**Sistem de monitorizare intersecție inteligent IoT**

Muste Alexandru Gavril

Departamentul de Inginerie Electrică, Electronică şi Calculatoare

Universitatea Tehnică din Cluj Napoca

Centrul Universitar Nord Baia Mare,

Baia Mare, România

muste.al.alexandru@student.utcluj.ro

Pop Adelin

Departamentul de Inginerie Electrică, Electronică şi Calculatoare

Universitatea Tehnică din Cluj Napoca

Centrul Universitar Nord Baia Mare,

Baia Mare, România

pop.ad.adelin@student.utcluj.ro

**Rezumat**

Proiectul prezent analizează posibilitățile de simulare a unei intersecții prin monitorizarea traficului în contextul aplicațiilor IoT. Sistemul include un server central, un proces pietoni, un proces vehicule, un proces semafor și un proces vizualizator care simulează monitorizarea traficului în timp real, colectând informații cu privire la trecerile în intersecție la culoarea semaforului. Sistemul demonstrează capacitatea de a analiza și interpreta datele stocate într-o bază de date și vizualizate cu pygame. Proiectul demonstrează integrarea tehnologiilor de viziune computerizată și IoT pentru gestionarea traficului.

***Cuvinte cheie***: IoT, monitorizare trafic, Python, vizualizare date, ML

**Abstract**

The present project analyzes the possibilities of simulating an intersection by monitoring traffic in the context of IoT applications. The system includes a central server, a pedestrian process, a vehicle process, a traffic light process, and a viewer process that simulates real-time traffic monitoring, collecting information about crossings at the traffic light. The system demonstrates the ability to analyze and interpret data stored in a database and visualized using pygame. The project showcases the integration of computer vision and IoT technologies for traffic management.

***Keywords***: IoT, traffic monitoring, Python, data visualization, ML

1. **Introducere**

Cu ani în urmă, sistemele inteligente de transport (ITS) au apărut ca o tehnologie transformatoare cu potențial de a crește evoluția sectorului transporturilor și mai multe aplicații Internet of Things (IoT) și Internet of Vehicles (IoV). Învățarea automată ML (Machine Learning) este un tool al inteligenței artificiale (AI), care ajută mașina să ia decizii similare cu oamenii. Algoritmi ML învață din datele de intrare și produc ieșire deterministă fără a fi programați într-un mod explicit.

O rețea IoT este un set de senzori sau dispozitive AI care comunică între ele fără intervenția umană.

Sistemele Inteligente de Transport (ITS) se concentrează pe următoarele probleme:

* Reducerea la minim a accidentelor rutiere;
* Reducerea consumului de combustibil;
* Reducerea timpului petrecut cu ambuteiajele;
* Creșterea siguranței transportului;
* Traversarea unui vehicul de urgență fără întreruperi.

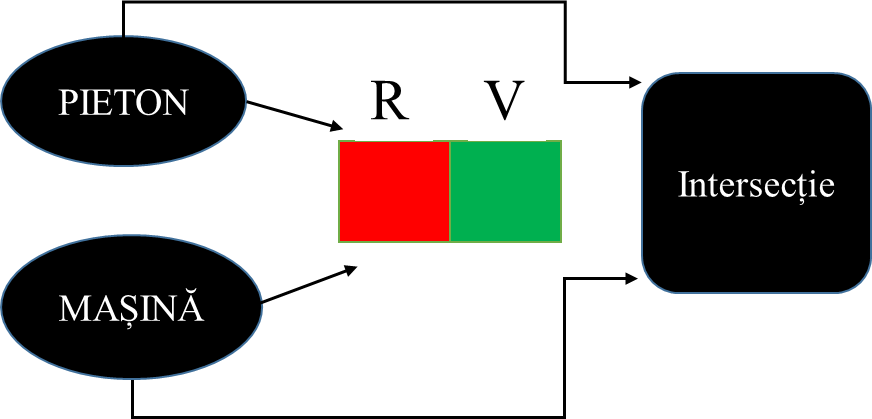
Sistemul unui semafor tradițional se bazează pe principiul unei configurații manuale cu un timp static. Configurația poate fi realizată cu o altă perspectivă, un sistem de semafor dinamic (DTLS) care evită timpul de așteptare inutil pe drumuri datorită comportamentului său inteligent, prin auto-învățare.

1. **Contextul Proiectului**

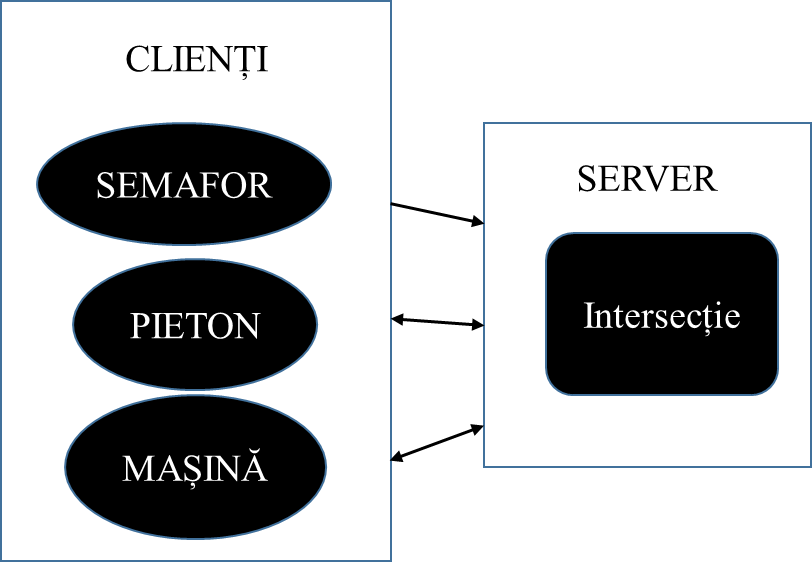
Scopul acestui proiect este de a realiza o posibilă simulare a unei intersecții în Python, bazat pe un sistem dinamic de învățare automată ML (Machine Learning) și folosind algoritmi specifici. Proiectul prezent conține mai multe procese: *un proces server*, *un proces pietoni*, *un proces vehicule*, *un proces semafor* și *un proces vizualizator*, aceste procese comunică cu serverul, iar mai apoi putând fi vizualizat funcționarea intersecției autonome cu pygame.

1. **Metode și Tehnici de programare folosite**

Proiectul Sistem de monitorizare intersecție inteligent IoT se bazează pe următoarele structuri:



*Figura 1 – Structura 1*



Vizualizator

*Figura 2 – Structura 2*

În structurile de mai sus din figurile 1 și 2 sunt ilustrate 2 structuri care reprezintă principiul de funcționare a unei intersecții IoT, în care avem un client pieton, un client mașină, un client semafor, un server intersecție și un vizualizator.

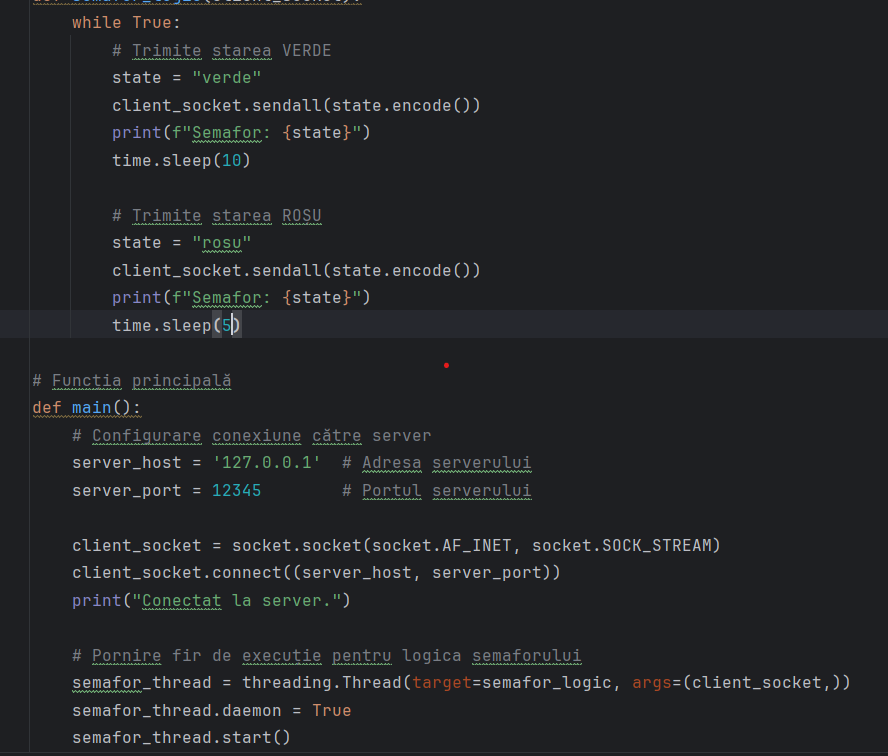
Proiectul a fost gândit să transmită culoarea clientului semafor spre serverul intersecție, iar mai apoi clienții pieton și mașină să interogheze starea semaforului de la server, după să vizualizăm funcționarea intersecției.

Tehnici de programare folosite:

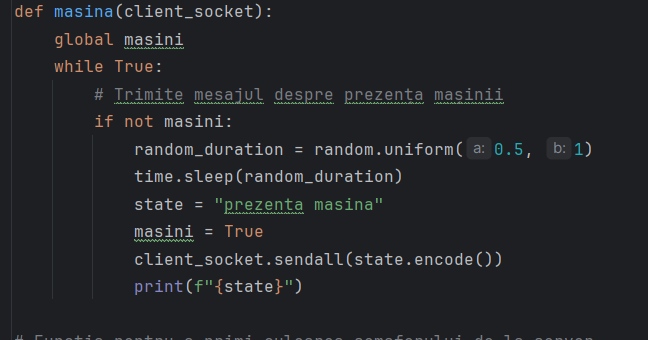
* Limbaj de progrmare Python
* Programare în rețea: Socket
* Programare concurentă: Threading
* Vizualizare simulare : pygame

1. **Rezultate**

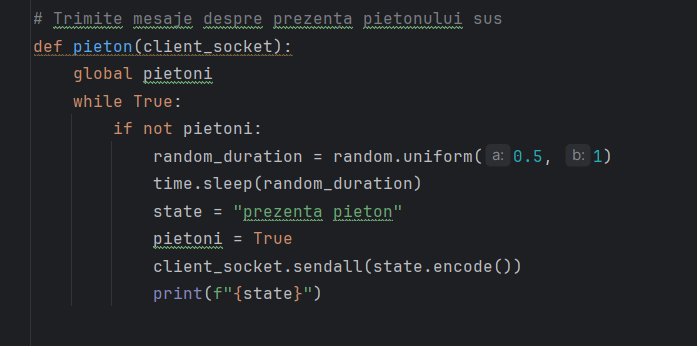
În secțiunea rezultate am obținut toate cele propuse, mai jos am ilustrat câteva imagini cu părți ale codului și cu rezultatele din vizualizator:



*Figura 3 - Semafor*



*Figura 4 – Mașina*



*Figura 5– App. Pieton - furnizează prezenţa pietonilor*

A computer screen with text on it

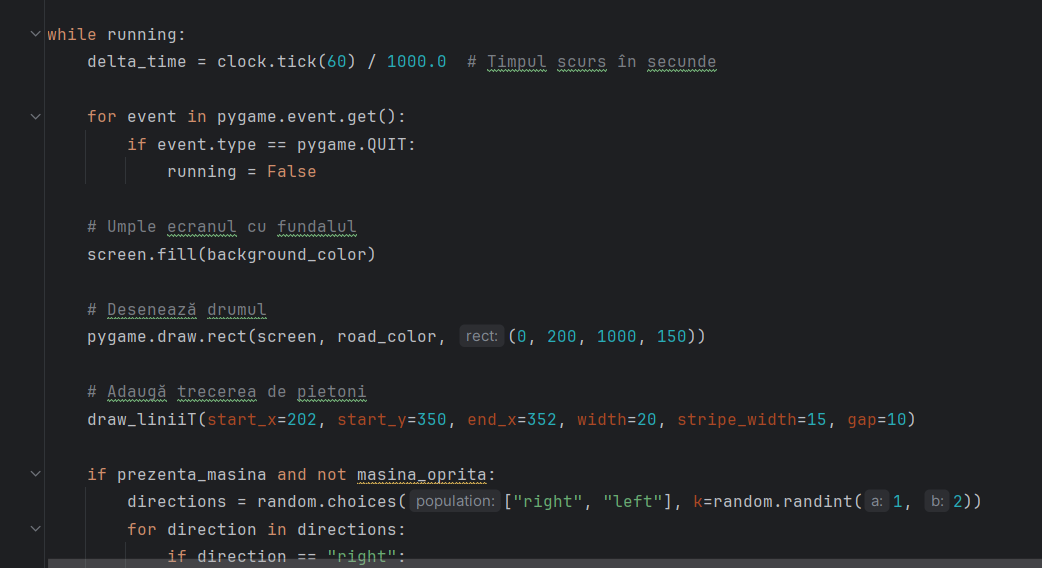
Description automatically generated

*Figura 6 – App Server - Deschiderea Server-ului*

A computer screen shot of a program

Description automatically generated

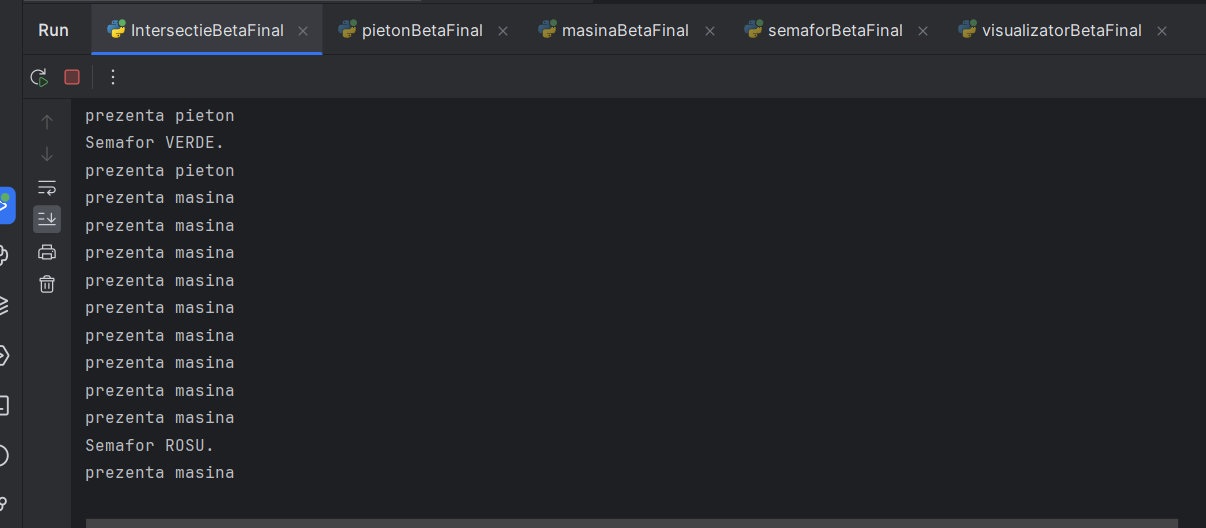
*Figura 7 –App Server - citirea mesajelor de la clienţi*



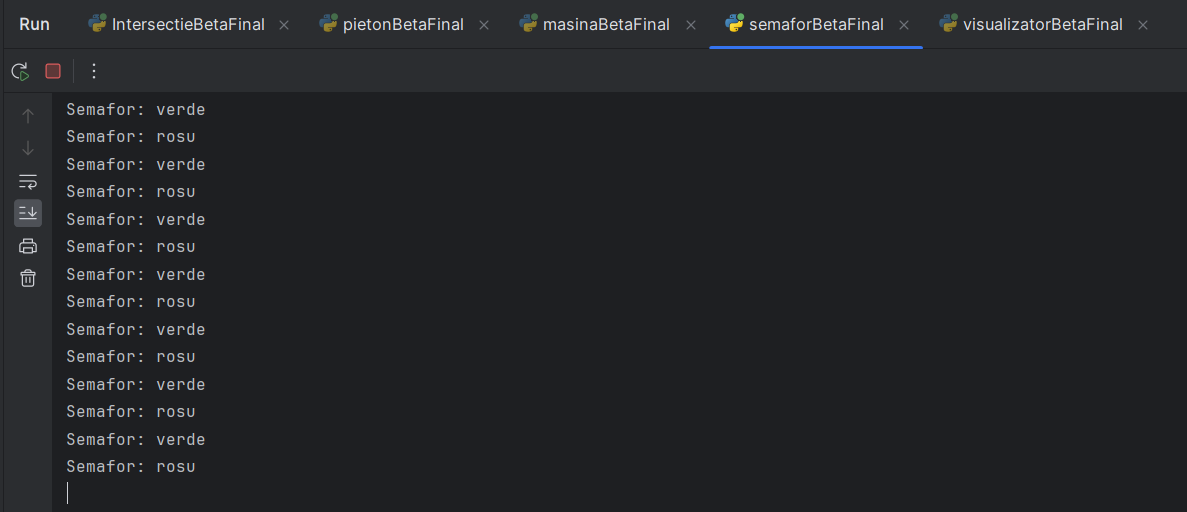
*Figura 8 – App. Vizualizator – Simularea intersecţiei cu date de pe server*

În aplicaţia “Vizualizator” apar doar entităţile care sunt primite de la server, astfel vom avea o simulare reală şi sincronizată cu datele primite de la clienţi care furnizează prezența maşinilor, pietonilor şi culoarea semaforului.

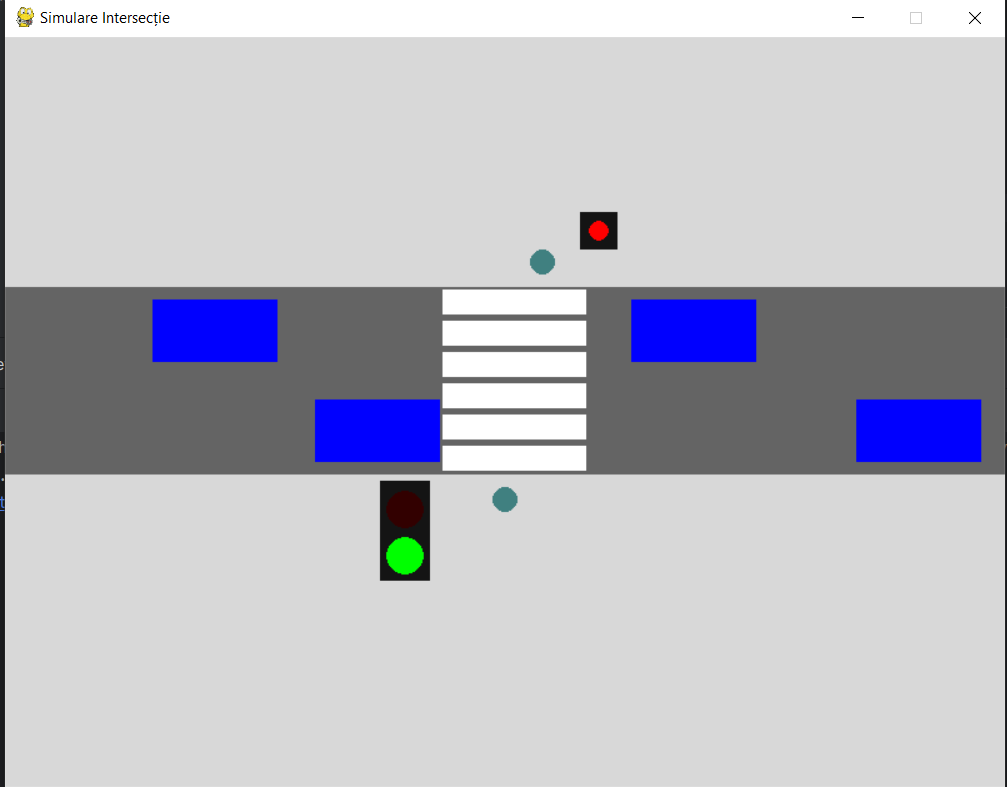
*Rezultate în urma testării*:



*Figura 9 – Afișare consolă Server*



*Figura 10 – Afișare consolă Semafor*



*Figura 11 – Simulare Intersecție*

**Logica evenimentelor**

Pentru a putea detecta dacă sunt vehicule care trec pe roşu sau pentru a detecta accidente între maşină şi pieton am considerat trecerea de pietoni, o zonă la care are acces doar o identitate umană, pieton

În acest proiect am implementat următoareale situaţii pentru fiecare entitate în parte:

Entităţi vehicule :

- Se opresc înainte de trecerea de pietoni la cularea semaforului;

* La schimbarea culorii de la verde la roşu, dacă sunt pe trecere sau au trecut vor continua deplasarea;
* La schimbarea culorii semaforului de la roşu la verde, dacă sunt pietoni încă pe trecere, va aştepta ca pietonul să treacă;

Entităţi pietoni :

-Calcă pe trecere doar dacă culoarea semaforului pentru pietoni este verde;

* La schimbarea culorii de la verde la roşu, dacă sunt pe trecere vor continua deplasarea;
* La schimbarea culorii de la roşu la verde, dacă sunt maşini pe trecere, vor aştepta să treacă, abia apoi va călca pe trecere;
* În cazul în care trece o maşină peste un pieton, acel pieton dispare.

Am realizat surprinderea evenimentului mai puțin fericit “accident” sau de trecere pe roșu a oricărui dintre entități, sub formă de „anomalie” care este realizată aleatoriu şi destul de rar. În care vehiculele nu mai respectă regulile de acces în zona verificată (trecerea de pietoni).

A screenshot of a computer

Description automatically generated

*Figura 12 – Detectarea unui autovehicul care a trecut pe culoarea roşie a semaforului*

A screenshot of a computer

Description automatically generated

*Figura 13 – Detectarea unui autovehicul care a trecut pe culoarea roşie a semaforului şi a unui accident între pieton şi vehicul*

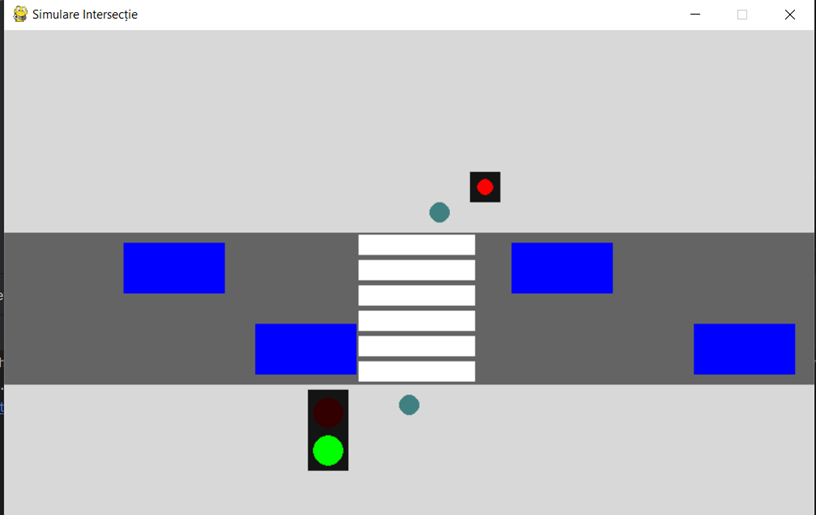
1. **Concluzii și Îmbunătățiri**

Acest proiect scoate în evidență aplicarea unui sistem de monitorizare a intersecție inteligent IoT, printr-o simulare în limbajul de programare Python, venind cu un raspuns în timp real a datelor, fiind vizualizate prin intermediul tehnicii pygame din Python.

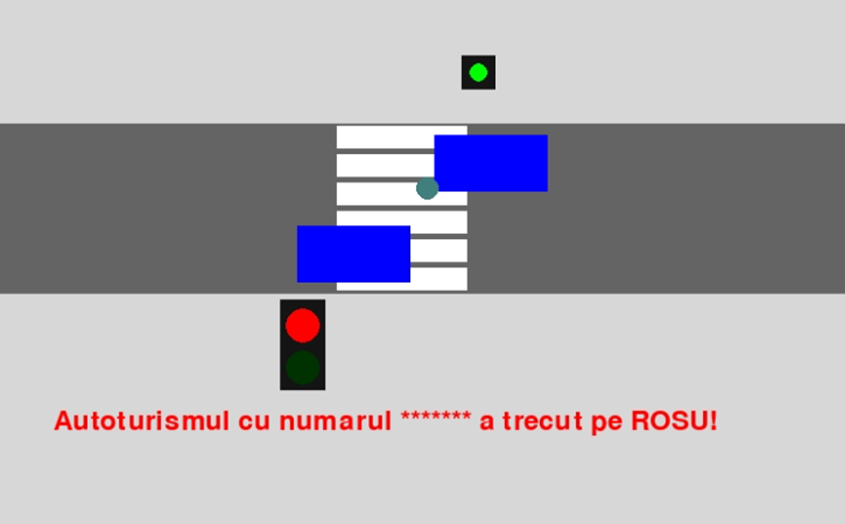
Îmbunătăţirea acestui proiect se poate face prin trimiterea numărului de înmatriculare către o bază de date şi de unde se pot prelucra şi numărul de evenimente într-o anumită perioadă.

O altă îmbunătățire a sistemului este de a duce proiectul la nivel software să gestioneze semafoarele în cazul unei urgențe, la trecerile unui vehicul de urgență prin aducerea în sistem de noi senzori care să detecteze în timp real poziția vehiculului de urgență, iar mai apoi să mențină toate semafoarele pe direcția vehicului de urgență pe verde.

1. **Anexe**
2. Rularea Aplicației:

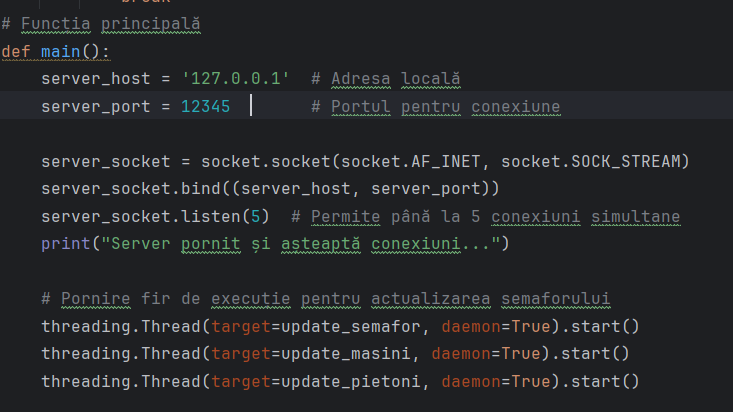


*Figura 14 – Simulare Intersecție fără anomalii*

**

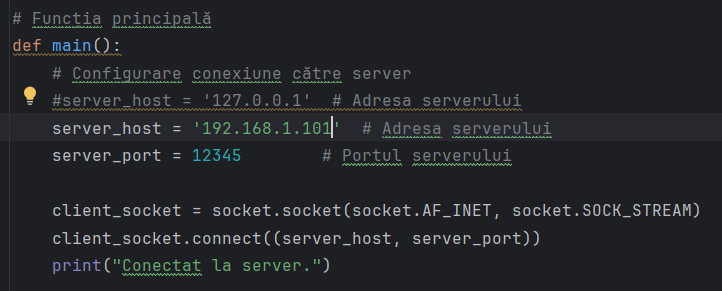
*Figura 15 – Simulare Intersecție cu anomalii*

1. Fragment de cod



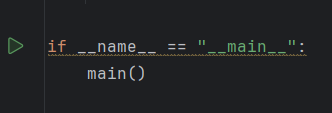
*Figura 16 – Server*

În figura 16 avem funția principală a server-ului, unde se pot declara portul și adresa de conexiune sub formă de parametri la funcție sau în interiorul blocului. Metoda de programare folosită în rețea este UDP care este un protocol de comunicație care funcționează la nivelul Transport din modelul OSI și nu ține cont de conexiune ca și protocolul TCP, am folosit fire de execuție pentru fiecare entitate client pentru o gestionare mai bună.



*Figura 17 – Entitate client*

La fel ca la server se configurează port și adresă spre server pentru conexiune, am folosit același protocol de comunicație UDP, iar după conexiune afișează și un measj în consolă.

**

*Figura 18 – Apelarea funcției principale*

**Bibliografie**

1. <https://ieeexplore.ieee.org/abstract/document/10436105>
2. <https://towardsdatascience.com/traffic-intersection-simulation-using-pygame-689d6bd7687a>
3. Github: <https://github.com/alexMuste/intersectieIoT/tree/Intersectie_IoT_Final>