

UNIVERSITATEA POLITEHNICA DIN BUCUREŞTI  
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE  
DEPARTAMENTUL DE CALCULATOARE



# PROIECT DE DIPLOMĂ

Implementarea algoritmului Ray Tracing  
folosind arhitectura DirectX Raytracing

Alex-Andrei Cioc

**Coordonator științific:**

Conf. Dr. Ing. Victor Asavei

**BUCUREŞTI**  
2024

UNIVERSITY POLITEHNICA OF BUCHAREST  
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS  
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT



## DIPLOMA PROJECT

Implementation of the Ray Tracing algorithm  
using the DirectX Raytracing architecture

Alex-Andrei Cioc

**Thesis advisor:**

Conf. Dr. Ing. Victor Asavei

**BUCHAREST**

2024

# **CUPRINS**

<b>1</b>	<b>Introducere</b>	<b>1</b>
1.1	Context . . . . .	1
1.2	Problema . . . . .	1
1.3	Obiective . . . . .	3
1.4	Soluția propusă . . . . .	4
1.5	Rezultatele obținute . . . . .	4
1.6	Structura lucrării . . . . .	4
<b>2</b>	<b>Motivație de cercetare</b>	<b>5</b>
<b>3</b>	<b>Metode Existente</b>	<b>6</b>
<b>4</b>	<b>Soluția Propusă</b>	<b>7</b>
<b>5</b>	<b>Detalii de implementare</b>	<b>8</b>
5.1	Indicații formatare tabele . . . . .	8
<b>6</b>	<b>Evaluare</b>	<b>10</b>
<b>7</b>	<b>Concluzii</b>	<b>12</b>
<b>Anexe</b>		<b>14</b>
<b>Anexa A</b>	<b>Extrase de cod</b>	<b>15</b>

## SINOPSIS

Algoritmul Ray Tracing este o tehnică de randare a imaginilor care simulează propagarea și comportamentul razelor de lumină într-o scenă tridimensională. Acesta este adesea folosit în industria cinematografică pentru a obține imagini fotorealiste. Până de curând, natura computațională intensivă a acestui algoritm a limitat utilizarea sa în aplicații interactive, precum jocurile video. Totuși, cu avansul tehnologiei, utilizarea acestuia a devenit tot mai accesibilă și pentru aceste aplicații. Suport hardware pentru Ray Tracing în contextul consumatorilor a fost introdus de NVIDIA în 2018, prin intermediul arhitecturii Turing<sup>1</sup>. Tot în același an, Microsoft a anunțat DirectX Raytracing<sup>2</sup> (DXR), o extensie a API-ului DirectX 12 care permite programatorilor să folosească Ray Tracing în aplicațiile lor, utilizând hardware-ul compatibil.

Lucrarea de față își propune să studieze și să implementeze algoritmul Ray Tracing pe un sistem de calcul modern, folosind accelerarea hardware oferită de arhitectura DirectX Raytracing. Acest algoritm va fi folosit pentru randarea iluminării unor scene arbitrare, în timp real, oferind o reprezentare fotorealistă a acestora. În cadrul lucrării se va realiza o analiză a performanțelor implementării curente, atât a fidelității imaginilor generate, cât și a eficienței spațio-temporale a implementării. Se vor explora și posibilitățile de optimizare a algoritmului, precum și modul în care acestea pot fi folosite pentru a îmbunătăți performanțele sistemului.

## ABSTRACT

The Ray Tracing algorithm is an image rendering technique that simulates the propagation and behavior of light rays in a three-dimensional scene. It is often used in the film industry to achieve photorealistic images. Until recently, the computationally intensive nature of this algorithm has limited its use in interactive applications, such as video games. However, with technological advances, its use has become increasingly accessible for these applications as well. Hardware support for Ray Tracing in the consumer context was introduced by NVIDIA in 2018, through the Turing<sup>1</sup> architecture. In the same year, Microsoft announced DirectX Raytracing<sup>2</sup> (DXR), an extension of the DirectX 12 API that allows programmers to use Ray Tracing in their applications, using compatible hardware.

This paper aims to study and implement the Ray Tracing algorithm on a modern computing system, using the hardware acceleration provided by the DirectX Raytracing architecture. This algorithm will be used for rendering the lighting of arbitrary scenes, in real-time, providing a photorealistic representation of them. The paper will perform an analysis of the current implementation's performance, both in terms of the fidelity of the generated images and the spatio-temporal efficiency of the implementation. It will also explore the possibilities for optimizing the algorithm, as well as how these can be used to improve the system's performance.

---

<sup>1</sup><https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf>. Accesat 06.10.2024.

<sup>2</sup><https://devblogs.microsoft.com/directx/announcing-microsoft-directx-raytracing/>. Accesat 06.10.2024.

## **MULTUMIRI**

Adresez mulțumiri coordonatorului meu de proiect, Conf. Dr. Ing. Victor Asavei, pentru îndrumarea și sprijinul acordat pe parcursul realizării acestei lucrări, dar și pentru inspirația și motivația oferită în cadrul cursurilor de Elemente de Grafică pe Calculator. De asemenea, mulțumesc familiei și prietenilor pentru susținere și încurajare.

# 1 INTRODUCERE

## 1.1 Context

Industria jocurilor video este una dintre cele mai mari și mai profitabile industrii de divertisment din lume. În 2021, piața jocurilor video era evaluată la aproximativ 202.64 miliarde de dolari și este estimat să se extindă la o rată anuală compusă de creștere de 10.2% în perioada 2022-2030<sup>1</sup>. Această industrie este alimentată de cererea pentru experiențe interactive și captivante, care să ofere o experiență de joc cât mai realistă și cât mai imersivă. Toate studio-urile de dezvoltare de jocuri video AAA (i.e., jocuri cu bugete mari și echipe de dezvoltare extinse) investesc resurse semnificative în dezvoltarea de tehnologii care să le permită să creeze jocuri cu grafică de înaltă calitate. Aceste tehnologii includ motoare grafice puternice, care să permită randarea unor scene complexe, cu iluminare realistă și efecte speciale impresionante. Multe studio-uri folosesc propriile motoare dezvoltate in-house (e.g., Frostbite de la EA, CryEngine de la Crytek, Anvil de la Ubisoft), dar există și motoare comerciale, precum Unreal Engine și Unity. Aceste motoare oferă un set de instrumente și funcționalități care permit dezvoltatorilor să creeze jocuri video de înaltă calitate, fără a fi nevoie să dezvolte de la zero toate componentele necesare. O componentă critică a acestor motoare este motorul grafic, care se ocupă de randarea scenei jocului, de la geometria obiectelor până la iluminare și efecte speciale. Astfel, programatorii, artiștii, animatorii și designerii de jocuri pot să se concentreze pe crearea conținutului jocului, fără a fi nevoie să se ocupe de detalii tehnice ale randării grafice.

## 1.2 Problema

Tehnica tradițională și cea mai răspândită de randare a imaginilor în jocurile video este rasterizarea. Această tehnică se bazează pe proiecția obiectelor 3D pe un plan bidimensional, folosind o serie de algoritmi și tehnici pentru a simula iluminarea și efectele speciale. Rasterizarea este o tehnică eficientă și rapidă, care permite randarea unui număr mare de obiecte în timp real, dar are și limitări. Una dintre cele mai mari limitări ale rasterizării este incapacitatea de a simula iluminarea globală, care este esențială pentru obținerea unor imagini fotorealiste. De asemenea, reflexiile și refractiile pot fi doar approximate, de exemplu prin tehnici de cubemapping sau screen-space reflections. Aceste tehnici sunt eficiente, dar nu oferă rezultate realistice, iar în multe cazuri pot fi observate artefacte vizuale care afectează calitatea imaginii.

În continuare se evidențiază aceste limitări (care nu sunt deloc exhaustive) ale rasterizării, prin

---

<sup>1</sup><https://www.grandviewresearch.com/industry-analysis/gaming-industry>. Accesat 06.10.2024.

comparație cu tehnica de Ray Tracing (așa numita *RTX* în jocurile sponsorizate de Nvidia). Comparând Figurile 1 și 2, se observă neajunsul reflexiilor în screen space. Atâtă timp cât obiectele reflectate se află în viewport, reflexiile sunt corecte și realiste. Însă, dacă obiectele ies din viewport, reflexiile se pierd, ceea ce duce la o imagine nerealistă. Un alt exemplu și mai elovent este ilustrat în Figura 4, unde imaginea randată cu Ray Tracing redă reflexii ale exploziei care nu este vizibilă decât parțial în cadru.



Figura 1: Screen space reflections în Minecraft<sup>2</sup>



Figura 2: Artefacte vizuale în screen space reflections<sup>2</sup>. Se poate observa cum reflexiile se pierd dacă obiectele reflectate ies din viewport

Am văzut cum tehnica de Ray Tracing poate oferi rezultate mult mai realiste decât rasterizarea, dar această tehnologie vine cu un cost. Algoritmul Ray Tracing este computațional intensiv,

<sup>2</sup>Shader folosit: <https://continuum.graphics/>. Accesat 06.10.2024.

<sup>3</sup>©Nvidia Corporation: <https://blogs.nvidia.com/blog/geforce-rtx-real-time-ray-tracing/>. Accesat 06.10.2024.

<sup>4</sup>©Nvidia Corporation: <https://www.youtube.com/watch?v=WoQr0k2IA9A>. Accesat 06.10.2024.

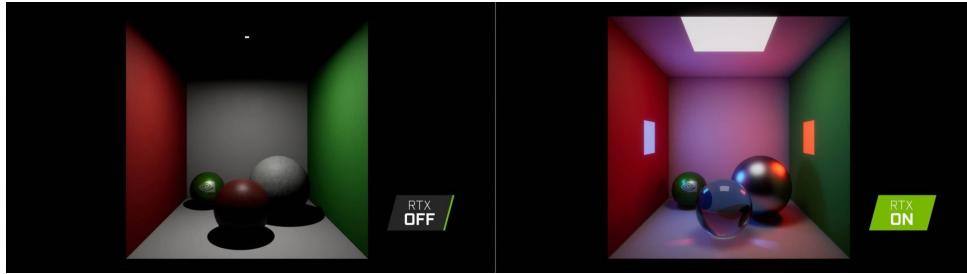


Figura 3: Iluminarea globală este absentă în imaginea randată cu rasterizare<sup>3</sup>

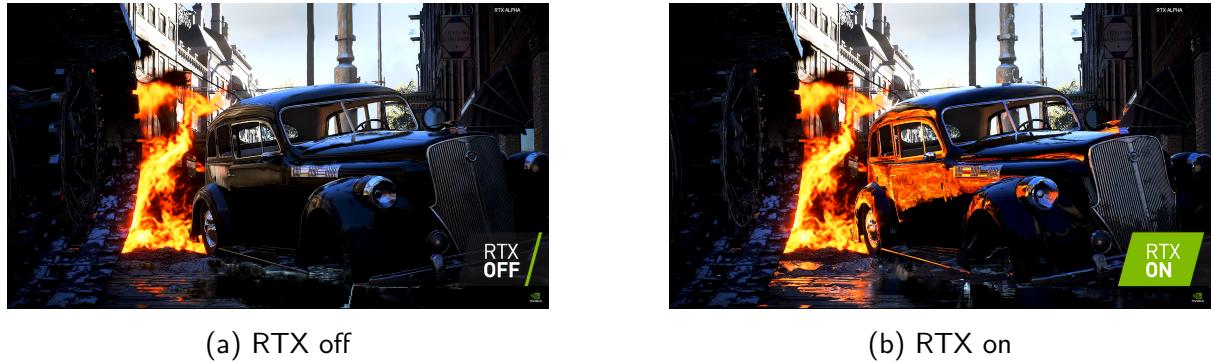


Figura 4: Comparație RTX on/off în Battlefield V<sup>4</sup>

deoarece necesită calcularea intersecțiilor mai multor raze de lumină pentru fiecare pixel cu obiectele din scenă și calcularea contribuției acestora la culoarea pixelului. Pentru a obține o imagine de calitate, este nevoie de un număr mare de raze de lumină și tehnici de denoising, ceea ce face ca algoritmul să fie greu de balansat între fidelitate și performanță.

### 1.3 Obiective

Scopul acestei lucrări este de a cerceta și implementa algoritmul Ray Tracing în context de timp real și a evalua performanțele acestuia, prin comparație cu implementări destinate producției cinematografice. Eforturile vor fi concentrate pe implementarea unui motor grafic simplu, cu câteva funcționalități de bază:

- Controle de cameră simple (mișcare, zoom, rotire)
- Randarea obiectelor definite ca mesh-uri triunghiulare
- Randarea obiectelor definite prin funcții implicate (e.g., sfere, planuri)
- Suport pentru materiale PBR (Physically Based Rendering)
- Suport pentru încărcare de scene predefinite,

precum și a unor efecte de iluminare care să ofere o imagine fotorealistică a scenei:

- Iluminare globală

- Reflexii și refracții
- Umbre
- Denoising.

Dorim să obținem o implementare eficientă, care să țintească un frametime de 16.6ms (60 de cadre pe secundă) pentru hardware entry-level cu suport pentru DirectX Raytracing (e.g., Nvidia GeForce RTX 2060). Pentru a atinge acest obiectiv, vom explora și implementa tehnici de optimizare a algoritmului propriu-zis și vom urmări utilizarea API-ului DirectX 12 (cu extensia DXR) într-un mod eficient.

## 1.4 Soluția propusă

Soluția propusă este un motor grafic simplu de utilizat. Din perspectiva utilizatorului, acesta are un meniu din care poate configura diverse parametru ai algoritmului de Ray Tracing, precum și a scenei. Utilizatorul poate încărca scene predefinite și poate interacționa cu acestea folosind controalele de cameră.

//TODO adaugat flowchart, descriere algoritm

Din perspectiva implementării, blabla

## 1.5 Rezultatele obținute

//TODO

Descriere pe scurt a rezultatelor obținute, eventual de ce acestea sunt importante față de alte soluții sau studii.

## 1.6 Structura lucrării

//TODO

Un paragraf în care fiecare dintre secțiunile următoare este prezentată în 1-2 fraze, punând accentul pe elementele cele mai semnificative din fiecare secțiune.

## **2 MOTIVATIE DE CERCETARE**

//TODO

Proiectul de față cercetează tehnici de randare a imaginilor în timp real. Acest domeniu este de mare interes pentru industria jocurilor video, care investesc resurse semnificative în dezvoltarea de motoare grafice puternice.

### 3 METODE EXISTENTE

//TODO read papers motherfucker and benchmark your shit

[*Cercetare*] Metode existente (sau “State of the Art”) se referă, de regulă, la nivelul curent de dezvoltare: care este starea curentă a domeniului, unde ne găsim, care este contextul. Care sunt soluțiile actuale prezente în literatura de specialitate și care sunt limitările lor? Ce direcții de explorare sunt recomandate în literatura de specialitate? Literatura de specialitate se referă la articole științifice recente, publicate în reviste cu factor de impact mare, sau în volumele unor conferințe de top, sau în cărți.

[*Ambele*] În încheierea acestui capitol se dorește descrierea tehnologiilor folosite în lucrare, cu alternative și cu argumente convingătoare calitative și cantitative.

Criterii pentru calificativul *NeSatisfăcător*:

- [*Dezvoltare de produs*] Sunt analizate superficial câteva produse de pe piață;
- [*Cercetare*] analiza literaturii limitată la grupuri de cercetare din România;
- [*Ambele*] Sunt descrise tehnologiile folosite în lucrare.

Criterii pentru calificativul *Satisfăcător*:

- [*Dezvoltare de produs*] Există un interviu, un client, analiza cerințelor este elaborată pe baza interviului.
- [*Cercetare*] analiza literaturii de specialitate din lume, fără poziționarea precisă a lucrării în peisajul domeniului studiat;
- [*Ambele*] Sunt descrise câteva tehnologii alternative pentru fiecare din tehnologiile folosite în lucrare. Există o argumentare referitoare la alegere.

Criterii pentru calificativul *Bine*:

- [*Dezvoltare de produs*] Proces iterativ pe baza unor interviuri cu mai mulți clienți, dezvoltare MVP, reevaluare cerințe;
- [*Cercetare*] analiza literaturii de specialitate din lume, cu poziționarea precisă a lucrării în peisajul actual al domeniului studiat;
- [*Ambele*] Sunt descrise tehnologii alternative. Sunt analizate cantitativ și calitativ, folosite benchmarkuri și teste efectuate de student. Analiza este rezumată prin tabele și grafice.

## 4 SOLUȚIA PROPUȘĂ

//TODO

Capitolul conține o privire de ansamblu a soluției ce rezolvă problema, prin prezentarea structurii / arhitecturii acesteia. În funcție de tipul lucrării acest capitol poate conține diagrame (clase, distribuție, workflow, entitate-relație), demonstrații de corectitudine pentru algoritmii propuși de autor, abordări teoretice (modelare matematică), structura hardware, arhitectura aplicației.

Criterii pentru calificativul *NeSatisfăcător*:

- Descriere în limbaj natural.

Criterii pentru calificativul *Satisfăcător*:

- Descriere + diagrame de baze de date, workflow, clase, algoritmi.

Criterii pentru calificativul *Bine*:

- Descriere + diagrame de baze de date, workflow, clase, algoritmi + descrierea unui proces prin care s-a realizat arhitectura/structura soluției.

## 5 DETALII DE IMPLEMENTARE

//TODO

În plus fata de capitolul precedent acesta conține elemente specifice ale rezolvării problemei care au presupus dificultăți deosebite din punct de vedere tehnic. Pot fi incluse configurații, secvențe de cod, pseudo-cod, implementări ale unor algoritmi, analize ale unor date, scripturi de testare. De asemenea, poate fi detaliat modul în care au fost utilizate tehnologiile introduse în capitolul 3.

Criterii pentru calificativul *NeSatisfăcător*:

- Sunt prezentate pe scurt scheme și pseudo-cod.

Criterii pentru calificativul *Satisfăcător*:

- Descriere sumară a implementării, prezentarea unor secvențe nerelevante de cod, scheme, etc.

Criterii pentru calificativul *Bine*:

- Descrierea detaliată a algoritmilor/structurilor utilizați; Prezentarea etapizată a dezvoltării, inclusiv cu dificultăți de implementare întâmpinate, soluții descoperite; (dacă este cazul) demonstrarea corectitudinii algoritmilor utilizați.

### 5.1 Indicații formatare tabele

Se recomandă utilizarea tabelelor de forma celui de mai jos. Font size : 9. Orice tabel prezent în teză va fi referit în text; exemplu: a se vedea Tabel 1.

Tabela 1: Sumarizare criterii

Calificativ	Criteriu	Observații
<b>Nesatisfacator</b>	Sunt prezentate pe scurt scheme și pseudo-cod	
<b>Satisfacator</b>	Descriere sumara a implementării, prezentarea unor secvențe nerelevante de cod, scheme, etc.	
<b>Bine</b>	Descrierea detaliată a algoritmilor/structurilor utilizati; Prezentarea etapizată a dezvoltării, inclusiv cu dificultăți de implementare întâmpinate, soluții descoperite; (dacă este cazul) demonstrarea corectitudinii algoritmilor utilizati.	Pot fi incluse configurații, secvențe de cod, pseudo-cod, implementări ale unor algoritmi, analize ale unor date, scripturi de testare.

## 6 EVALUARE

//TODO

Acest capitol trebuie să răspundă, în principiu, la 2 întrebări și să se încheie cu o discuție a rezultatelor obținute. Cele două întrebări la care trebuie să se răspundă sunt:

1. **Merge corect?** (Conform specificațiilor extrase în capitolul 2); Evaluarea dacă merge corect se face pe baza cerințelor identificate în capitolele anterioare.
2. Cât de *Bine* merge / cum se compară cu soluțiile existente? (pe baza unor metrii clare). Evaluarea cât de *Bine* merge trebuie să fie bazată pe procente, timpi, cantitate, numere, **comparativ cu soluțiile prezentate în capitolul 3**. Poate fi vorba de performanță, overhead, resurse consumate, scalabilitate etc.

În realizarea discuției, se vor utiliza tabele cu procente, rezultate numerice și grafice. În mod obișnuit, aici se fac comparații și teste comparative cu alte proiecte similare (dacă există) și se extrag puncte tari și puncte slabe. Se ține cont de avantajele menționate și se demonstrează viabilitatea abordării / aplicației, de dorit prin comparație cu alte abordări (dacă acest lucru este posibil). Cuvântul cheie la evaluare este “metrică”: trebuie să aveți noțiuni măsurabile și cuantificabile. În cadrul procesului de evaluare, explicați datele, tabelele și graficele pe care le prezentați și insistați pe relevanța lor, în următorul stil: “este de preferat ... deoarece ...”; explicați cititorului nu doar datele ci și semnificația lor și cum sunt acestea interpretate. Din această interpretare trebuie să rezulte poziționarea proiectului vostru printre alternativele existente, precum și cum poate fi acesta îmbunătățit în continuare.

Criterii pentru calificativul *NeSatisfăcător*:

- Aplicația este testată dar rulează pe calculatorul studentului, nu există posibilități de testare, nu a fost validată cu clienți / utilizatori;
- Nu au fost realizate comparații cu alte sisteme similare.

Criterii pentru calificativul *Satisfăcător*:

- [Dezvoltare de produs] Există teste unitare și de integrare, există o strategie de punere în funcțiune (deployment), există validare minimală cu clienții / utilizatorii.
- [Cercetare] Principalele componente și soluția în ansamblu au fost evaluate din punct de vedere al performanței, însă nu sunt folosite seturi de date standard, există unele erori de interpretare a datelor.
- [Ambele] Discuție minimală asupra relevanței rezultatelor prezentate, comparație minimală cu alte sisteme similare.

Criterii pentru calificativul *Bine*:

- **[Dezvoltare de produs]** Teste unitare și de integrare, instrumente de punere în funcțiune (deployment) utilizate și care arată lucru constant de-a lungul semestrului, lucrare validată cu clienții / utilizatorii, produs în producție.
- **[Cercetare]** Componentele și soluția în ansamblu au fost evaluate din punct de vedere al performanței, folosind seturi de date standard și cu o interpretare corectă a rezultatelor.
- **[Ambele]** Discuție cu prezentarea calitativă și cantitativă a rezultatelor, precum și a relevanței acestor rezultate printr-o comparație complexă cu alte sisteme similare.

## 7 CONCLUZII

//TODO

În acest capitol este sumarizat întreg proiectul, de la obiective, la implementare, și la relevanta rezultatelor obținute. În finalul capitolului poate exista o subsecțiune de "Dezvoltări ulterioare".

Criterii pentru calificativul *NeSatisfăcător*:

- Concluziile nu sunt corelate cu conținutul lucrării;

Criterii pentru calificativul *Satisfăcător*:

- Concluziile sunt corelate cu conținutul lucrării, însă nu se oferă o imagine asupra calității și relevantei rezultatelor obținute;

Criterii pentru calificativul *Bine*:

- Concluziile sunt corelate cu conținutul lucrării, și se oferă o imagine precisa asupra relevantei și calității rezultatelor obținute în cadrul proiectului.

În Latex este foarte ușor să folosiți referințe într-un mod corect și unitar, fie prin adăugarea unei secțiuni `\begin{thebibliography}` (vezi la sfârșitul acestei secțiuni), fie printr-un fișier separat de tip `bib`, folosind comanda `\bibliography{}`, așa cum procedăm mai jos prin folosirea fișierului "bibliography.bib". În orice caz, în Latex va trebui să folosiți comanda `\cite{}` pentru a adăuga referințe, iar această comandă trebuie folosită direct în text, acolo unde vreți să apară citația, ca în exemplele următoare:

- Articol jurnal: [3];
- Articol conferință: [1];
- Carte: [2];
- Weblink: [4];

**Important:** în această secțiune de obicei apar doar intrările bibliografice (adică doar listarea referințelor). Citarea lor prin comanda `cite` și explicații legate de ele trebuie facute în secțiunile anterioare. Citarea de mai sus a fost facută aici doar pentru exemplificare.

## BIBLIOGRAFIE

- [1] *Proc. 23rd International Symposium on Distributed Computing (DISC, Elche, Spain, September 2009)*, volume 5805 of *Lecture Notes in Computer Science*, Berlin, Germany, 2009. Springer.
- [2] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The L<sup>A</sup>T<sub>E</sub>X Companion*. Addison-Wesley, Reading, Massachusetts, 1993.
- [3] A. Amira H. Baali, H. Djelouat and F. Bensaali. Empowering technology enabled care using iot and smart devices: A review. *IEEE Sensors Journal*, 322(10):891–921, 1905.
- [4] J. Silva-Martinez. Elen-325. introduction to electronic circuits: A design approach,. <http://www.ece.tamu.edu/~spalermo/ecen325/Section%20III.pdf>. Last accessed: 28 February 2018.

## **ANEXE**

Anexele sunt optionale. Ce poate intra în anexe:

- Exemplu de fișier de configurare sau compilare;
- Un tabel mai mare de o jumătate pagină;
- O figura mai mare mai mare de jumătate pagină;
- O secvență de cod sursa mai mare de jumătate pagină;
- Un set de capturi de ecran (“screenshot”-uri);
- Un exemplu de rulare a unor comenzi plus rezultatul (“output”-ul) acestora;
- În anexe intră lucruri care ocupă mai mult de o pagină ce ar îintrerupe firul natural de parcursere al textului.

## **A EXTRASE DE COD**

...