department of Computer Engineering

Testing and quality control (QA) of embedded systems

Working program of the academic discipline (Syllabus)

Details of the academic discipline

Level of higher education	***First (bachelor)***
Scope of knowledge	*12 Information technologies*
Specialty	*123 Computer Engineering*
Educational program	*Computer Engineering*
Discipline status	*Selective*
Form of education	*daytime*
Grade, semester	*4-th year, spring*
Scope of the discipline	*4 credits, 120 hours* *Lectures 18 (36 hours), Laboratory 9 (18 hours)*
Semester control/control measures	*Test*
Lessons schedule	*According to the schedule for the spring semester of the current academic year at http://rozklad.kpi.ua*
Language of teaching	*Ukrainian*
Information about the head of the course / teachers	*Lecturers: Ph.D., docent, Klymenko Iryna Anatoliivna, ikliryna@gmail.com , assistant of the department of OT, Taraniuk Victoria Anatoliivna taranyuk.viktoria@gmail.com;* *Laboratory staff: Kaplunov Artem Volodymyrovych, art.kaplunov@gmail.com , assistant of the department of OT, Taraniuk Victoria Anatoliivna taranyuk.viktoria@gmail.com;*
Placement of the course	*The training course is hosted on the "Sikorsky" distance learning platform in the Google Workspace for Education environment: https://classroom.google.com/c/NTM3NTgzODE2ODkx?cjc=2opss67*

Program of educational discipline

1. Description of the educational discipline, its purpose, subject of study and learning outcomes

The purpose of the QA/Embedded discipline is to study the theoretical foundations and technologies of quality assurance of IT products (Quality Assurance, QA), in particular in the field of development of embedded systems; achieving the goals of product quality control through testing and quality analysis at various stages of the product development life cycle, studying the basics of compiling test documentation according to global QA standards.

The discipline enhances the following general and professional competencies:

- ЗК2 - ability to learn and acquire modern knowledge;
- ФК1 - ability to apply the legislative and regulatory framework, as well as national and international requirements, practices and standards in order to carry out professional activities in the field of computer engineering.
- ФК2 - ability to use modern methods and programming languages for development of algorithms and software.
- ФК5 - ability to use the tools and systems of design automation to development of components of computer systems and networks, Internet applications, cyber-physical systems, etc.
- ФК9 - ability to systematically support, use, adapt and operate existing information technologies and systems.
- ФК13 - ability to solve problems in the field of computer and information technologies, determine the limitations of these technologies.
- ФК14 - ability to design systems and their components taking into account all their aspects. life cycle and task, including creation, configuration, operation, maintenance and disposal.
- ФК16 - ability to design, implement and maintain the high-performance parallel and distributed computer systems and their components using FPGAs, modules and CAD systems.

In accordance with the above, strengthened general and professional competencies will provide the following learning outcomes:

- form a system of knowledge and skills in the field of quality assurance of IT products (Quality Assurance, QA), in particular in the field of development of embedded systems;
- acquire knowledge of fundamental concepts, paradigms and basic principles of QA;
- receive concepts and qualification bases of the profession of quality engineer (QA engineer);
- learn to monitor compliance with development standards of embedded systems and software for them and use unified principles and methods of error detection and prevention;
- get acquainted with the basic principles of testing embedded systems;
- get acquainted with the basic principles of building, configuring and testing the environment of embedded systems;
- get acquainted with the principles of deploying and functioning of the Linux operating system on processors for embedded systems;
- study the main tools of the operating system to ensure the interaction of embedded system devices with the external environment;
- learn how to assemble, debug and use Linux-type operating systems on ARM processors;
- learn how to compile executable programs from OpenSource source codes;
- learn how to configure the Linux operating system and computer network;
- learn how to deploy an operating system (Linux kernel, U-Boot, BusyBox) for ARM Cortex A8 processor architecture by compiling the source code;
- learn how to flash processor microcircuits on a board over a computer network, in particular in remote mode;
- learn to use various interfaces, as well as terminal emulation programs for communication and debugging of built-in devices;
- learn how to test your own product based on the BeagleBone Black platform;
- develop basic practical programming skills in low-level languages, in particular C, for the ARM processor core
- get practical experience in finding errors in software and hardware provision of embedded systems;
- learn how to troubleshoot computer network problems;
- learn how to compile test documentation;
- gain experience in teamwork;

- gain experience defending their own decisions in a professional discussion and presenting the results of their own developments;
- gain experience using computer engineering competencies in practice;
- receive basic training for the ISTQB specialist QA certification;

2. Pre-requisites and post-requisites of the discipline (a place in the structural and logical scheme of training according to the relevant educational program)

When studying the discipline "Technology of testing (QA) of embedded systems" it is advisable to use the knowledge gained during the study of previous disciplines: "Introduction to the Linux operating system", "Programming technologies on FPGA" "Computer logic", "Theory of electric circuits and signals", "Computer electronics", "Computer circuitry", "Computer architecture. Part 1. Control and arithmetic devices", "Computer architecture. Part 2. Processors", "Programming", "Algorithms and data structures", "Programming technologies in C for embedded systems" (selective), "Foreign language".

The discipline is basic for the courses: "Intelligent systems design technologies" (elective), "Infrastructural IT project management" (elective), "Computer modeling", "Organization of computing processes", "Computer architecture. Part 3. Microprocessor devices", "Computer architecture. Course work", "System programming", "Computer networks", "Computer systems", "Technologies of designing computer systems" (selective), "Research and design of parallel systems" (selective), "Technologies of parallel programming computer systems" (selective).

3. Structure of the credit module

Introduction
Chapter 1. Introduction to the discipline of QA/Embedded
Topic 1.1. Basic aspects of QA in modern technologies.
Topic 1.2. Trends in the development of modern embedded systems.
Topic 1.3. Basics of using Linux OS in embedded systems technologies. Installing the Linux OS on a PC. Working with the command line. Performing basic operations with files. Updating and installing software.
Chapter 2. Creation and testing of a network environment based on the TCP/IP protocol stack
Topic 2.1. The concept of a global network. OSI network model. An example of application layer protocol operation. Overview of the physical layer of the OSI model.
Topic 2.2. The channel layer of the OSI model. Concept and format of MAC address. Ethernet frame format. ARP protocol.
Topic 2.3. The concept of QA Embedded Testing on the example of the ARP protocol. Development of ARP Test Case Topic 2.4. Network layer of the OSI model. IP addresses. Classes of IP addresses. IP address format.
Topic 2.4. DHCP protocol. DHCP server configuration.
Topic 2.5. Development of test documentation. Development of DHCP Test Plan, DHCP TestCase. DHCP Defect Report.
Topic 2.6. DNS service. NAT service. Transport level protocols TCP, UDP. Development of DHCP + NAT Test Plan test documentation. DHCP + NAT TestCase. DHCP + NAT Defect Report Topic 2.7. Encapsulation and decapsulation on the example of an application layer protocol.
Topic 2.8. Practical work. HTTP testing levels HTTP session. Troubleshooting of HTTP protocol.
Chapter 3. Development of an embedded device
Topic 3.1. Features of assembly, debugging and use of Linux-type operating systems on AVRm7 processors with Von Neumann architecture for implementation of embedded systems and computers. Topic 3.2. Collection of bootloader (u-boot), kernel (Kernel), file system (RootFs) executables for AVRv7 processor.
Topic 3.3. Stages of deployment of the operating system. Architecture initialization processes in the kernel space of the operating system on the example of the architecture of x86 and AVRv7 processors.
Topic 3.3. Processor firmware over the network. Implementation of a separate console using the UART protocol.
Chapter 4. Theory of QA

Topic 4.1. Introduction to the basics of embedded systems, definition of an embedded system and problem statement. Giving examples of modern embedded systems.
Topic 4.2. Basic types of test documentation and modern practices for writing test cases and defect reports. Basic testing processes.
Topic 4.3. Fundamental approaches to software development (SDLC), understanding of development processes and the ability to build test processes depending on the development process.
Topic 4.4. Definition of the main terms, requirements, testing objects. Functional and quality requirements. Testing Requirements.
Topic 4.5. Levels of testing, definition of test objects at the level of code, integration modules and systems. Definition of system verification and validation.
Topic 4.6. Types of testing. A view of the system from the perspective of business, user and operational tasks. Definition of functional and non-functional types of testing.
Topic 4.7. Embedded system performance testing. Risk-based testing Topic 4.8. Test design techniques

Chapter 5. Basics of developing self-tests

Topic 5.1. An overview of the Pytest framework for embedded system testing.
Topic 5.2. Demonstration of the automation of embedded system testing using the example of a traffic analyzer.

Chapter 6. Development directions of embedded systems technologies in the conditions of the 4th industrial revolution (Industry 4.0/IoT)

Topic 6.1. Overview of IOT architecture and Edge intelligence.
Topic 6.2. An overview of technologies related to the 4th industrial revolution.

4. Literature

4.1. Basic literature

1. Testing and quality control (QA) of embedded systems [Electronic resource]: tutorial for bachelor's study degree applicants under the educational program "Computer systems and networks" specialty 123 "Computer engineering" / I. A. Klymenko,V. A. Taraniuk, V. V. Tkachenko, O.O. Pysarchuk; Igor Sikorsky KPI. – Electronic text data (1 file: XX MB). – Kyiv: Igor Sikorsky KPI, 2022. – 58 p.
2. Lee Copeland. A Practioners guide to software test design – 2013, Artech House. – 312 p. https://dahlan.unimal.ac.id/files/ebooks/2004%20A%20Practitioner's%20Guide%20to%20Software%20Test%20Design_Good.pdf

4.2. Additional literature

1. ISTQB - SYLLABUS ® Advanced Level Test Analyst (CTAL-TA). https://istqb.in/foundation/syllabus-and-resources-2018-version-applicable-from-5th-june-2019/syllabus.
2. ISTQB - SYLLABUS ® Certified Tester Foundation Level (CTFL) https://istqb.in/foundation/syllabus-and-resources-2018-version-applicable-from-5th-june-2019/syllabus.
3. ISTQB - SYLLABUS ® Certified Tester Performance Testing (CT-PT) https://istqb.in/foundation/syllabus-and-resources-2018-version-applicable-from-5th-june-2019/syllabus.
4. Karl Wiegers, Joy Beatty. Software Requirements (Developer Best Practices) 3rd Edition. – 2013, Microsoft Press. Software Requirements, 3rd Edition [Book] (oreilly.com).
5. Black Rex. Advanced Software Testing by Rex Black Advanced Software Testing Vol. 1: Guide to the ISTQB Advanced Certification as an Advanced Test Analyst (Rockynook Computing). – 2015. – 352 p. https://www.amazon.de/-/en/Rex-Black/dp/1937538680
6. ISTQB - SYLLABUS ® Advanced Level Test Analyst (CTAL-TA). https://istqb.in/foundation/syllabus-and-resources-2018-version-applicable-from-5th-june-2019/syllabus.
7. ISTQB - SYLLABUS ® Certified Tester Foundation Level (CTFL) https://istqb.in/foundation/syllabus-and-resources-2018-version-applicable-from-5th-june-2019/syllabus.
8. ISTQB - SYLLABUS ® Certified Tester Performance Testing (CT-PT) https://istqb.in/foundation/syllabus-and-resources-2018-version-applicable-from-5th-june-2019/syllabus.
9. Karl Wiegers, Joy Beatty. Software Requirements (Developer Best Practices) 3rd Edition. – 2013, Microsoft Press. Software Requirements, 3rd Edition [Book] (oreilly.com).

10. Black Rex. Advanced Software Testing by Rex Black Advanced Software Testing Vol. 1: Guide to the ISTQB Advanced Certification as an Advanced Test Analyst (Rockynook Computing). – 2015. – 352 p. https://www.amazon.de/-/en/Rex-Black/dp/1937538680

4.3. Information resources

1. A course of video lectures - on the "Sikorsky" distance learning platform in the Google Workspace for Education environment: https://classroom.google.com/c/NTM3NTgzODE2ODkx?cjc=2opss67
2. Subprocess management — Python 3.11.1 documentation https://docs.python.org/3/library/subprocess.html.
3. About fixtures — pytest documentation. https://docs.pytest.org/en/stable/explanation/fixtures.html#what-fixtures-are
4. How to use fixtures — pytest documentation. https://docs.pytest.org/en/stable/how-to/fixtures.html#how-to-fixtures

5. Laboratory work

The purpose of the laboratory work is to acquire practical skills in working with equipment based on the BeagleBone Black platform, independently creating an embedded system, setting up the environment, gaining experience in testing the embedded operating system taking into account the setting of the test environment, in the framework of which issues are consideredv:

- network troubleshooting (Network Troubleshooting);
- configuration of the Linux operating system and computer network;
- deployment of the operating system (Linux kernel, U-Boot, BusyBox) for the architecture of ARM Cortex A8 processors by compiling the source code; various methods of flashing the processor microcircuit on the board;
- testing of own product based on the BeagleBone Black platform;
- verification of functional and non-functional attributes of software and hardware in an independently created embedded system;
- creation of tests for verification and confirmation of built-in software and hardware in accordance with the client's requirements;
- using Git and GitLub for version control and work in the development team.

Topics of laboratory work

Laboratory work 1. Introductory lesson. Working in the Linux command line. Joining the Gitea KPI common project management environment.
Laboratory work 2. Configuration and testing of obtaining MAC addresses in the local domain by means of the ARP protocol. Development of ARP Test Case.
Laboratory work 3. Configuring and testing the DHCP network protocol for obtaining an IP address by the built-in system. Development of test documentation. Development of DHCP Test Plan, DHCP TestCase. DHCP Defect Report.
Laboratory work 4. Configuring and testing DNS and NAT services. Development of DHCP test documentation, NAT Test Plan. DHCP, NAT TestCase. DHCP, NAT Defect Report.
Laboratory work 5. Configuration and testing of the transport layer based on TCP and UDP protocols. Network traffic analysis using the WireShark utility.
Laboratory work 6. Creating your own device for testing the network environment on the board Flashing the bbb board in different ways. Firmware in remote mode. Working with the UART interface.
Laboratory work 7. Development of a kernel module for testing network traffic of an embedded system based on the TCP dump implementation. Visualization of results on the Host device.
Laboratory work 8. Development of autotests for testing the network environment of an embedded device in Python.

6. Independent work of the student

Types of independent work:
− preparation for classroom classes (0.5 hours x 18 lectures = 9 hours);
− preparation for express tests (recommended 1-2 hours x 2 tests = 2-4 hours)
− preparation and processing of calculations based on primary data obtained in laboratory classes, performing laboratory work, solving problems, sending results to GitLab (recommended 2-3 hours x 8 laboratory works = 16 - 24 hours);
− execution of modular control work (2 Module Test x 4-8 hours = 8-16 hours).

– Policy and control

5. Policy of academic discipline (educational component)

Deadlines are set for the performance of laboratory work and modular control work.

Performance of laboratory work outside of the established deadlines is accompanied by penalty points, which are deducted from the grade for the protocol. Modular control work is not accepted beyond the set time.

Penalty points are issued for: untimely submission of laboratory work. The number of penalty points is no more than 10.

Bonus points are awarded for: active participation in lectures; completing current homework, keeping a summary, preparing a message with a presentation on one of the topics of the SRS discipline, etc. The number of bonus points is no more than 10.

Some lecture topics are accompanied by short express tests (for 15 minutes), which include the material of the studied topic and questions that are asked for independent study. The points obtained for the test are included in the semester rating. Current tests are not retaken.

The performance of each laboratory work is preceded by the completion of an individual task and its preparation in the form of a protocol. A student who came to class without a completed protocol is not allowed to do laboratory work. In the first stage, the student defends the results obtained during the performance of an individual task for laboratory work, in the second stage - defends the theory through an oral survey or test. Most of the laboratory works are accompanied by tests to evaluate the studied theoretical and practical material for the laboratory work. The points obtained for the performance of laboratory work, for the test and for the protocol are included in the assessment for the laboratory work. Testing is carried out in a laboratory session after checking the results of laboratory work. A student who has not completed the individual task before the laboratory work and the test is not admitted.

Performance of laboratory work is mandatory for admission to semester control. The condition of admission to the semester control is the enrollment of all laboratory works and a starting rating of at least 30 points.

A modular test is written during a lecture session without the use of aids (mobile phones, tablets, etc.); the result is forwarded to the corresponding Google Drive directory via a Google form.

The modular control paper is not rewritten in case of a negative grade, a negative grade for the MCR (less than 9 points (<60%)) is equal to 0 points, in this case the modular control work is not counted.

The grade that a student can receive for each laboratory work and for each modular control work is given in table 1 of semester work evaluations, chapter 8 of the syllabus.

Thus, the minimum grade that a student must receive for admission to the semester exam is 60 points, the maximum is 100 points for the completion of all current works for the semester.

Applicants who have fulfilled all admission requirements (completed all laboratory work) and have a rating of less than 60 points, as well as applicants who wish to improve their rating, have the opportunity to pass a semester test in the form of a credit test at the last class on the schedule .

In the case of performance of credit control work, the rating is defined as the sum of points for credit control work and points for individual semester tasks.

The individual work of the student related to the performance of laboratory work is included in the individual semester tasks. The maximum number of points for individual work per semester is 60 points. The maximum mark for the test is 40 points. In this way, the applicant has the opportunity to increase his rating by writing a final test and adding additional points to the number of points received during the semester for individual semester work.

After completion of the credit control work, if the grade for the credit control work is higher than the rating, the applicant receives a grade based on the results of the credit control work. If the grade for the final test is lower than the rating, the applicant's previous rating (with the exception of points for the semester individual task) is canceled and he receives a grade based on the results of the final test. This option forms a responsible attitude of the applicant towards making a decision on the completion of the credit control work, forces him to critically assess the level of his training and carefully prepare for the credit.

6. Types of control and rating system for evaluating learning outcomes (RSE)

The student's semester rating from the credit module is calculated based on a 100-point scale. The rating consists of the points that the student receives for completing 8 laboratory works R_L, two modular control works R_{MCW} and expert tests R_{ET}.

The maximum number of points for laboratory work is 60 points, i.e $R_L = 60$.

The generalized criteria for evaluating laboratory work are as follows:

- the timeliness of the preparation of the protocol for the laboratory session, completeness of the theoretical or practical task in the protocol, the protocol is posted on GitLab on time;
- correct functioning of the developed models on software or hardware, demonstration of own repository on GitLab with laboratory work materials and availability of commits;
- a survey on the subject of laboratory work for crediting the practical part of the work, protection of the results obtained in the work, answers to additional theoretical questions of the teacher, completeness of the report/protocol on the work on GitLab.

A detailed approach to the assessment of each laboratory work is given in Table 1.

Table 1. Details of the evaluation of each laboratory work

The name of the class	Form of control	Scores	Admission to the exam by automatic evaluation	Total points
Laboratory work 1.	Entrance test	4	2	4
Laboratory work 2	Completing the task	3	5	8
	Polling	3		
	Protocol on GitLab	2		
Laboratory work 3	Completing the task	3	5	8
	Polling on QA	3		
	Protocol on GitLab	2		
Laboratory work 4	Completing the task	3	5	8
	Polling on QA	3		
	Protocol on GitLab	2		
Laboratory work 5	Completing the task	3	5	8
	Polling	3		
	Protocol on GitLab	2		
Laboratory work 6	Completing the task	3	5	8
	Polling	3		
	Protocol on GitLab / demonstration	2		
Laboratory work 7	Completing the task	3	5	8
	Polling	3		
	Protocol on GitLab / demonstration	2		
Laboratory work 8	Completing the task	3	5	8
	Polling	3		

	Protocol on GitLab / demonstration	2		
Number of points for individual work				60
Express tests at lectures	2 x 5	10	5	10
Modular control work	MCW1 (Тест)	15	9	15
	MCW2	15	9	15
Total points		100	60	100

The maximum number of points per MCW R_{MCW} = 2x15 = 30 points.

MCW1 is conducted in the form of automated testing on the Google Workspace for Education platform. The test consists of 60 questions R_{MCW_2} = 0,25x60 = 15 points

Modular control work MCW2 is performed independently according to an individual task. MCW2 assessment criteria at four levels:

- correct and meaningful answer with explanations in the terms of the subject area: 13 - 15 points;
- correct answer, incomplete explanations: 11 - 12 points;
- the answer contains errors: 9 - 10 points;
- the answer contains significant errors, there are no explanations: 4-8 points;
- no answer: 0 points.

The score for MCW2 is reduced by:

- incorrect registration;
- lack of comments in meaningful terms;
- lack of explanations during calculations.

The maximum number of points for express tests is 10 points, tests are conducted during lectures in the form of automated testing on the Google Workspace for Education platform.

The maximum number of points for the credit control work is R_T = 40 points.

The credit control work is conducted in the form of automated testing on the Google Workspace for Education / moodle platform, consisting of selected questions that were during the semester in MCW, express tests, and defenses of laboratory works. The maximum score for the credit control work R_T = 40 points.

Calendar certification of students (for 8 and 14 weeks of semesters) in the discipline is carried out according to the value of the student's current rating at the time of certification. If the value of this rating is at least 50% of the maximum possible at the time of certification, the student is considered certified. Otherwise, the attestation information is marked as "uncertified".

The number of points a student receives per semester is determined by the formula

$$\mathbf{R = R_L + R_{MCW} + R_{ET}.}$$

The maximum number of points per semester does not exceed RS = 100.

Taking into account the received sum of points, the final grade is determined according to table 3.

If a student writes a test paper, the number of points the student receives per semester is determined by the formula

$$\mathbf{R = R_{IP} + R_T}$$

where, $\mathbf{R_{IP} = R_L}$.

The maximum number of points per semester does not exceed R = 100.

Taking into account the received sum of points, the final grade is determined by table 3.

Table 2. Determination of the semester grade

Scores	*Rating*
100-95	Perfectly
94-85	Very good

84-75	Good
74-65	Satisfactorily
64-60	Enough
Less than 60	Unsatisfactorily
Admission conditions not met	Not allowed

Working program of the academic discipline (syllabus)
Made by, Ph.D., associate professor, professor of the department of CE
Klymenko Iryna Anatoliivna, assistant of the department of CE,
Taraniuk Victoria Anatoliivna.

Approved by the Department of Computing Engineering (Protocol No.10 dated 25.05.2022 p.)

Agreed by the methodical commission of FICT (protocol No.10 dated 09.06.2022 p.)