Universitatea Naţională de Ştiinţă şi Tehnologie Politehnica Bucureşti
Facultatea de Electronică, Telecomunicaţii şi
Tehnologia Informaţiei

COURSE DESCRIPTION

1. Program identification information

1.1 Higher education institution	**National University of Science and Technology Politehnica Bucharest**
1.2 Faculty	**Electronics, Telecommunications and Information Technology**
1.3 Department	**Electronic Devices, Circuits and Architectures**
1.4 Domain of studies	Electronic Engineering, Telecommunications and Information Technology
1.5 Cycle of studies	Masters
1.6 Programme of studies	Advanced Computing in Embedded Systems

2. Date despre disciplină

2.1 Course name (ro) (en)				Dezvoltare software şi testare Software Development Process and Testing			
2.2 Course Lecturer				Conf. Dr. Radu Hobincu			
2.3 Instructor for practical activities				Conf. Dr. Radu Hobincu			
2.4 Year of studies	1	2.5 Semester	2	2.6. Evaluation type	V	2.7 Course regime	Ob
2.8 Course type		S	2.9 Course code	2		2.10 Tipul de notare	Nota

3. Total estimated time (hours per semester for academic activities)

3.1 Number of hours per week	4	Out of which: 3.2 course	2	3.3 seminary/laboratory	2
3.4 Total hours in the curricula	56	Out of which: 3.5 course	28	3.6 seminary/laboratory	28
Distribution of time:					hours
Study according to the manual, course support, bibliography and hand notes Supplemental documentation (library, electronic access resources, in the field, etc) Preparation for practical activities, homework, essays, portfolios, etc.					75
Tutoring					5
Examinations					0
Other activities (if any):					14

3.7 Total hours of individual study	94.00
3.8 Total hours per semester	150
3.9 Number of ECTS credit points	6

4. Prerequisites (if applicable) (where applicable)

4.1 Curriculum	Programming knowledge in C/C++ languages, Data structures and algorithms
4.2 Results of learning	Computer usage abilities, technical competencies.

5. Necessary conditions for the optimal development of teaching activities (where applicable)

5.1 Course	Video projector and Internet access.

5.2 Seminary/ Laboratory/Project	Laboratory with Internet access for students and video projector.

6. General objective *(Reffering to the teachers' intentions for students and to what the students will be thought during the course. It offers an idea on the position of course in the scientific domain, as well as the role it has for the study programme. The course topics, the justification of including the course in the currcula of the study programme, etc. will be described in a general manner)*

The overall objective is to familiarize the student with the complete software development process, which is not only about writing code, but also about all related processes: compilation, testing, documentation and implementation. Thus, students will study how to effectively use version control systems, what options they have regarding efficient build tools, how to test applications using unit tests, how to generate documentation for an application, and how to use container systems for integration continuous, cross-compilation and deployment.

7. Competences *(Proven capacity to use knowledge, aptitudes and personal, social and/or methodological abilities in work or study situations and for personal and proffesional growth. They refflect the empolyers requirements.)*

Specific Competences	C3. Application of basic knowledge, concepts and methods regarding the architecture of computing systems, microprocessors, microcontrollers, programming languages and techniques (3/4)
Transversal (General) Competences	-

8. Learning outcomes *(Synthetic descriptions for what a student will be capable of doing or showing at the completion of a course. The learning outcomes reflect the student's acomplishments and to a lesser extent the teachers' intentions. The learning outcomes inform the students of what is expected from them with respect to performance and to obtain the desired grades and ECTS points. They are defined in concise terms, using verbs similar to the examples below and indicate what will be required for evaluation. The learning outcomes will be formulated so that the correlation with the competences defined in section 7 is highlighted.)*

Knowledge	*The result of knowledge aquisition through learning. The knowledge represents the totality of facts, priciples, theories and practices for a given work or study field. They can be theoretical and/or factual.* Following this course, students will understand the entire tool chain needed to bring software from design to implementation and will be able to discern where it is not working and needs improvement.
Skills	*The capacity to apply the knowledge and use the know-how for completing tasks and solving problems. The skills are described as being cognitive (requiring the use of logical, intuitive and creative thinking) or practical (implying manual dexterity and the use of methods, materials, tools and intrumentation).* Specifically, students will know how to use the Git version control system as well as the Gitlab platform in complex scenarios, they will be able to write their own unit tests using the Google Test library, they will be able to create efficient complex build environments using GNU Make or CMake, they will be able to generate documentation for code written using Doxygen and will be able to integrate all these tools into continuous integration systems using the Docker container system.

Responsability and autonomy	*The student's capacity to autonomously and responsably apply their knowledge and skills.* After this course, students will be able to develop autonomously, the entire software development chain, starting from writing the code, up to implementation in production.

9. Teaching techniques *(Student centric techniques will be considered. The means for students to participate in defining their own study path, the identification of eventual fallbacks and the remedial measures that will be adopted in those cases will be described.)*

The teaching will be done in an interactive way, through practical examples, presented in real time, related to complex software projects.

10. Contents

COURSE

Chapter	Content	No. hours
1	Version Control Systems - Git	8
2	Using the Gitlab platform	2
3	Build systems - GNU Make and CMake	4
4	Unit tests - Google Test and Google Mock	2
5	Calculation of test coverage rate - GCOVR	2
6	Generating Documentation - Doxygen	2
7	Container systems - Docker	2
8	Cross-compile using GCC/G++	2
9	Continuous Integration (CI) in Gitlab	4
	Total:	28

Bibliography:

1. Git Reference Documentation - https://git-scm.com/docs
2. Scott Chacon and Ben Straub, Pro Git, 2nd Edition, Apress Publishing, 2024 - https://github.com/progit/progit2/releases/download/2.1.422/progit.pdf
3. Gitlab Documentation - https://docs.gitlab.com/
4. GNU Make documentation - https://www.gnu.org/software/make/manual/make.pdf
5. CMake documentation - https://cmake.org/cmake/help/latest/
6. Google Test Documentation - https://google.github.io/googletest/
7. gcovr, user guide - https://gcovr.com/en/stable/guide.html
8. Doxygen Manual - https://www.doxygen.nl/manual/index.html
9. Docker Documentation - https://docs.docker.com/get-started/overview/
10. Ruvinda Dhambarage, A master guide to Linux cross compiling - https://ruvi-d.medium.com/a-master-guide-to-linux-cross-compiling-b894bf909386
11. Gitlab CI Documentation - https://docs.gitlab.com/ee/ci/

LABORATORY

Crt. no.	Content	No. hours

1	Implementation of the build system	1
2	Software project development	14
3	Writing unit tests	7
4	Annotation and documentation generation	2
5	Building the Docker image for compilation and cross-compilation	2
6	Setting up the continuous integration system in Gitlab	2
	Total:	28
Bibliography:		

11. Evaluation

Activity type	11.1 Evaluation criteria	11.2 Evaluation methods	11.3 Percentage of final grade
11.4 Course	Assimilation of theoretical knowledge.	Quiz on the Moodle platform.	50%
11.5 Seminary/laboratory/project	The student understood and used the techniques in the development of the project.	Accumulation of points associated with tasks on Gitlab, points awarded upon successful completion of tasks.	50%
11.6 Passing conditions			
The student will obtain: • at least one point associated with a "development" type task; • at least one point associated with a "test" type task; • at least one point associated with a "process" type task; • at least one point associated with a "documentation" type task; The student will obtain a minimum of 50% of total points.			

12. Corroborate the content of the course with the expectations of representatives of employers and representative professional associations in the field of the program, as well as with the current state of knowledge in the scientific field approached and practices in higher education institutions in the European Higher Education Area (EHEA)

In the context of software engineering, companies develop increasingly complex applications, which require an increasingly automated development environment. The lack of such an environment leads to serious performance issues that significantly increase the time required to fully implement the solution. Thus, discipline is critical for any engineer working in the field, equipping them with a body of knowledge that will enable them to be effective throughout the development process.

Date	Course lecturer	Instructor(s) for practical activities
25.09.2025	Conf. Dr. Radu Hobincu	Conf. Dr. Radu Hobincu

Universitatea Națională de Știință și Tehnologie Politehnica București
Facultatea de Electronică, Telecomunicații și
Tehnologia Informației

Date of department approval	Head of department
26.09.2025	Prof. Dr. Claudius Dan
Date of approval in the Faculty Council	Dean
26.09.2025	Prof. Dr. Mihnea Udrea