

# Sistem intelligent de parcare pentru biciclete

June 6, 2025

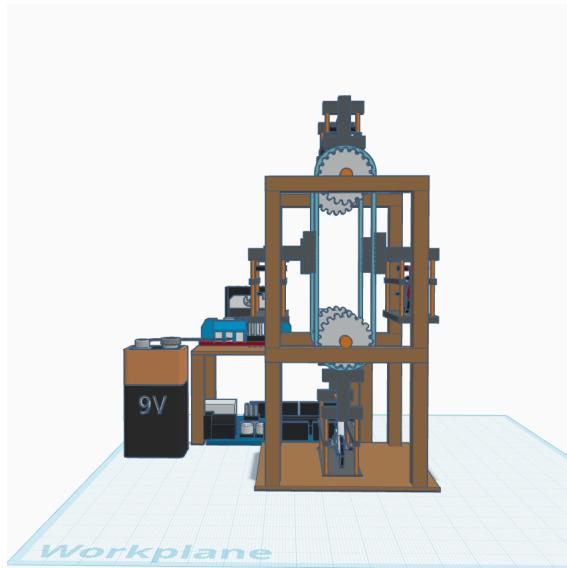


## Cuprins

1. Obiectivele proiectului propus
2. Descrierea domeniului ales și a soluțiilor similare
3. Descrierea soluției implementate cu prezentarea funcționalităților aferente soluției
4. Testarea soluției

## 1 Obiectivele proiectului propus

Figure 1: Designul 3D al proiectului propus



Dezvoltarea unui sistem inovator de parcare pentru biciclete în miniatură cu patru locuri de parcare, cu verificarea locurilor libere și afișarea disponibilității acestora.

Implementarea unui mecanism rotativ eficient, actionat de un motor pas cu pas în tandem cu un driver, pentru a asigura controlul mai precis al vitezei, alimentate de la o sursă de 5V pentru a nu consuma foarte mult, și cu plăcuța Arduino, care să comande mișcarea parcării.

Realizarea unei interfețe în Python pentru ca utilizatorul să poată alege un loc de parcare, cu implementarea unei logici pentru ocuparea și eliberarea locurilor garajului.

## 2 Descrierea domeniului ales și a soluțiilor similare

Automatizarea sistemelor de parcare a evoluat semnificativ în ultimii ani, fiind impulsionată de nevoia de a optimiza spațiul urban și de a reduce timpul de parcare. Aceste soluții sunt deja implementate la scară largă în parcări automate pentru autovehicule, unde platforme robotizate, ascensoare verticale sau sisteme de tip "șiruită" (ca cele din Japonia) mută mașinile într-o rețea compactă de celule de depozitare. De exemplu, în orase precum Tokyo sau New York, aceste sisteme reduc spațiul necesar parcării cu până la 70%, folosind algoritmi pentru a prioriza accesul și distribuția vehiculelor.

Figure 2: Parcare inteligentă Japonia: ECO Park™ — GIKEN



## Adaptarea la biciclete

Pentru biciclete, soluțiile existente se axează în mare parte pe rack-uri fixe sau structuri verticale manuale. Unele orașe europene, precum Amsterdam sau Copenhaga, au introdus parcări subterane cu capacitate mare (3.000–7.000 de biciclete), dar acestea necesită intervenție umană pentru depozitare și extragere.

Figure 3: Parcarea subacavatică din Amsterdam: [Stationsplein](#)



## Inovația proiectului

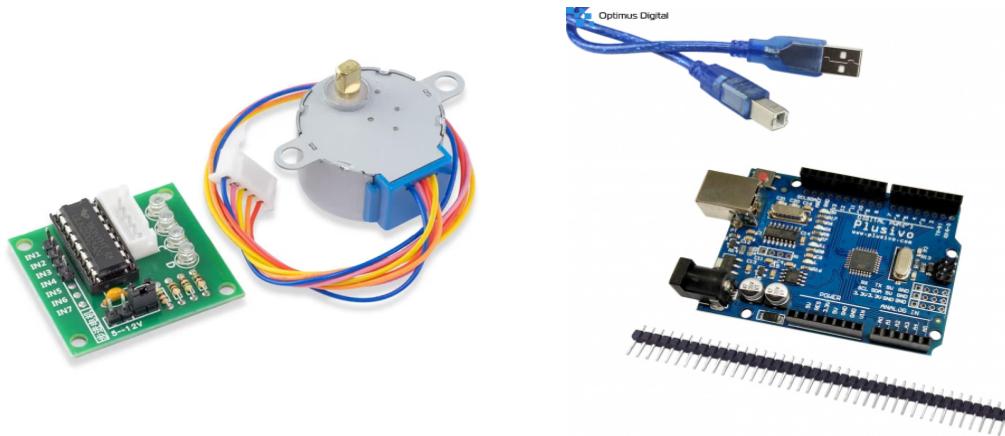
Sistemul propus se diferențiază prin simplitate, cost redus și adaptabilitate. În loc să folosească multiple motoare puternice sau structuri complexe, mecanismul rotativ se bazează pe sincronizarea a patru platforme independente, conectate printr-un sistem de curele și roți dințate. Această abordare permite:

1. Integrarea în spații limitate: Poate fi instalat lângă stații de transport public, clădiri de birouri sau chiar în curți rezidențiale.
2. Compatibilitate cu tehnologii IoT: Se pot furniza date în timp real despre ocuparea locurilor prin intermediul unui dashboard central.
3. Extindere modulară: Adăugarea de etaje sau brațe suplimentare nu necesită decât replicarea componentelor existente și ajustarea codului de control (de exemplu, modificarea numărului de pași ai motorului pentru a acoperi un cerc mai mare).

## 3 Descrierea soluției implementate cu prezentarea funcționalităților aferente soluției

Soluția dezvoltată constă într-un sistem automatizat de parcare pentru biciclete, construit sub forma unui mecanism rotativ cu patru roți printate 3D, conectate două către două prin curele de transmisie și două bare de lemn. De aceste benzi sunt atașate patru platforme destinate parcării bicicletelor. Mișcarea întregului ansamblu este realizată cu ajutorul unui motor pas cu pas de tip

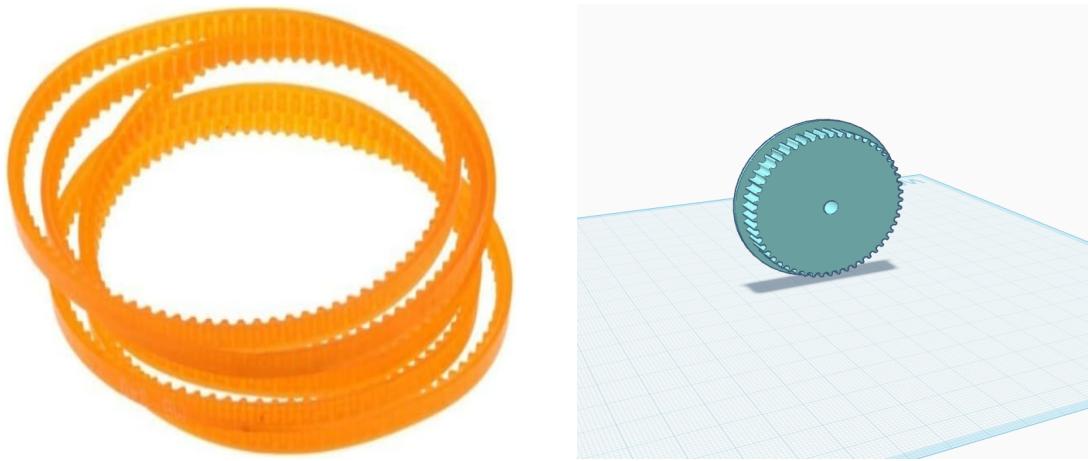
28BYJ-48, cu un raport de reducere 1:64, controlat printr-un driver ULN2003 și o placă Arduino. Microcontrollerul este alimentat prin cablu de la portul USB al calculatorului, iar driverul este conectat la sursa de 5V a plăcuței Arduino, la Ground și la pinii digitali: 8,9,10,11.



(a) Driver ULN2003, motor pas cu pas 28BYJ-48

(b) Microcontroller

Figure 4: Mechanism actionare



(a) Curea de transmisie

(b) Roată 3D

Figure 5: Mechanism rotativ

Am implementat o interfață în Python cu Tkinter, pentru a oferi utilizatorului control asupra parcării. Aceasta poate să aleagă orice loc liber de parcare, însă este obligatorie introducerea unui nume și a unei parole pentru a putea securiza bicicletă. Această acțiune este indicată de GUI prin afișarea numelui persoanei ce ocupă în prezent fiecare loc de parcare. Codul selectat trebuie introdus pentru a își putea extrage bicicletă înapoi, moment în care parcarea se eliberează. Prin pachetul Serial transmitem către plăcută arduino ce are încărcată codul de mai sus, locul de parcare ce tocmai a fost închiriat sau eliberat, microcontrollerul având rolul de a calcula mișcarea necesară de a ajunge de la locul curent până la locul dorit de utilizator ( $[4 + locul_{dorit} - locul_{current}] \% 4$ , unde 4 este numarul total de locuri) și de a transmite motorului comanda, că acesta să realizeze rotația calculată, pentru a aduce platforma dorită jos, în punctul de acces pentru cicliști.

```
import tkinter as tk
import serial
import time

# == Conexiunea cu Arduino ==
try:
    arduino = serial.Serial('COM3', 9600)
    time.sleep(2)
    print("Connected to Arduino.")
except Exception as e:
    arduino = None
    print("Could not connect to Arduino:", e)

# == Date Parcare ==
parking_spots = [
    {'taken': False, 'name': '', 'code': ''},
    {'taken': False, 'name': '', 'code': ''},
    {'taken': False, 'name': '', 'code': ''},
    {'taken': False, 'name': '', 'code': ''},
]

# == Comanda spre Arduino ==
def send_command(slot):
    print(f"Trimis: Parcare {slot}")
    if arduino:
        arduino.write(f"{slot}\n".encode())

# == Centrare ==
def center_window(window):
    window.update_idletasks()
    width = window.winfo_width()
    height = window.winfo_height()
    x = (window.winfo_screenwidth() // 2) - (width // 2)
    y = (window.winfo_screenheight() // 2) - (height // 2)
    window.geometry(f'+{x}+{y}')

# == Click asupra unei parcări ==
def handle_parking_spot(slot):
    current = parking_spots[slot]
    if not current['taken']:
        # Dialog pentru rezervare
        dialog = tk.Toplevel(root)
        dialog.title("Introduceti detalii")

        # Introducerea Numelui
        tk.Label(dialog, text="Nume:").grid(row=0, column=0, padx=5, pady=5)
        name_entry = tk.Entry(dialog)
        name_entry.grid(row=0, column=1, padx=5, pady=5)
```

```
# Alegerea Codului
tk.Label(dialog, text="Cod:").grid(row=1, column=0, padx=5, pady=5)
code_entry = tk.Entry(dialog, show="*")
code_entry.grid(row=1, column=1, padx=5, pady=5)

# Button OK
def ok_clicked():
    name = name_entry.get().strip()
    code = code_entry.get().strip()
    if name and code:
        parking_spots[slot]['taken'] = True
        parking_spots[slot]['name'] = name
        parking_spots[slot]['code'] = code
        buttons[slot-1].config(text=f"Parcare {slot}\n{name}", bg="#ff9999")
        send_command(slot)
        dialog.destroy()
    else:
        tk.Label(dialog, text="Va rugam, introduceti toate detaliile!", fg="red").grid(row=2, columnspan=2)
tk.Button(dialog, text="OK", command=ok_clicked).grid(row=3, columnspan=2, pady=5)

center_window(dialog)
dialog.grab_set()

else:
    # Introducerea Codului
    dialog = tk.Toplevel(root)
    dialog.title("Introduceti Codul")
    tk.Label(dialog, text="Cod:").grid(row=0, column=0, padx=5, pady=5)
    code_entry = tk.Entry(dialog, show="*")
    code_entry.grid(row=0, column=1, padx=5, pady=5)

    # Button OK
    def ok_clicked():
        entered_code = code_entry.get().strip()
        if entered_code == parking_spots[slot]['code']:
            # Free the parking spot
            parking_spots[slot]['taken'] = False
            parking_spots[slot]['name'] = ''
            parking_spots[slot]['code'] = ''
            buttons[slot-1].config(text=f"Parcare {slot}", bg="#4da6ff")
            send_command(slot)
            dialog.destroy()
        else:
            tk.Label(dialog, text="Cod gresit!", fg="red").grid(row=1, columnspan=2)
    tk.Button(dialog, text="OK", command=ok_clicked).grid(row=2, columnspan=2, pady=5)

    center_window(dialog)
    dialog.grab_set()

# == Setup GUI ==
root = tk.Tk()
root.title("Selector Parcare")
root.geometry("300x400")
root.configure(bg="#f5f5f5")
```

```
# Centerare
root.update_idletasks()
width = root.winfo_width()
height = root.winfo_height()
x = (root.winfo_screenwidth() // 2) - (width // 2)
y = (root.winfo_screenheight() // 2) - (height // 2)
root.geometry(f'{x}x{y}+{x}+{y}')

# == Interfata ==
title = tk.Label(root, text="Selecteaza o parcare",
font=("Helvetica", 18, "bold"), bg="#f5f5f5", fg="#333")
title.pack(pady=30)

# == Butoane ==
buttons = []
for i in range(1, 5):
    button = tk.Button(root, text=f"Parcare {i}",
                        font=("Helvetica", 14),
                        bg="#4da6ff", fg="#4C7D4C",
                        activebackground="#3399ff",
                        relief="flat",
                        padx=10, pady=5,
                        command=lambda n=i: handle_parking_spot(n))
    button.pack(pady=10)
    buttons.append(button)

# == Run GUI Loop ==
root.mainloop()

#include <Stepper.h>
const int stepsPerRevolution = 2048;
const int stepsPerSlot = 1500;

Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);

int currentSlot = 1;

void setup() {
    Serial.begin(9600);
    myStepper.setSpeed(15); // RPM
    Serial.println("Ready for slot selection (1-4)");
}

void loop() {
    if (Serial.available()) {
        int targetSlot = Serial.parseInt();
        if (targetSlot >= 1 && targetSlot <= 4 && targetSlot != currentSlot){
            // Calculul numarului de spatii, pasi
            int slotsToMove = (4 + targetSlot - currentSlot) % 4;
            int stepsToMove = -slotsToMove * stepsPerSlot;
            // Negativ pentru a se misca 28BYJ-48 in sensul acelor de ceasornic
            myStepper.step(stepsToMove);
            currentSlot = targetSlot;

            Serial.print("Moved to slot: ");
            Serial.println(currentSlot);
        }
    }
}
```

Table 1: Lista componentelor

Componentă	Sursă	Pret (lei)
Placă de Dezvoltare Compatibilă cu Arduino UNO (ATmega328p și CH340) și Cablu 50 cm	Optimus Digital	39.27
Motor pas cu pas 28BYJ-48	Emag	20.33
Driver ULN2003		
Breadboard	Optimus Digital	4.56
Fire mamă-tată	Optimus Digital	9.99
Fire tată-tată	Optimus Digital	7,98
Incarcator universal adaptor AC / DC cu 6 conectori, putere 12W, tensiune reglabilă 3-12V, 1.2A, compatibilitate universală	Optimus Digital	59.79

Table 2: Lista materialor pentru machetă:

Material	Sursă	Pret (lei)
2 x Curea de transmisie	Fizic Piața Obor	20
4 x Roată dințată printată 3D	Link Design	0
4 x Platformă parcare printată 3D	Link Design	0
4 x Suport platformă printată 3D	Link Design	0
2x Rigla rindeluită lemn balsa 1000 x 15 x 15 mm	Dedeman	24.40
Rigla rindeluită lemn balsa 1000 x 10 x 10 mm	Dedeman	8.30
Placaj pânzat 25x35	Profiart	17.20
Betășoare rotunde 0.4x10cm	Jumbo	4.99
2m Cablu izolat	Fizic magazin Tei	4
3 x Bicicletă în miniatură	Bavixo	60
5 x Superglue lichid	Emag	23.75
Superglue gel	Emag	6.5
Poxipol	Emag	28.60
Cutter de precizie	Profiart	38.90
Smirghel abraziv 60 granulatie	Emag	59.08

## 4 Testarea soluției

Testarea sistemului hardware s-a concentrat pe validarea funcționalității motorului pas cu pas, preciziei de poziționare și integrării cu logica software. Am folosit un motor 28BYJ-48 controlat prin driver ULN2003 și o placă Arduino Uno, programată să execute rotații precise în funcție de disponibilitatea locurilor de parcare. Pentru a verifica corectitudinea mișcării, am marcat pozițiile platformei și am testat răspunsul la comenzi de rotație (orar/antiorar).

Am mai testat scenarii de suprasarcină prin funcționarea continuă a motorului timp de 30 de minute, monitorizând încălzirea și stabilitatea. Interfața Python a fost conectată la Arduino prin comunicare serială, verificând sincronizarea între comenzi software și acțiunile hardware. Testele au confirmat că sistemul poate gestiona ciclurile de rotație fără erori, iar starea locurilor este actualizată în timp real.