Universitatea Naţională de Ştiinţă şi Tehnologie Politehnica Bucureşti

Facultatea de Electronică, Telecomunicaţii şi

Tehnologia Informaţiei

FIŞA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	**Universitatea Naţională de Ştiinţă şi Tehnologie Politehnica Bucureşti**
1.2 Facultatea	**Electronică, Telecomunicaţii şi Tehnologia Informaţiei**
1.3 Departamentul	**Dispozitive, Circuite şi Arhitecturi Electronice**
1.4 Domeniul de studii	Inginerie Electronică, Telecomunicaţii şi Tehnologii Informaţionale
1.5 Ciclul de studii	Licenţă
1.6 Specializarea	Microelectronică, Optoelectronică şi Nanotehnologii

2. Date despre disciplină

2.1 Denumirea disciplinei (ro) (en)				Programarea calculatoarelor şi limbaje de programare 1			
2.2 Titularul activităţilor de curs				S.l./Lect. Dr. Vlad-Alexandru Grosu			
2.3 Titularul activităţilor de seminar / laborator				S.l./Lect. Dr. Vlad-Alexandru Grosu / As. drd. Grigore Ţiplea			
2.4 Anul de studiu	1	2.5 Semestrul	1	2.6. Tipul de evaluare	E	2.7 Regimul disciplinei	Ob
2.8 Tipul disciplinei		F	2.9 Codul disciplinei	04.F.01.O.004		2.10 Tipul de notare	Nota

3. Timpul total estimat (ore pe semestru al activităţilor didactice)

3.1 Număr de ore pe săptămână	4	Din care: 3.2 curs	2	3.3 seminar/laborator	2
3.4 Total ore din planul de învăţământ	56	Din care: 3.5 curs	28	3.6 seminar/laborator	28
Distribuţia fondului de timp:					ore
Studiul după manual, suport de curs, bibliografie şi notiţe Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate Pregătire seminarii/ laboratoare/proiecte, teme, referate, portofolii şi eseuri					38
Tutorat					2
Examinări					4
Alte activităţi (dacă există):					0

3.7 Total ore studiu individual	44.00
3.8 Total ore pe semestru	100
3.9 Numărul de credite	4

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	• Algebră şi Analiză Matematică • Logică matematică
4.2 de rezultate ale învăţării	Aplicarea cunoştinţelor, conceptelor şi metodelor de bază din algebră şi analiză matematică în modelarea pe calculator a problemelor propuse spre rezolvare.

5. Condiţii necesare pentru desfăşurarea optimă a activităţilor didactice (acolo unde este cazul)

5.1 Curs	Se va desfăşura într-o sală dotată cu videoproiector şi computer, împreună cu toate accesoriile aferente: cabluri de alimentare (date şi semnal video), telecomandă videoproiector.
5.2 Seminar/ Laborator/Proiect	• Se va desfăşura într-o sală cu dotare specifică, care trebuie să includă: • Sisteme de calcul de uz general (calculatoare personale), necesare implementării metodelor şi algoritmilor specifici, pe care este instalat suportul software necesar (sistem de operare şi aplicaţii de lucru specific necesar – mediu de lucru integrat de tip IDE) • Tablă albă alături de dotările specifice: marker-e speciale, burete pentru tablă albă, soluţii de curăţare a tablei.

6. Obiectiv general *(Se referă la intenţiile profesorilor pentru studenţi, la ceea ce studenţii vor fi învăţaţi în timpul cursului. Oferă o orientare cu privire la locul cursului în cadrul domeniului ştiinţific abordat, precum şi la rolul pe care acesta îl are în cadrul specializării studiate. Vor fi descrise de o manieră generală tematicile abordate, justificarea includerii cursului în planul de învăţământ al specializării studiate etc.)*

Disciplina este prevăzută în anul 1, semestrul 1 (trunchiul comun al pregătirii inginereşti).

Reprezintă o disciplină inginerească de tip fundamental, care oferă bazele de studiu în special pentru specializările ulterioare bazate pe lucrul intensiv cu calculatorul personal, care oferă în planul de învăţământ discipline fundamental bazate pe utilizarea/proiectarea algoritmilor, scrierea de cod sursă (implementare) precum şi activităţi de depanare (identificarea erorilor semantice).

Pentru curs se urmăreşte însuşirea conceptelor programării structurate (în perspectiva fundamentării din punct de vedere sintactic a celei orientate pe obiecte în C++, care va urma) necesare modelării pe calculator a diverselor probleme din viaţa reală, dar şi cele implicate de proiectarea şi implementarea unor aplicaţii dedicate sau a temelor întâlnite în practică. Sunt analizate situaţii practice variate şi sunt implementaţi algoritmi într-un limbaj de nivel înalt proiectat folosind principiile obiectuale de proiectare software (ISO C). Programele realizate la laborator ajută la formarea reflexelor mentale în modelarea realităţii şi ajută studenţii în activitatea viitoare de inginer. Abilităţile şi deprinderile oferite de parcurgerea disciplinei constituie cunoştinţe fundamentale absolut necesare disciplinelor specifice direcţiilor de specializare orientate către discipline software, pentru care studenţii optează la trecerea în ciclul de studii următor (începând cu anul III de studiu).

Pentru laborator (aplicaţii) se propune dezvoltarea unor rutine generale de nivel cel mult mediu, dedicate rezolvării unor situaţii întâlnite în viaţa reală. Astfel, se are în vedere construirea facilă de către proiectanţii hardware/software a unei biblioteci specifice. Utilizarea în conceperea programelor a limbajului ISO C/C++, asigurând astfel portabilitatea programelor şi permiţând interfaţarea şi lucrul imediat cu majoritatea pachetelor de dezvoltare/proiectare, care se bazează pe aceste limbaje înrudite.

Abilităţile deprinse în urma scrierii programelor realizate la laborator (şi a mediului de dezvoltare integrat) devin instrumente de lucru pentru studenţi, mai întâi pentru activităţile din cadrul anilor de specializare (în funcţie de opţiunile exprimate) iar apoi, în perspectivă, pentru activităţile desfăşurate la proiectele de an şi licenţă (diplomă) şi în cele din urmă în mediul privat (viitor inginer software).

7. Competenţe *(Capacitatea dovedită de a utiliza cunoştinţe, aptitudini şi abilităţi personale, sociale şi/sau metodologice în situaţii de muncă sau de studiu şi pentru dezvoltarea profesională şi personală. Reflectă cerinţele angajatorilor.)*

Specifice	C3: Aplicarea cunoştinţelor, conceptelor şi metodelor de bază privitoare la arhitectura sistemelor de calcul, microprocesoare, microcontrolere, limbaje şi tehnici de programare.
Transversale (generale)	• CT1: Analiza metodică a problemelor întâlnite în activitate, identificând elementele pentru care există soluţii consacrate, asigurând astfel îndeplinirea sarcinilor profesionale; • CT3: Adaptarea la noile tehnologii, dezvoltarea profesională şi personală, prin formare continuă folosind surse de documentare tipărite, software specializat şi resurse electronice în limba română şi, cel puţin, într-o limbă de circulaţie internaţională.

8. Rezultatele învăţării *(Sunt enunţuri sintetice referitoare la ceea ce un student va fi capabil să facă sau să demonstreze la finalizarea unui curs. Rezultatele învăţării reflectă realizările studentului şi mai puţin intenţiile profesorului. Rezultatele învăţării informează studenţii despre ceea ce se aşteaptă de la ei din punct de vedere al performanţei, pentru a obţine notele şi creditele dorite. Sunt definite în termeni concreţi, folosind verbe similare exemplelor de mai jos şi indică ceea ce se va urmări prin evaluare. Rezultatele învăţării vor fi astfel redactate încât să fie evidenţiată clar relaţia faţă de competenţele definite la punctul 7.)*

Cunoştinţe	*Rezultatul asimilării de informaţii prin învăţare. Cunoştinţele reprezintă ansamblul de fapte, principii, teorii şi practici legate de un anumit domeniu de muncă sau de studiu. Pot fi teoretice şi/sau faptice.* • **Enumeră** conceptele programării în general şi structurate în special (cf. teoremei Bohm-Jacopini), care stau la baza limbajului studiat. • **Înţelege** faptul că implementarea nu este singura activitate de bază în contextul programării calculatoarelor şi mai general al dezvoltării software. • **Analizează** cerinţele de proiectare şi paşii necesari proiectării în sine a unei aplicaţii (consecinţă a enunţului anterior). • **Asimilează** vocabularul de specialitate (jargonul) folosit în domeniul programării. • Este capabil să **descrie** zonele componente ale programului ISO C implementat, necesar rezolvării unei probleme date (prezentată în limbaj natural).

Aptitudini	*Capacitatea de a aplica cunoştinţe şi de a utiliza know-how pentru a duce la îndeplinire sarcini şi a rezolva probleme. Aptitudinile sunt descrise ca fiind cognitive (implicând utilizarea gândirii logice, intuitive şi creative) sau practice (implicând dexteritate manuală şi utilizarea de metode, materiale, unelte şi instrumente).* • **Identifică** şi descrie conceptele fundamentale împreună cu modalităţile de implementare ale acestora: ascunderea datelor/funcţiilor, reutilizarea codului, supraîncărcarea şi suprascrierea funcţiilor. • **Modelează** problemele propuse, formalizând textul problemei (exersează etapa de proiectare a programului/aplicaţiei). • **Identifică** componentele programului care interacţionează în aplicaţia ce urmează să fie scrisă. • **Identifică soluţii** şi **elaborează planuri** de rezolvare pentru problemele propuse (exprimate în limbaj natural). • **Analizează** şi compară soluţia găsită problemei propuse spre rezolvat cu sugestiile oferite de către titularul de laborator pentru problema specifică de rezolvat. • **Testează, depanează şi rulează** programul scris (aplicaţia implementată) şi este capabil să îndrepte posibilele erori, identificându-le în prealabil, alături de posibilele consecinţe.
Responsabilitate şi autonomie	*Capacitatea cursantului de a aplica în mod autonom şi responsabil cunoştinţele şi aptitudinile sale.* • **Demonstrează receptivitate** pentru contexte noi de învăţare (principiile programării structurate pot constitui o noutate, din perspectiva planului de învăţământ). • **Respectă** principiile de etică academică, înţelegând responsabilităţile asumate de rezolvare individuală a problemelor propuse, folosind metodele şi tehnicile învăţate. • **Demonstrează autonomie** în organizarea situaţiei/contextului de învăţare sau a situaţiei expuse de către problema de rezolvat. • **Conştientizează** valoarea contribuţiei sale în domeniul ingineriei software, din perspectiva vieţii active, odată cu identificarea şi propunerea unor soluţii proprii, care să rezolve probleme din viaţa socială şi economică (responsabilitate socială).

9. Metode de predare *(Se vor avea în vedere metode care să asigure predarea centrată pe student. Se va descrie modul în care se asigură participarea studenţilor la stabilirea propriului parcurs de învăţare, cum se identifică eventualele rămâneri în urmă şi ce măsuri remediale se adoptă în astfel de cazuri.)*

Plecând de la analiza caracteristicilor de învăţare ale studenţilor şi de la nevoile lor specifice, identificate de către titularul de curs în urma experienţei personale în mediul privat, procesul de predare foloseşte metode de predare atât expozitive (prelegerea, expunerea), cât şi conversative-interactive, bazate pe modele de învăţare bazate pe acţiune, precum exerciţiul sau rezolvarea de probleme de programare. Interactivitatea cu studenţii prin intermediul părţii aplicative asociate conceptelor predate este în mod special evidenţiată.

Sunt rezervate intervale de prezentare şi rezolvare a unor probleme curente pe care studenţii le întâlnesc la discipline din anul 2 din domeniul electronic: modelarea dispozitivelor electronice, analiza componentelor şi circuitelor pasive precum şi pregătirea abilităţilor inginereşti de rezolvare a problemelor cu ajutorul calculatorului (metode numerice).

Partea de modelare se reduce adesea la rezolvarea unor probleme practice, identificate în viaţa reală. Conceptele prezentate sunt cele pregătitoare disciplinelor corelate următoare din anii 1 şi 2 (Programarea Calculatoarelor şi Limbaje de Programare 2, Structuri de Date şi Algoritmi, Metode Numerice) şi mai târziu în anii de specializare (anii 3 şi 4), în cadrul unor discipline precum: Deep Learning şi Inteligenţă Artificială, Robotică sau Reţele Neuronale şi Sisteme Fuzzy.

Dialogul din timpul cursului continuă în cadrul şedinţelor de laborator. Acestea sunt necesare pregătirii studenţilor pentru testele de verificare individuale de pe parcurs, care formează abilităţile de rezolvare a problemelor contra-timp. O şedinţă tipică începe cu o scurtă trecere în revistă a noţiunilor de programare specifice lucrării. Ulterior, studenţii urmăresc să conceapă şi să scrie programe complete, funcţionale - plecând de la enunţul în limbaj natural (limba română).

Limbajul folosit este ISO C, în varianta standard C 1999. Mediul de dezvoltare open-source folosit în laborator este configurabil în acest sens şi poate fi instruit să folosească varianta standard indicată a limbajului ISO C prin folosirea opţiunii de compilare -std=c99. Mediul integrat de programare şi materialul aferent platformelor pentru laborator sunt disponibile studenţilor sub formă electronică.

10. Conţinuturi

CURS		
Capitolul	**Conţinutul**	**Nr. ore**
1	Introducere. Noţiunea de compilator. Standardele C. Comentariu bibliografic.	2
2	Structura programelor C. Distincţia sintaxă-semantică. Componentele unui program. Alfabetul limbajului. Cuvintele-cheie.	2
3	Memoria: zone şi spaţii de memorie. Alinierile în memorie. Baze de numeraţie. Tipuri de memorie. Modele de memorie folosite de către compilator. Tipurile de date fundamentale din C. Declararea şi definirea variabilelor şi constantelor. Clase de memorare. Variabile globale.	2
4	Operatori şi expresii. Clasificarea operatorilor. Exemple pentru clasele de operatori identificate. Ordine de precedenţă şi asociativitate. Conversiile implicite şi explicite. Detalii asupra zonei de comentarii din structura generală a programului. Detalii asupra fişierelor antet. Detaliu asupra zonei directivelor preprocesor. Funcţii inline (macro): introducere.	2
5	Instrucţiuni. Instrucţiunile expresie, bloc, decizie (if), selecţie (switch), ciclare. Exemple. Instrucţiunile de salt: continue, break, goto.	2
6	Prototipuri şi definiţii de funcţii. Funcţii inline (continuare): diferenţele între macro-expandare şi apelul de funcţie. Funcţia main(): detalii. Modularizare. Detaliu asupra zonei de definire a tipurilor de date de către utilizator (typedef). Detaliu asupra zonei prototipurilor de funcţii.	2
7	Funcţii. Declaraţie şi definiţie. Variabile locale şi externe funcţiilor. Apel de funcţii. Parametri formali şi efectivi. Stiva. Convenţii de apel. Funcţii macro (detalii). Comparaţia funcţiilor iterative şi recursive.	2

8	Pointeri. Declarare, iniţializare. Operatorii de preluare a adresei şi de-referenţiere. Valoarea specială NULL. Aritmetica pointerilor. Tipul void aplicat la pointeri. Funcţii cu argument pointer şi tip returnat pointer. Pointeri către funcţii. Tehnica alocării dinamice de memorie.	4
9	Vectori. Vectori 1-dimensionali şi n-dimensionali: declarare, definire. Iniţializare. Indexare. Şiruri. Legătura vector-pointer. Funcţii cu argument vector. Enumerări. declarare/definire. Enumerările şi pointerii. Funcţii cu argument enumerări.	3
10	Uniuni: declarare/definire, iniţializare. Tipuri de date neomogene. Structuri: declarare, definire, iniţializare. Operaţii permise asupra structurilor. Structurile şi pointerii. Operatorul sizeof aplicat structurilor.	3
11	Limbajul C în operaţii de intrare/ieşire (I/O). Noţiunile de flux şi fişier. Fişiere. Operaţii asupra fişierelor: deschidere, închidere, scriere, citire, interogare/stabilirea valorii indicatorului de poziţie. Ştergerea fizică a fişierelor. Redenumire. Funcţii de tratare a erorilor.	2
12	Aspecte avansate ale limbajului. Conceptul de variabilă. Declaraţii/definiţii (detalii de implementare). Domeniul de vizibilitate şi durata de viaţă a variabilelor. Legarea variabilelor. Clase de memorare şi spaţii ale numelor. Valoare dreapta-valoare stânga. Etapele compilării. Funcţii cu număr variabil de argumente. Funcţii recursive (detaliu). Crearea de tipuri de date proprii: tipuri de bibliotecă.	2
	Total:	28

Bibliografie:

1. Şl. Dr. ing. GROSU Vlad-Alexandru, PCLP1, suport de curs electronic (Moodle): (https://archive.curs.upb.ro/2021/course/view.php?id=8868, https://curs.upb.ro/2024/course/view.php?id=3871)
2. I. Rusu, Dana Gavrilescu, Vlad Al. Grosu - *Programarea calculatoarelor în limbaj C*, Editura MatrixRom, Bucureşti, 2002.
3. I. Rusu, Vlad Al. Grosu – *Programarea calculatoarelor în limbaj C: probleme rezolvate şi comentate*, Editura MatrixRom, Bucureşti, 2008.
4. D.I. Năstac, *Programarea calculatoarelor în limbajul C – Elemente fundamentale*, Editura Printech, Bucureşti, 2006.
5. D. Burileanu, C. Dan, M. Pădure, *Programare în C. Culegere de probleme*, Editura Printech, Bucureşti, 2004.
6. Brian Kernighan, Dennis Richie – *The C programming language*, Prentice Hall, New Jersey, ediţiile 1978 şi 1988.
7. Emanuela Cerchez, Marinel Şerban – *Programarea în limbajul C/C++ pentru liceu*, vol. 1, Ed. Polirom, ≥ 2013.

LABORATOR		
Nr. crt.	**Conţinutul**	**Nr. ore**

1	Baze de numeraţie. Conversii între bazele de numeraţie uzuale.	2
2	Alfabetul limbajului. Cuvintele-cheie. Declararea şi definirea variabilelor.	2
3	Corp de instrucţiuni. Operatori (introducere).Construcţia expresiilor. Instrucţiuni	2
4	Operatori (continuare): operatorul de atribuire. Conversia explicită (cast). Operatorul de secvenţiere. Definiţia tipurilor utilizator: typedef. Instrucţiunea switch-case. Întreruperi şi salturi în C. Ciclul for.	2
5	Instrucţiunile de ciclare while şi do-while. Vectori (introducere).	2
6	Vectori (continuare): şiruri. Alocarea statică. Pointeri (I) - Introducere.	2
7	Pointeri (II) - Pointer NULL. Aritmetica pointerilor. Legătura vector-pointer.	4
8	Pointeri (III): pointerii şi alocarea dinamică.	4
9	Structuri. Operatorul typedef aplicat structurilor. Pointerii şi structurile: construcţia unui tip abstract de dată.	2
10	Enumerări. Uniuni. Deosebiri între uniuni şi structuri.	2
11	Operaţii I/O. Funcţiile de bibliotecă utilizate în lucrul cu fişierele.	2
12	Verificare finală laborator.	2
	Total:	28

Bibliografie:

1. Şl. Dr. ing. GROSU Vlad-Alexandru, PCLP1, suport de curs electronic (Moodle) (https://archive.curs.upb.ro/2021/course/view.php?id=8868)
2. I. Rusu, Dana Gavrilescu, Vlad Al. Grosu - *Programarea calculatoarelor în limbaj C*, Editura MatrixRom, Bucureşti, 2002.
3. I. Rusu, Dana Gavrilescu, Vlad Al. Grosu – *Îndrumar de laborator pentru programarea calculatoarelor: C*, Editura MatrixRom, Bucureşti, 2004.
4. I. Rusu, Vlad Al. Grosu – *Programarea calculatoarelor în limbaj C: probleme rezolvate şi comentate*, Editura MatrixRom, Bucureşti, 2008.
5. Brian Kernighan, Dennis Richie – *The C programming language*, Prentice Hall, New Jersey, ediţiile 1978 şi 1988.
6. Herbert Schildt – *C – manual complet*, Editura Teora, 1999-2003.
7. Florin Munteanu, Gh. Muscă, Florin Moraru – *C - tehnici de programare*, Editura Joint Printing House, Bucureşti, 1995.

11. Evaluare

Tip activitate	11.1 Criterii de evaluare	11.2 Metode de evaluare	11.3 Pondere din nota finală

11.4 Curs	- Identificarea corectă a contextelor teoretice şi practice de aplicare a algoritmilor şi tehnicilor de programare predate. - Definirea conceptelor, principiilor şi metodelor folosite în domeniile: programarea calculatoarelor, limbaje de nivel înalt și specifice, arhitectura sistemelor de calcul, sisteme electronice programabile, grafică, arhitecturi software reconfigurabile.	- Susținerea probei individuale scrise de examinare (total: 40p). - Se investighează atât capacitatea studenților de a formaliza o problemă cât și pe cea de transpunere în program a conceptelor specifice. - Subiectele acoperă atât partea teoretică aferentă definirii conceptelor programării obiectuale, cât şi pe cea practică, din perspectiva capacității de rezolvare cu ajutorul limbajului ISO C a unor probleme de programare (prezentate în limba română).	40%
11.5 Seminar/laborator/proiect	- C4.5: Susţinerea şi promovarea unei probe referitoare la arhitectura şi principiile funcţionale ale unei structuri software funcţionale. - C6.5: Susţinerea unei probe privind stabilirea şi descrierea operaţiilor necesare pentru realizarea şi testarea unui algoritm numeric.	Activitatea de laborator este verificată constant, pe tot parcursul semestrului, prin probe individuale la calculator. Studenţii pot acumula: • 10%, în urma testelor de activitate la fiecare laborator; • 20%, testul de verificare de la jumătatea semestrului; • 30%, colocviu – laboratorul se încheie cu o verificare finală, individuală, la staţie. Aceasta se bazează pe implementarea unui algoritm din materia parcursă pe parcursul întregului semestru și pe rezolvarea unui subiect teoretic cu rol sintetic.	60%
11.6 Condiţii de promovare			
• Se verifică abilitățile de acumulare și apoi de identificare a situațiilor practice specifice metodelor prezentate precum și a aplicării corecte a acestor metoda în domeniul ingineriei informaționale. • Promovarea materiei presupunea acumularea (fără praguri intermediare impuse) a ***cel puțin 50p*** din totalul de 100p aferent.			

12. Coroborarea conţinutului disciplinei cu aşteptările reprezentanţilor angajatorilor şi asociaţiilor profesionale reprezentative din domeniul aferent programului, precum şi cu stadiul actual al cunoaşterii în domeniul ştiinţific abordat şi practicile în instituţii de învăţământ superior din Spaţiul European al Învăţământului Superior (SEÎS)

Universitatea Naţională de Ştiinţă şi Tehnologie Politehnica Bucureşti

Facultatea de Electronică, Telecomunicaţii şi

Tehnologia Informaţiei

În prezent se impune pregătirea viitorilor ingineri şi cercetători în domeniul algoritmilor numerici de rezolvare prin diferite metode matematice a problemelor de proiectare, testare sau prelucrare a diferitelor semnale. Formarea studenţilor pe trunchiul de programare asigură cunoştinţele fundamentale necesare în orice activitate profesională ulterioară: învăţământ, cercetare şi/sau proiectare.

Activităţile aferente Programării Calculatoarelor oferă bazele gândirii algoritmice şi ale programării într-un limbaj de programare actual, aşa cum este limbajul ANSI/ISO C conform ultimei sale standardizări din anul 2011 (ISO/IEC 9899:2011).

Prin structurarea informaţiei conform activităţilor prevăzute în curriculum, ca şi prin activitatea de tutorat desfăşurată, materia oferă paşii necesari aprecierii calităţii, meritelor şi limitelor unor procese, programe, proiecte, concepte, metode şi teorii.

Utilizarea adecvată a criteriilor şi metodelor de evaluare, conforme normelor academice europene la care universitatea POLITEHNICA Bucureşti este parte, permite studenţilor să se autoevalueze continuu, pornind de la calificativele obţinute dar şi ţinând cont de observaţiile şi indicaţiile metodologice pe care titularul de curs/laborator le oferă.

Data completării	Titular de curs	Titular(i) de aplicaţii
21.09.2026	S.l./Lect. Dr. Vlad-Alexandru Grosu 	S.l./Lect. Dr. Vlad-Alexandru Grosu 
		As. drd. Grigore Ţiplea 

Data avizării în departament	Director de departament
22.09.2026	Prof. Dr. Claudius Dan 

Data aprobării în Consiliul Facultăţii	Decan
22.09.2026	Prof. Dr. Mihnea Udrea

Universitatea Națională de Știință și Tehnologie Politehnica București

Facultatea de Electronică, Telecomunicații și

Tehnologia Informației

