THE UNIVERSITY OF DA NANG

DA NANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

FACULTY OF MECHANICAL ENGINEERING

Capstone Project

Major: Mechatronics Engineering

Topic: Design of A Smart Parking System

Adviser: **Ph.D TRAN DINH SON**

Co-Adviser: **Mr. NGUYEN THANH TAN**

Reviewer: **Ph.D LE HOAI NAM**

Student: **PHAM MINH TOAN**

101200419

Student: **NGUYEN NGOC QUANG**

101200408

Class: **20CDTCLC3**

Da Nang, June 16/2025

SUMMARY

Topic: Design of A Smart Parking System

Students: Pham Minh Toan ID: 101200419 Class: 20CDTCLC3

Nguyen Ngoc Quang ID: 101200408 Class: 20CDTCLC3

Adviser: Ph.D Tran Dinh Son

Reviewer: Ph.D Le Hoai Nam

In response to the increasing demand for intelligent infrastructure in urban areas, the multi-level smart parking system project aims to provide an efficient and automated solution to current parking challenges. By integrating modern technologies such as image processing for license plate recognition and real-time monitoring, this system enhances space utilization, minimizes manual intervention, and reduces congestion in high-traffic zones.

The project is the result of careful research, hardware and software integration, and system optimization tailored for both educational purposes and practical application. It emphasizes the advantages of automation, the potential challenges encountered during development, and proposes future improvements for broader deployment. This work not only supports smart city initiatives but also contributes to the evolving field of intelligent transportation systems.

THE UNIVERSITY OF DA NANG
DA NANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

SOCIALIST REPUBLIC OF VIETNAM
Independence – Liberty - Happiness

GRADUATION PROJECT TASK (CAPSTONE PROJECT)

TT	Name	Student ID	Class	Major
1	Pham Minh Toan	101200419	20CDTCLC3	Mechatronics Engineering
2	Nguyen Ngoc Quang	101200408	20CDTCLC3	Mechatronics Engineering

1. Project Title:

Design of A Smart Parking System

2. Project Category: ☐ An intellectual property agreement has been signed regarding the project outcomes.

3. Initial Figures and Data:

4. Content of the Explanation and Calculation Sections:

1. Chapter 1: Overview
2. Chapter 2: State of the art
3. Chapter 3: Calculation and design of mechanical system
4. Chapter 4: Calculation and design of control system
5. Chapter 5: Conclusion

5. Drawings and Graphs:

1. General Assembly Drawing (1 A0)
2. Assembly Drawing (1 A0)
3. Detailed Drawings (2 A0)
4. Electrical Diagram (1 A0)
5. Algorithm Flowchart (1 A0)
6. Mechanical Structure Drawing (1 A0)
7. Kinematic Diagram (1 A0)

6. Adviser: Ph.D Tran Dinh Son

7. Project task date: ……/……./202…..

8. Project Completion Date: ……/……./202…..

Da Nang, June 16/2025

Head of Department …….. **Adviser**

PREFACE

In the context of rapid urbanization, efficient parking management has become a major challenge for modern cities. Traffic congestion, the shortage of parking spaces, and the time wasted in searching for available spots have driven the demand for integrating technology into parking management systems. Based on this reality, the project titled "Design of a Smart Parking System" was developed to propose an optimal solution that combines automation and electronic technologies to enhance efficiency, save space, and provide a convenient user experience.

This smart parking system is designed with an automated operating mechanism, integrating modules such as surveillance cameras, a time-tracking and electronic payment system, and a vertical lift mechanism to maximize the use of vertical space. With its capability for fast and accurate data processing, the system not only minimizes manual intervention but also ensures security and transparency throughout the usage process.

The project aims to achieve the following objectives:

- Space optimization through a vertical column design, suitable for urban areas with limited land availability.
- Comprehensive automation from vehicle recognition and fee calculation to data storage, minimizing errors and reducing waiting time.
- User-friendly interface with simple operation and support for multiple payment methods (cash, card, e-wallet).

With these advanced features, the project is expected to contribute to solving the parking management problem in major cities, while also introducing a new approach to applying Industry 4.0 technologies in the field of urban transportation.

COMMITMENT

We hereby affirm that this graduation thesis is the result of our own study, research, and sincere effort. Throughout the process, we have strictly adhered to the university's regulations on academic integrity and ethics. All references and cited materials have been properly acknowledged, and we fully respect the intellectual property rights of the original authors. We take full responsibility for any issues that may arise regarding this matter.

Da Nang, June 16/2025

Students

CONTENTS

LIST OF TABLES AND FIGURES

CHAPTER 1: OVERVIEW

1. Introduction to automation systems

1.1. Introduction

– Automation (or automatic control) refers to the use of control systems for machines and processes in both industrial and domestic applications, with the aim of minimizing human intervention. In practice, there are numerous automated systems employed in production lines, boilers, furnaces, telephone networks, ship stabilization, aircraft, and more.

Figure 1.1 Automation system

– In a comprehensive automated system, controlling the entire production process plays a crucial role. The flexibility of the process is reflected in the continuity, rhythm, ratio, and parallelism of the motion flows.

1.2. Role and Significance of Automation Systems

– Automation of production processes allows for reduced costs and increased labor productivity. Production processes are subject to economic laws. The demand for higher product quality also increases costs in terms of increasing the complexity of processing (in terms of training workers and teams, equipment costs, etc.).

- Automation enables improved production conditions. Automated production processes eliminate the difficulties associated with manual labor. At the same time, it improves working conditions for workers, especially in hazardous, heavy, and repetitive tasks.

1.3. Challenges in applying automation systems

- Security threats and vulnerabilities.
- Unpredictability and excessive development costs.
- High initial costs.

1.4. Controllers used in automation

- **On/off controllers:** One of the simplest controllers is the on/off controller. For example, to control household thermal devices such as rice cookers, it requires a thermal relay to turn the power on and off. In the thermal relay, there is a resilient spring, and when the temperature increases, the spring expands.

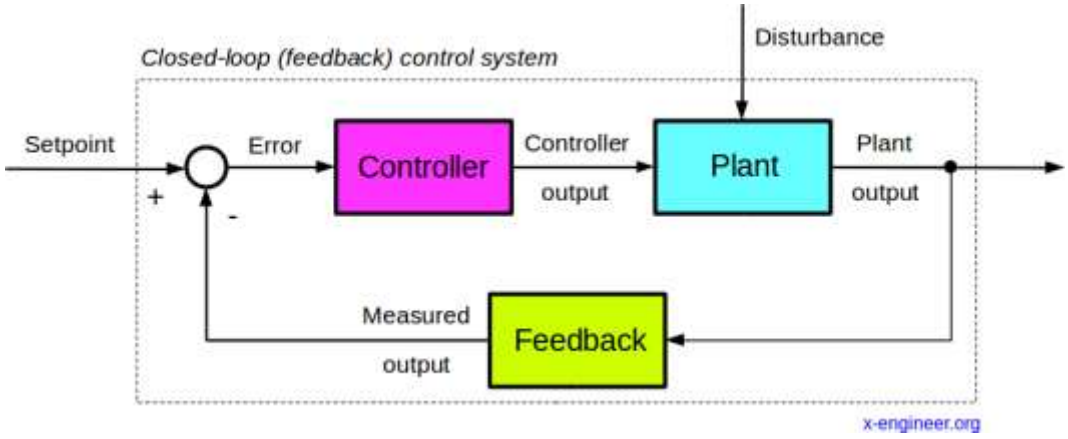

Figure 1.2 On/off controllers

- **Sequential and logic sequential controllers:** Sequential control can be a fixed sequence or the logic will perform different actions depending on different system states. An example of a non-fixed but sequential sequence is a timer on a lawn sprinkler. An early development of continuous control was relay, whereby electrical relays engaged electrical contacts that could start or interrupt power to a device.

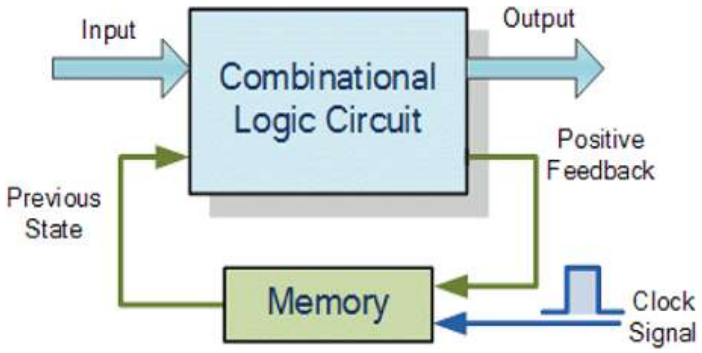

Figure 1.3 Sequential and logic sequential controllers

– **Computer-based controllers:** Computers can perform both continuous control and feedback control, and often a single computer will do both in an industrial application. Programmable logic controllers (PLCs) are a type of special-purpose microprocessor that replaces many of the components such as timers and sequencers that were used in systems. General-purpose process control computers have increasingly replaced individual controllers, with a single computer being able to perform the functions of hundreds of controllers.

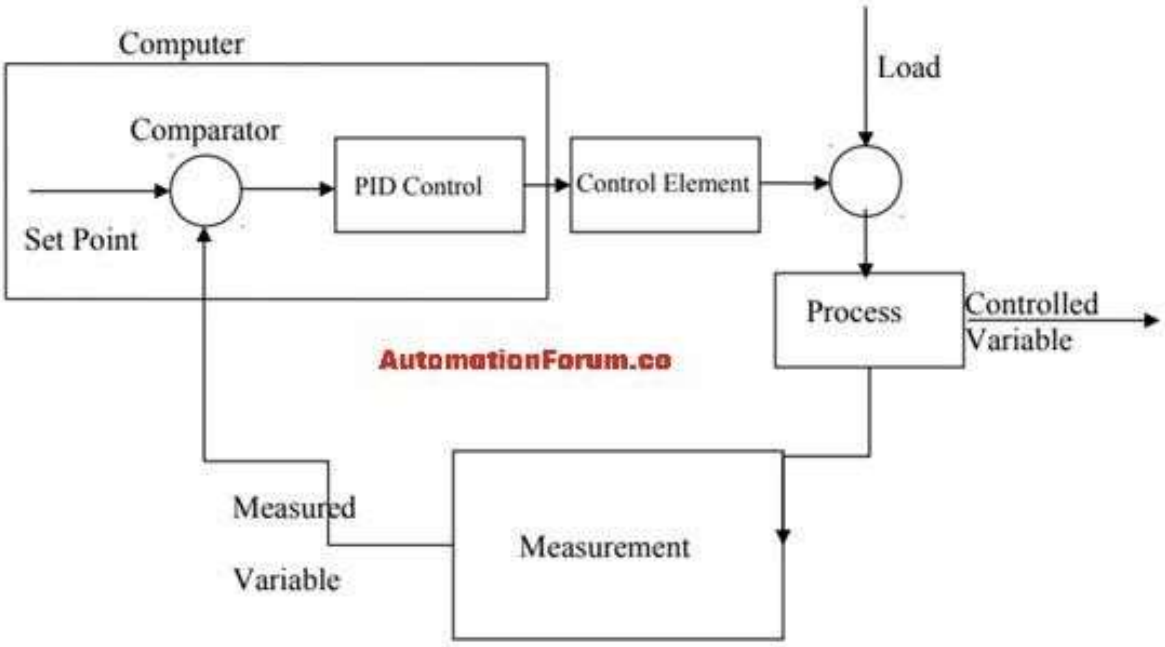

Figure 1.4 Computer-based controllers

2. Introduction to the Smart Parking System

In the context of rapid urbanization and the fast-growing density of vehicles, the demand for efficient, convenient, and modern parking management solutions has become increasingly urgent. The smart parking system has emerged as an advanced technological solution aimed at optimizing vehicle search, reservation, and management in both public and private parking areas.

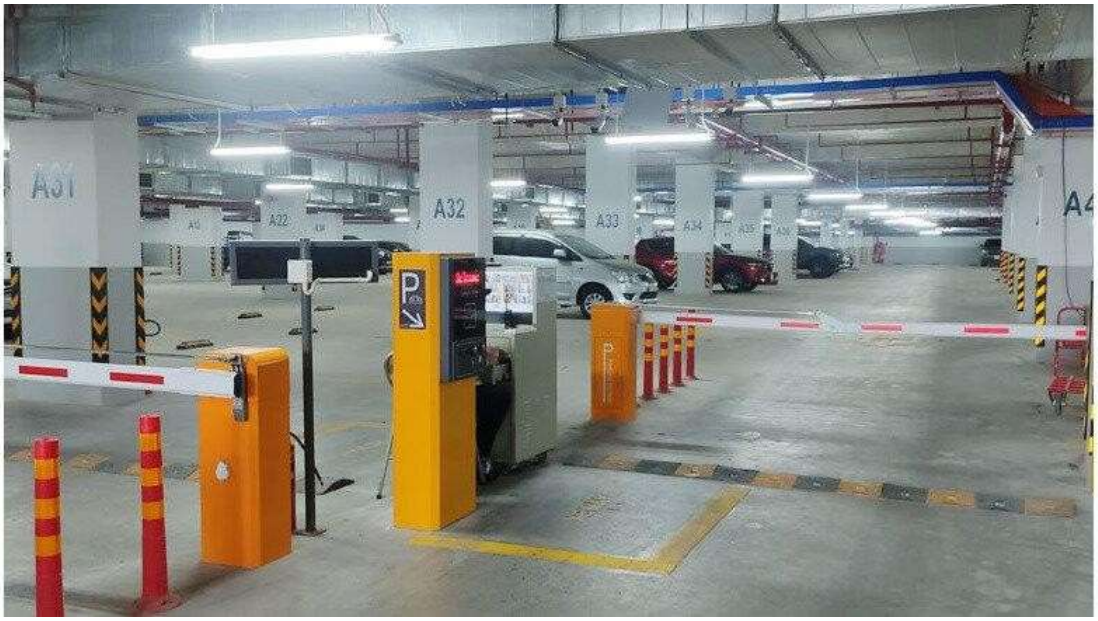

Figure 1.5 Traditional parking system

The smart parking system typically integrates technologies such as sensors, the Internet of Things (IoT), artificial intelligence (AI), GPS positioning systems, and centralized management software. Users can easily search for and reserve parking spaces via a mobile app or website, while the system automatically recognizes license plates, calculates fees, and guides parking quickly and accurately. Additionally, parking management can monitor available spaces, vehicle traffic flow, and process data in real-time.

The implementation of smart parking systems not only helps reduce traffic congestion, save time and fuel for drivers, but also contributes to improving urban management efficiency, reducing environmental pollution, and laying the foundation for the development of future smart cities.

3. Reason for choosing the topic

Figure 1.6 Multi-storey parking system

– The selection of a multi-level smart parking system is driven by practical needs and the outstanding advantages this model offers in the context of modern urban environments. Specifically, the main reasons include:

+ Urban space optimization: In major cities, the availability of land for parking is increasingly limited, while the number of private vehicles continues to rise. A multi-level parking system utilizes vertical space, saving ground area while increasing the number of vehicles that can be accommodated simultaneously.

+ Integration of smart technologies for management and operation: The system incorporates sensors, cameras, and automated or semi-automated control software to efficiently and accurately manage vehicle entry and exit, identify parking locations, calculate fees, and ensure security.

+ Reduced search time and minimized congestion: Drivers can easily find and reserve parking spots in advance through a mobile application. This not only saves time but also reduces traffic congestion around parking areas.

+ Enhanced urban aesthetics and modernization: Multi-level parking structures are designed with a modern, compact, and uniform architectural style, contributing to improved urban landscapes and aligning well with the “smart city” model.

+ High scalability and customization potential: The modular design of multi-level systems allows for easy expansion or relocation based on the development needs of the area, making them suitable for a wide range of scales, from small to large.

– With these advantages, the multi-level smart parking system is a suitable and essential solution for effectively addressing transportation and urban management challenges in the near future.

4. Target customer profile

4.1. Demographics

Factor	Detailed Description
Age	25 – 60 years old. This age group mostly owns private cars, has stable income, and frequently commutes in urban areas.
Gender	Both male and female. However, men may account for a higher percentage among regular car users or owners due to lifestyle and higher mobility needs.
Income	Middle to High – High (from 15 million VND/month and above). This group can afford smart parking services that offer convenience, technology, and efficiency.
Occupation	Office workers, engineers, company directors, government employees, technical specialists, ride-hailing drivers (GrabCar, BeCar), and urban residents.
Geographic Area	Living and working in major cities such as Ho Chi Minh City, Hanoi, Da Nang – where public parking spaces are extremely limited. Especially near business centers, office buildings, hospitals, universities, and apartment complexes.

4.2. Psychographics

Factor	Detailed Description
Lifestyle	Busy, modern lifestyle with high mobility. These customers value convenience, time-saving, and technological integration in everyday services.
Values	Emphasis on comfort, safety, efficiency, and environmental sustainability. Willing to pay more for services that provide peace of mind and ease of access.
Needs & Interests	Interested in smart, tech-driven solutions. Desire for automated, reliable, and secure parking experiences. Preference for mobile apps and digital control.
Preferences	Prefer cashless payment, real-time availability updates, and minimal waiting. Favor parking areas with good lighting, security, and proximity to destinations.

4.3. Consumer Behavior

Factor	Detailed Description
Shopping Habits	Accustomed to using technology in daily activities (e.g., ride-hailing, food delivery, navigation apps). Open to trying new digital or automated services.
Usage Channels	Frequently use smartphones, mobile apps, and smart systems to access services. Rely on search engines, Google Maps, Zalo, Facebook, and review platforms for decisions.
Decision Influencers	Influenced by convenience, time savings, proximity to destination, real-time availability, ease of payment, and overall safety of the parking environment.
Pain Points	Often frustrated by long search times for parking, high congestion in city centers, security concerns, and lack of transparency in pricing or availability.

4.4. Current Issues

- Insufficient parking spaces in central areas: In districts such as District 1, 3, and Tân Bình (Ho Chi Minh City) or densely populated areas in Da Nang, it is very difficult to find legal parking spaces, especially during peak hours.
- Long time spent searching for parking: Drivers often have to circle around for 10–20 minutes to find a parking lot or an available spot.
- Unsafe parking locations: Many parking lots lack surveillance cameras and security personnel, making customers worry about potential vehicle theft or damage.
- Lack of real-time information: Users do not know whether there are available spots in a parking lot, leading to wasted time traveling there only to find it full.
- Inconvenient payment methods: Some parking areas still use manual cash collection without a clear billing system.

4.5. Market Gaps

- Lack of multi-level and automated parking models that can increase capacity without requiring large ground space.
- Lack of real-time parking management systems that can be integrated with smartphones or urban traffic infrastructure.
- Most current parking lots still operate manually, without incorporating modern management technologies (such as vehicle detection sensors, license plate recognition AI, or automated navigation systems).

4.6. Unfulfilled customer needs

- Easily find and reserve parking spots in advance via mobile phone.
- Space-saving parking solutions suitable for densely populated residential areas or shopping centers.
- Modern and fast user experience: no staff needed, no waiting in line or taking physical tickets.
- High safety and security: equipped with cameras, alarms, and continuous monitoring by security personnel.

CHAPTER 2: STATE OF THE ART

1. Overview of the Smart Parking System

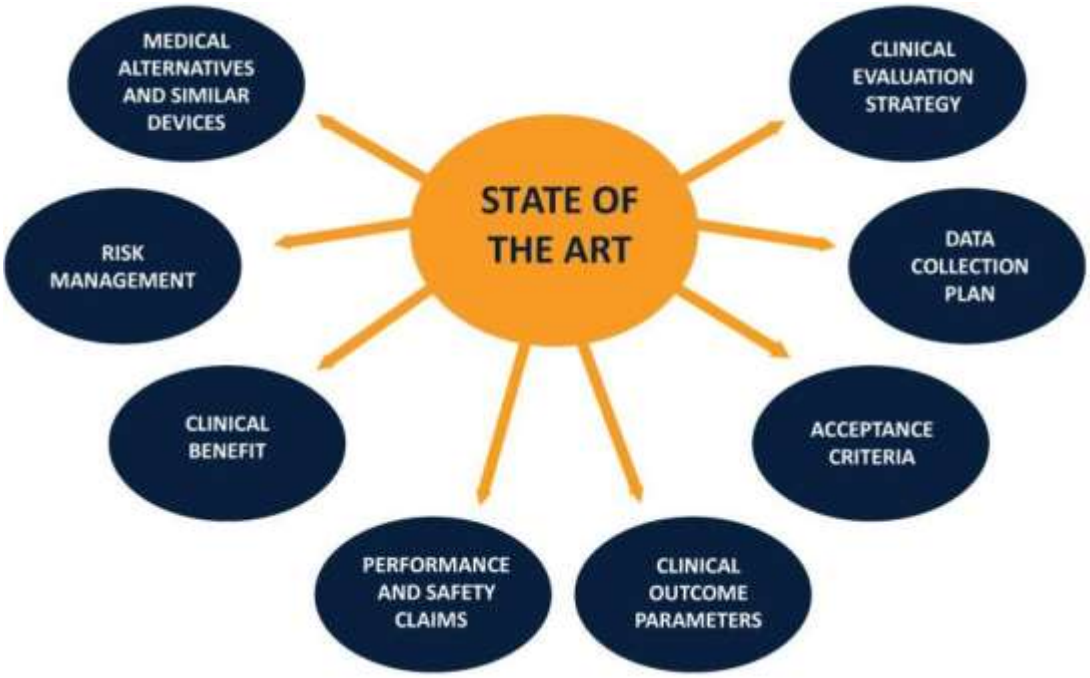

Figure 2.1 State of the art

A smart parking system is a solution that applies modern technological advancements to optimize parking space utilization, enhance user experience, and minimize the limitations of traditional parking systems—such as space shortages, traffic congestion in central areas, and low operational efficiency. At its core, a smart parking system integrates sensor technology, automated control, communication networks, management software, and mechanical structures to ensure flexibility, safety, efficiency, and environmental friendliness.

In particular, with the rapid pace of urbanization and the increasing demand for parking amidst limited land availability, vertical smart parking models—also known as multi-level parking systems—are considered a strategic and sustainable solution aligned with the development trends of smart cities.

2. Typical Smart Parking System Models Around the World

2.1. Automated Rotary Tower Parking Model in Japan

Figure 2.2 Rotary Tower Parking System

- Japan is a pioneer with extensive experience in the development of automated parking systems, particularly the rotary tower model. This solution is extremely effective in areas with limited ground space but potential for vertical development.
- System Components:
  + A vertical tower system with a circular or elliptical structure.
  + Vehicle trays installed around a central rotating vertical axis.
  + A vehicle entry cabin located on the ground floor.
  + A central control unit using a PLC or microcontroller.
- Operating Principle: When a vehicle is placed into the entry cabin, sensors and safety checks verify that the vehicle is properly positioned and stable. The system then controls the rotating axis to turn the tray and move the vehicle to an available parking slot. This process is fully automated, requiring no driver intervention.
- Advantages:
  + Saves floor space: A rotary tower system can accommodate 10–20 vehicles within an area equivalent to only two standard parking spaces.
  + Fully automated operation, requiring no staff assistance.
  + Fast vehicle retrieval and delivery, typically under 120 seconds.

- Disadvantages:
  + High investment and maintenance costs.
  + Limitations on vehicle weight and size.
  + Complex installation process requiring a strong foundation.

2.2. IoT-Integrated Smart Parking System Model in Germany

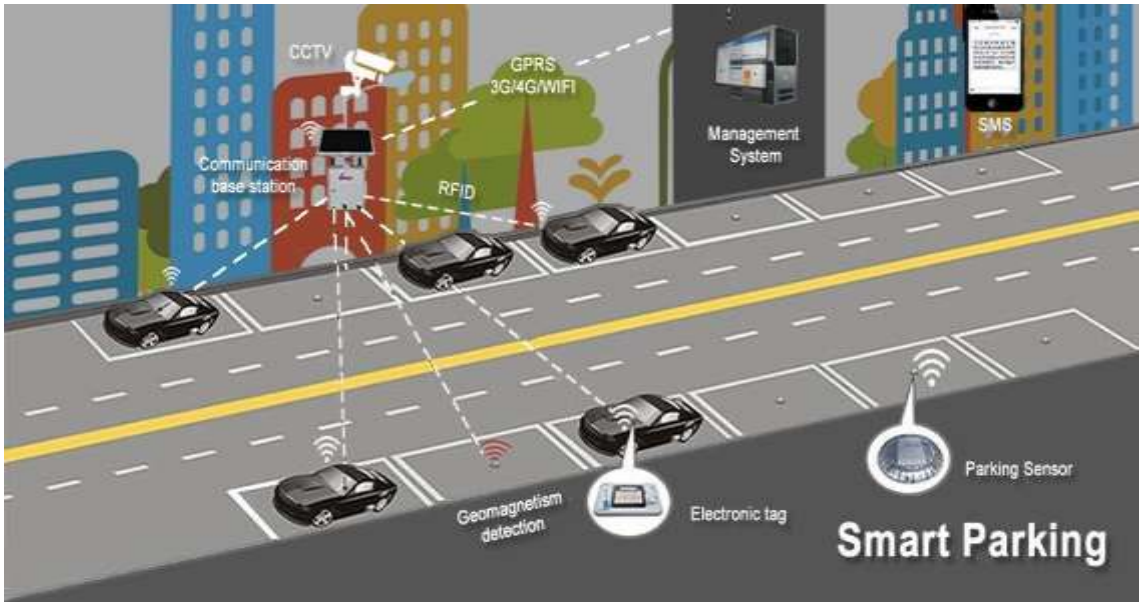

Figure 2.3 IoT-Integrated Smart Parking System

- European countries, especially Germany, are at the forefront of integrating sensor technologies, IoT, and real-time data into parking systems. The most common model here is a sensor-based parking system accompanied by guidance displays that direct drivers to available parking spaces.
- System Components:
  + Ultrasonic/magnetic sensors installed at each parking space.
  + Wireless communication networks (LoRa, ZigBee, Wi-Fi).
  + Electronic guidance displays at entrances and within the parking facility.
  + Central management software integrated with a mobile application.
- Operating Principle: Sensors detect the presence or absence of vehicles and transmit data to a central system. The processing software updates the availability map and

displays information on LED boards or mobile applications, guiding users to the correct parking spot.

- Advantages:
  + Easy to deploy on a large scale or integrate with existing parking facilities.
  + Enhances user experience by reducing the time spent searching for parking spaces.
  + Increases parking utilization efficiency by up to 30%.
- Disadvantages:
  + Does not increase the number of parking spaces within the same area.
  + Requires maintenance of a stable wireless network.
  + Depends on sensor accuracy (which can be affected by interference).

2.3. Modular Mechanical Parking System Model in South Korea

Figure 2.4 Modular Mechanical Parking System

- South Korea widely applies modular parking systems—automated parking solutions that can be quickly assembled and disassembled—deployed in residential areas, shopping centers, and small urban districts.

- System Components:
  + Mechanical multi-level frame system (3–10 levels), assembled in a modular fashion.
  + Lifting, lowering, and horizontal sliding mechanisms.
  + License plate recognition cameras and safety sensors.
  + Central control unit integrated with an HMI (Human-Machine Interface).
- Operating Principle: The vehicle is placed into the ground-floor cabin, then the lifting, lowering, and horizontal sliding mechanisms transport the vehicle to an available parking spot on an upper level. The system operates according to a predefined logical sequence.
- Advantages:
  + Flexible in construction and expansion.
  + Fully automated operation.
  + Can be remotely controlled via an application.
- Disadvantages:
  + Installation costs are relatively high due to the complex mechanical structure.
  + Requires regular mechanical maintenance.
  + Noise and vibrations may affect nearby residential areas if not properly managed.

2.4. Artificial Intelligence-Integrated Parking System Model in the USA

- The United States stands out with smart parking solutions integrating artificial intelligence (AI), big data analytics, and cloud computing platforms to forecast, optimize, and coordinate parking demand on a large scale.
- System Components:
  + Intelligent surveillance camera systems.
  + Multi-layer sensor networks.
  + Data centers integrated with AI and machine learning.
  + User interface via smartphone or web applications.

- Operating Principle: AI analyzes traffic flow, user habits, and parking duration to predict demand. Data is processed on a cloud platform to manage parking lots, provide recommendations, and optimize vehicle ingress and egress flow.
- Advantages:
  + Management based on forecasting and real-time data.
  + Optimization of parking lot performance.
  + Integration with public transportation systems and urban authorities.
- Disadvantages:
  + Requires robust network infrastructure and computing power.
  + Security and privacy are major concerns.
  + Initial deployment costs are very high.

3. Overview in Vietnam

- In Vietnam, smart parking system models are still in the early stages and experimental phases. Ho Chi Minh City and Hanoi are the two pioneering localities conducting trials of some multi-level parking lots with semi-automated integration; however, the scale remains small, lacking synchronization and standardized design.
- Notable Projects:
  + Rotary tower parking system on Le Lai Street, District 1 (Ho Chi Minh City).
  + Modular multi-level parking lot in the central area of Hanoi.
  + iParking and MyParking applications supporting parking search and payment via mobile phones.
- Overall, the systems remain isolated, lacking integration between hardware and software, and have yet to strongly apply IoT and AI technologies as seen in developed countries.

4. Comprehensive Evaluation

Model	Key Advantages	Main Disadvantages
Rotary Tower (Japan)	Space-saving, automated, fast operation	High cost, vehicle size limits, requires moving all vehicles simultaneously
IoT-based (Germany)	Quick deployment, improved user experience	Does not increase parking capacity, requires stable network
Modular (Korea)	Flexible, easy assembly/disassembly, scalable	Complex mechanics, high maintenance cost
AI-integrated (USA)	Intelligent management, accurate demand prediction	Expensive, requires strong digital infrastructure

5. Research Gaps

- From the above analysis, it is evident that Vietnam currently lacks a smart parking system model that:
  - + Has reasonable costs and is easy to localize.
  - + Suitable for small land plots, narrow roads, and densely populated urban areas.
  - + Can be automated without requiring overly advanced infrastructure.
  - + Easily integrates with remote control and payment applications.
- The project aims to develop a semi-automated multi-level smart parking model that integrates hardware and software, utilizing feasible technologies such as sensors, cameras, PLC or microcontrollers, along with remote control software. This solution seeks to reduce costs and enhance practical applicability in medium and small-sized cities in Vietnam.

6. Initial parameters assumed

The initial assumptions are based on the parameters of the Autostadt car tower model in Wolfsburg, Germany:

Table 1. Initial parameters assumed

Criteria	Autostadt Tower (Real)	Your Model (Scaled)	Scale Ratio & Assessment
Vehicle weight	~1,300–2,000 kg (real car)	0.1 kg	Reduced by ~1/15,000 – appropriate for small model
Number of levels	16 levels (~48 meters high)	3 levels (~0.6 m total height)	Height ratio ~1/80 – suitable for scaled-down version
Height per level	3 meters	0.2 meters	3 m → 0.2 m (approx. 1:15 scale)
Number of parking slots	~400 cars per tower (~800 total in 2 towers)	12 slots	Scaled down appropriately while maintaining full function
Time to store/retrieve 1 car	~1 minute (60 seconds)	15–30 seconds	Reasonable for small load and hobby-grade motors
Lifting system	Robotic lift with CNC control	Motor controlled by Arduino	Same principle: lift–rotate–park
RFID/vehicle tracking	Yes, used in customer handover and car tracking	100% RFID reading accuracy	Equivalent functionality
License plate recognition	Not used (vehicles are new, pre-registered)	100% recognition using OpenCV	Additional AI feature, enhances functionality

CHAPTER 3: CALCULATION AND DESIGN OF MECHANICAL SYSTEM

1. System Structure

1.1. Main Mechanical Frame

- Function: Serves as the load-bearing frame, forming the overall structure of the entire system.
- Material: Aluminum profiles (type 20x20 or 30x30), connected using screws, corner brackets, and support struts.
- Proposed dimensions: 600 mm × 600 mm (base area) and 1200 mm (height).
- Number of levels: 3 levels (evenly divided by height), with 4 vehicle slots per level arranged symmetrically around the center.
- Requirements: Must ensure rigidity, stability, and ease of assembly/disassembly for maintenance purposes.

1.2. Lifting Mechanism

- Function: Raises and lowers the vehicle tray vertically to the required floor level.
- Structure: Consists of a lead screw, sliding nut, coupling joint, linear guide rails, and a drive motor.
- Operation: When the motor rotates the lead screw, the nut (attached to the vehicle tray) moves up or down accordingly.
- Guidance: Movement is guided by two parallel square linear rails to ensure smooth motion and minimize vibration.
- Limit sensors: Equipped with limit switches at the top and bottom floors to restrict the travel range.

1.3. Rotating Platform Mechanism

- Function: Controls the horizontal rotation of the entire tray and leadscrew system to align the vehicle tray with the designated parking or retrieval position.
- Motor: Uses a stepper motor (NEMA 17 or NEMA 23, depending on load and model size), controlled via A4988 or TMC2209 driver.

- Pulley: Attached to both the motor shaft and the main rotating shaft (which supports the entire leadscrew and tray system).
- Timing Belt (GT2 or equivalent): Transfers motion from the motor to the rotating frame.
- Rotating Shaft: The central axis fixed to the leadscrew system, linear rails, and all tray levels; rotates the entire lifting assembly around the Z-axis.
- Bearing Block: Supports the main shaft and ensures smooth rotation.
- Position Limiting: Uses limit switches or angular sensors (optical or Hall effect) to detect fixed rotational positions corresponding to parking spots.

1.4. Vehicle Tray

- Function: Serves as the platform to hold the vehicle model during system operation.
- Structure: Made from thick (~5 mm) acrylic or plastic sheet, shaped to securely hold the vehicle's wheels in place.
- Mounting: Directly attached to the sliding joint to ensure stability and prevent slipping during rotation.

1.5. License Plate Recognition System (Camera)

- Function: Recognize vehicle license plates through image processing (OpenCV).
- Equipment: A fixed webcam installed at the vehicle entry/exit position, with lighting directed straight onto the license plate for clear recognition.
- Installation location: Mounted near the vehicle loading/unloading platform, positioned parallel to the license plate’s line of sight.

2. Design Methodology

2.1. Option 1: Traditional Parking Lot (Flat Ground Type)

Figure 3.1 Traditional Parking Lot

- Description:
  - + Vehicles are parked directly on the ground or on concrete/asphalt slabs with marked parking spaces.
  - + Drivers manually find parking spots and move their vehicles in and out.
- Advantages:
  - + Simple and easy to construct without requiring mechanical systems or automated control.
  - + Low initial investment cost.
  - + Suitable for large spaces with low vehicle density.
- Disadvantages:
  - + Requires a very large area for each parking spot plus access lanes.
  - + Difficult to manage a large number of vehicles; not optimal for crowded urban areas.
  - + Not automated, time-consuming, and dependent on the driver.
- Conclusion: Not suitable for the goals of space optimization and automation. Should only be applied in large open areas or temporary use.

2.2. Option 2: Rotary Tower Parking System

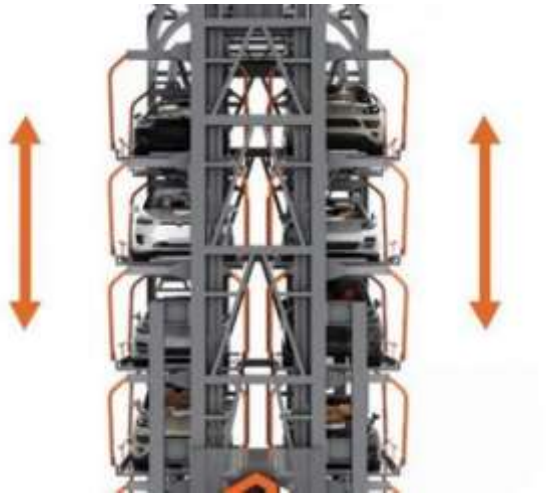

Figure 3.2 Rotary Tower Parking System

- Description:
  - + Cars are placed on rotating trays similar to a "carousel."
  - + When a car needs to be retrieved, the entire carousel rotates to bring the car to the entry/exit position.
  - + Similar to a vending machine mechanism.
- Advantages:
  - + Optimizes vertical space usage.
  - + Simple structure to assemble, with fewer complex lifting mechanisms.
  - + Compact design, easy to use for small-scale models.
- Disadvantages:
  - + Safety concerns: To retrieve one car, the whole system rotates, causing all other cars to move, increasing the risk of collision or cars falling.
  - + Slow operation: Retrieving a car requires rotating through multiple trays, which is time-inefficient.
  - + Limited scalability: The number of parking spots is constrained by the carousel size and motor load capacity.
  - + Difficult maintenance: If one position malfunctions, the entire system stops functioning.

- Conclusion: Not suitable for systems requiring high safety and fast processing speed. Recommended only for mini systems or display models.

2.3. Option 3: Fully Automated Vertical Tower Parking

Figure 3.3 Multi-level Parking System Combined with Lifting Mechanism

- Description:
  + Each parking spot is located on a level within a vertical tower structure.
  + A lifting mechanism moves vertically to transport the vehicle to the desired level.
  + The system combines with a rotating tray mechanism to position the vehicle into the correct slot.
  + Cameras assist in license plate recognition and control via software.
- Advantages:
  + Safety: Only one vehicle moves at a time, so other cars are not affected.
  + Space-saving: Optimizes vertical space, suitable for small areas.
  + High automation: Fully controlled by software, no manual operation needed.
  + Good scalability: Easy to upgrade to more levels, trays, or to install multiple stations in parallel.

+ Moderate cost: Simple structure, uses fewer motors, and easy to maintain.

– Disadvantages:

+ Requires careful calculation of the lifting mechanism and level positioning.
+ Demands close coordination between software, mechanical, and electronic components.

– Conclusion: Meets all requirements for safety, compactness, reasonable cost, and automation capability. Selected as the main option for system development.

3. Choose an idea

Criteria	Traditional Parking Lot	Rotary Tower Parking System	Fully Automated Vertical Tower Parking
Structure Type	4	6	9
Land Footprint	2	6	10
Vehicle Capacity	4	7	10
Construction Cost	9	6	4
Level of Automation	1	5	10
Vehicle Retrieval Time	9	7	6
Manpower Requirement	3	8	10
Operational Cost	9	6	5
Best Application Scenarios	6	7	9
Sum	47 / 90	58 / 90	73 / 90

During the evaluation process, we proposed three design options. After thorough analysis, we selected the third option – the Fully-automated Vertical Tower Parking System – for the following reasons:

– First, it saves ground space thanks to its vertical structure. This design is ideal for crowded urban areas where land is limited, allowing more vehicles to be stored within a smaller footprint.

– Second, its operation is straightforward, using only lift, slide, and rotate mechanisms. These basic movements are easier to construct, control, and maintain, making the system efficient and reliable.

- Third, it operates fully automatically, without human intervention. This enhances safety, reduces errors, and improves user convenience through seamless automation.
- Fourth, the system is scalable – more floors or slots can be added as needed. This flexibility makes it suitable not only for demonstration models but also for real-world deployment.

4. Calculation and Motor Selection

Table 2. Detailed mass

Component Group	Part Description	Quantity	Weight (kg)
Push Mechanism	Push block (main part)	1	0.08
	LM10UU linear bearings	2	0.038
	LM10UU bearing holders	2	0.02
Lift Mechanism (Lead Screw)	Weight of the push mechanism	–	0.2
	SK10 shaft supports	4	0.055
	Shaft Ø10mm, length 15cm	2	0.093
	KFL08 bearing block	1	0.12
	GT2 pulley 20T, bore 8mm	1	0.015
	GT2 pulley 20T, bore 6mm	1	0.012
	Base plate for push mechanism	1	0.11
	Shaft Ø8mm, length 32–35cm	1	0.01285
	12V DC motor	1	0.25
	SCS10UU linear bearing blocks	4	0.017
	3034 aluminum square bar	1	0.07
	Link plate (SCS10UU + push mechanism)	1	0.08
	Brass nut (connects 3034 bar & link plate)	1	0.02
360° Rotation Mechanism	Weight of lead screw lift mechanism	–	1.4
	Shaft Ø10mm, length 15cm	2	0.31
	T8 lead screw, length 400mm	1	0.158
	KP08 bearing	1	0.012
	SK10 shaft supports	5	0.055
	KFL000 bearing blocks	3	0.075
	Shaft Ø10mm, length 41mm	1	0.025
	Main box (housing)	1	0.12
	Box cover	1	0.03
	Aluminum corner brackets	16	0.025

	Aluminum profile, length 140mm	4	0.084
	Shaft Ø10mm, length 75mm	1	0.0465
	Coupling (stepper + lead screw)	1	0.015
	Upper acrylic plate (360° tray)	1	0.1
	Lower acrylic plate	1	0.14
	Stepper motor	1	0.29

4.1. Stepper Motor for Vertical Lift

- Input Parameters:
  - Car weight: 0.1 kg
  - Screw mechanism: 1.4 kg
  - Tray weight: 0.2 kg
  - Total mass (m): 1.7 kg
  - Lift height (H): ~1 m
  - Time to lift: ≤ 5 seconds
  - Screw type: Trapezoidal screw, lead pitch P=8mm
  - Efficiency (η): 0.35
- Required Lifting Force:

$$F = m \times g = 1.7 \times 9.81 \approx 16.677\ N$$

- Required Torque:

Formula:

$$T = \frac{F \times P}{2\pi \times \eta}\ [1]$$

Where:

- F = 16.677 N
- P = 8 mm = 0.008 m
- η = 0.35

Substituting into the formula:

$$T = \frac{16.677 \times 0.008}{2\pi \times 0.35} \approx 0.06\ N.m$$

- Adding safety factor k = 4:

  To account for sudden loads, friction, and impact forces, a safety factor is applied k = 4:

$$T_{safe} = 0.06 \times 4 = 0.24\ N.m$$

- Selected Motor:
  + Type: Stepper Motor NEMA 17
  + Nominal torque: 0.45 – 0.5 Nm
  + Speed: 200–400 RPM
  + Recommended driver: TB6600

  ⇒ Meets torque requirement with high positioning accuracy.

4.2. Stepper Motor for Tower Rotation

- Input Parameters:
  + Rotating mass (including tray + screw): 4 kg
  + Rotation radius (r): 0.2 m
  + Rotation angle per operation: 36° = 0.628 rad
  + Rotation time: ~1 second
  + Transmission type: GT2 timing belt with 20-tooth pulley
- Moment of Inertia:

$$J = m.r^2 = 4 \times 0.2^2 = 0.16\ kg.m^2\ [2]$$

- Angular Acceleration:
  + Rotation angle: $36^\circ = 0.628\ rad$
  + Acceleration time: t = 1s

$$\alpha = \frac{2\theta}{t^2} = \frac{2 \times 0.628}{1^2} = 1.256\ rad/s^2\ [3]$$

- Required Torque:

$$T = J \times \alpha = 0.16 \times 1.256 = 0.2\ N.m\ [3]$$

- With safety factor k =2:

$$T_{safe} = 0.2 \times 2 = 0.4\ N.m$$

- Selected Motor:
  + Type: Stepper Motor NEMA 17
  + Nominal torque: 0.45 – 0.5 Nm
  + Speed: 200–400 RPM
  + Recommended driver: TB6600

  ⇒ Meets torque requirement with high positioning accuracy.

4.3. DC Motor for Tray Push/Pull Mechanism

- Input Parameters:
  + (Car + tray) mass: 0.5 kg
  + Tray travel distance: 0.1 m
  + Desired time: ~1 second
  + Friction coefficient (f): 0.2 (linear rail or wheel guide assumed)
- Required Force:

$$F = f \times m \times g = 0.2 \times 0.5 \times 9.81 = 0.981\ N$$

- Required Power:

$$v = \frac{s}{t} = \frac{0.1}{1} = 0.1\ m/s$$

$$P = F \times v = 0.981 \times 0.1 = 0.0981\ W$$

- With safety factor k = 5:

$$P_{safe} = 0.0981 \times 5 \approx 0.5W$$

- Selected Motor:
  + Type: DC Motor 12V
  + Power: 3 – 5 W
  + Examples: JGA25-370 or RS-360
  + Transmission: Direct or rack-and-pinion style push

5. Webcam for image processing

5.1. Webcam Specifications

- Resolution: 1920×1080 pixels (Full HD)
- Frame Rate: 30 frames per second (fps)
- Sensor Type: CMOS
- Lens: Fixed focal length, e.g., 3.6mm
- Infrared (IR) Support: No

5.2. Pixel Density (Pixels Per Foot - PPF)

- Pixel density is crucial for LPR. A minimum of 40 PPF is generally required for reliable license plate recognition.
- Calculation:
  + Horizontal Resolution: 1920 pixels
  + Field of View (FOV): Assuming 40 feet

$$PPF = \frac{1920}{40} = 48PPF$$

- Since 48 PPF > 40 PPF, the webcam meets the minimum requirement for LPR at a 40-foot distance.

5.3. Lighting Conditions

- The webcam does not support infrared (IR), which makes it less effective in low-light or nighttime conditions.
- Recommended illumination level: at least 50 lux for accurate license plate detection.
- If used during daylight hours or in areas with supplemental lighting, the system can perform well.
- If installed at an entrance or gate for nighttime use, it is necessary to add white LED lighting or replace with a camera that supports IR (infrared night vision).

5.4. Lens Focal Length & Mounting Angle

- A 3.6mm lens provides a wide field of view, which is ideal for close-range capture, but reduces image detail at longer distances.
- Recommended camera mounting angles for optimal LPR (License Plate Recognition):
  + Vertical tilt angle: ±30°
  + Horizontal skew angle: ±20°
  + License plate skew: Should not exceed ±10° in any direction
- The camera should be **mounted directly facing the license plate**, and the angle should be **as perpendicular as possible** for the best recognition accuracy.

5.5. Compatible License Plate Recognition (LPR) Software

- To perform license plate recognition using a webcam, you will need LPR software capable of processing video frames in real-time. Below are some popular options:
  + OpenALPR (now known as PlateRecognizer): An open-source library that is easy to integrate with Python and supports license plate detection across multiple regions.
  + LPRNet or YOLO+CRNN: Deep learning-based solutions that use convolutional and recurrent neural networks for high-accuracy license plate recognition from images or video.
  + OpenCV combined with Tesseract OCR: A more manual but flexible approach that involves detecting the license plate frame using OpenCV, then applying optical character recognition (OCR) with Tesseract to extract the plate number.
- **All of these options can work effectively with webcam input**, provided the image resolution and clarity are sufficient.

5.6. Conclusion

- With a Full HD resolution and 30 fps frame rate, your webcam is suitable for license plate recognition, provided that:
  + The distance to the license plate is less than 12 meters

+ There is adequate lighting, or external lights are added
+ The camera is mounted at an appropriate angle, ideally perpendicular to the license plate

– For **higher recognition accuracy**, **faster processing**, and **nighttime operation**, it is recommended to:
+ Upgrade to a camera that supports infrared (IR) or night vision
+ Use a lens with a longer focal length if the camera needs to capture plates from distances greater than 3 meters
+ Add external 12V white LED lighting to illuminate the license plate at night

6. Mechanical design

6.1. Active Roller Transfer Mechanism

– In tower-type smart parking systems, transferring vehicles between the main elevator and the parking slots requires an auxiliary mechanism capable of generating horizontal linear motion, while remaining compact, flexible, and precise. The mechanism you propose is an intelligent design based on friction drive (vehicle-pushing rollers), utilizing a DC electric motor combined with a rotating shaft and a guided sliding platform.

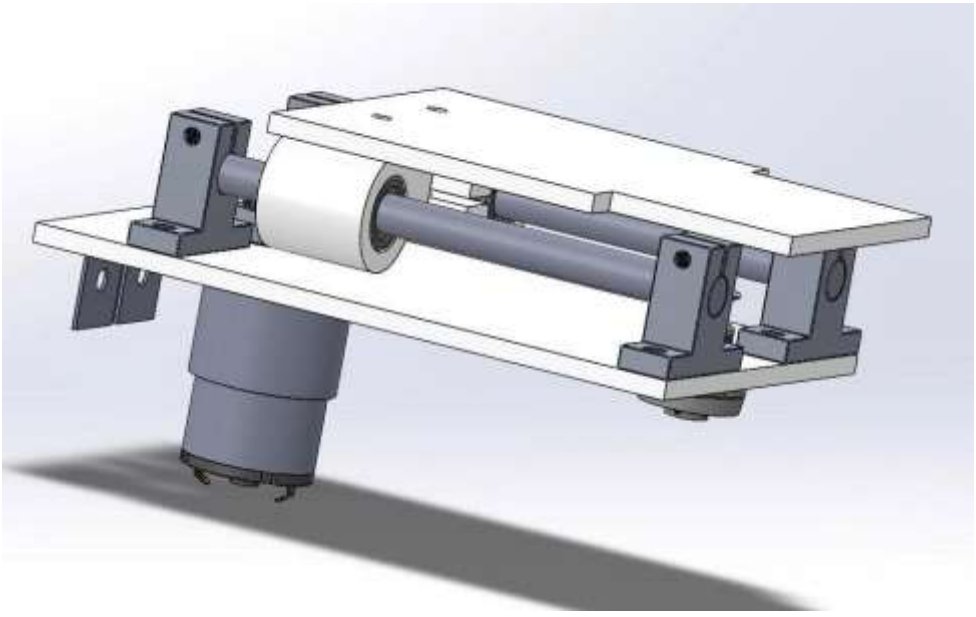

Figure 3.4 Active Roller Transfer Mechanism

- Main Components:
  + DC Motor
    - Position: Mounted at the bottom of the mechanism.
    - Function: Generates torque to rotate the drive shaft.
    - Type: A geared DC motor is used to deliver high torque at low speed, ensuring accurate and forceful movement.
  + Rotating Drive Shaft
    - Material: Typically made of steel or a strong alloy.
    - Function: Transmits torque from the motor to the roller.
    - Structure: A continuous shaft supported at both ends with bearing housings, allowing smooth rotation.
  + Friction Roller
    - Position: Mounted on the drive shaft and in direct contact with the vehicle's tire.
    - Function: When the roller spins, it generates tangential frictional force to push or pull the vehicle.
    - Material: Rubber-coated exterior for grip, with a metal core (usually aluminum or steel).
    - Highlight: Centrally placed to ensure balanced force application on the tire.
  + Mounting Frame and Plates
    - Function: Holds all components together and transmits force efficiently.
    - Structure: CNC-machined metal plates (usually aluminum or steel) with precise mounting holes for screws and bolts.
  + Sliding Base and Linear Guide Rails
    - Position: Located underneath the mechanism, connecting it to the elevator platform.
    - Function: Allows the entire pushing unit to slide horizontally toward or away from the car.

- Structure: Uses linear guide rails and sliding bearings for smooth, accurate movement.

– Working Principle:

+ When the elevator reaches the designated parking level: The pushing mechanism slides horizontally into position.
+ The DC motor activates, rotating the drive shaft and thus the roller.
+ The roller contacts the car's wheel, generating friction to push or pull the car into or out of the parking slot.
+ After completion, the mechanism retracts to its original position, ready for the next operation.

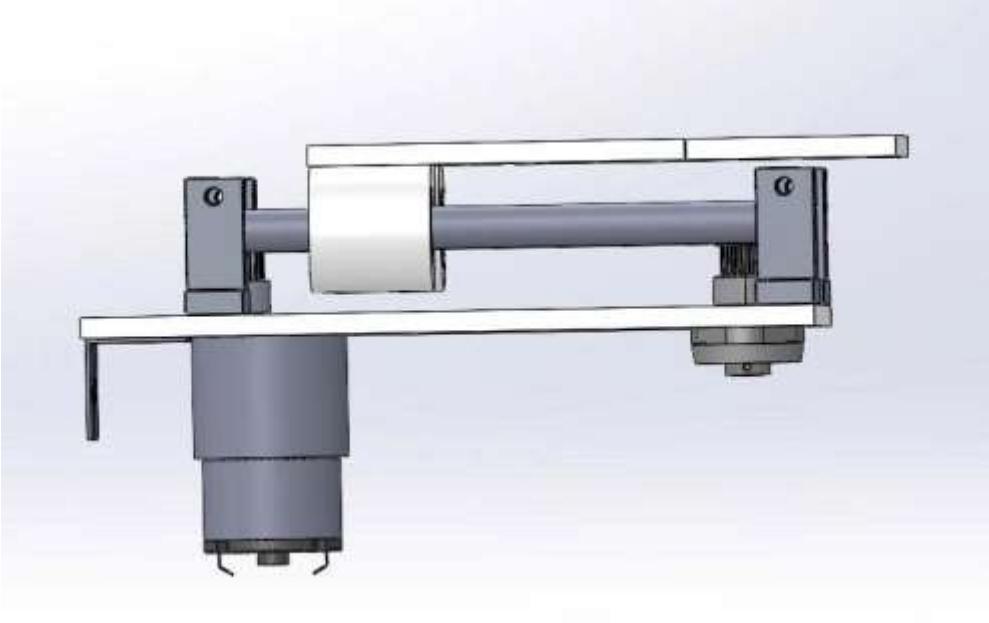

Figure 3.5 Active Roller Transfer Mechanism

– Advantages of the Design:

+ Compact size: Fits easily within the elevator structure.
+ Bidirectional: Roller rotation can be reversed to either push or pull the vehicle.
+ Precision and stability: Thanks to linear guides and geared motor.
+ Easy maintenance: Simple mechanical parts are easy to replace or repair.

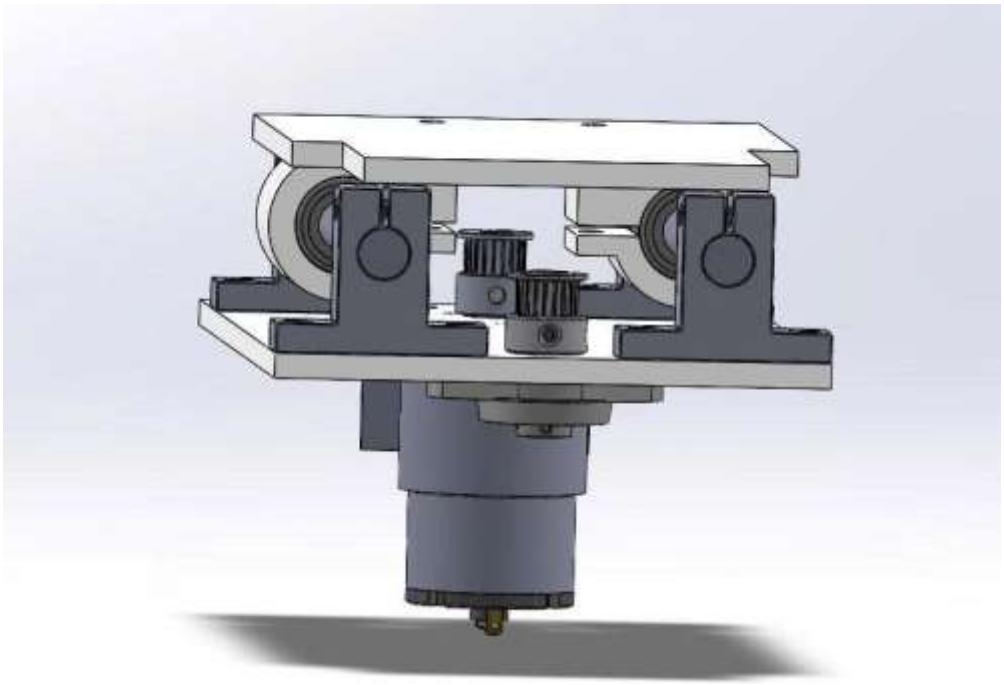

Figure 3.6 Active Roller Transfer Mechanism

- Applications and Potential Improvements:
  + Application: Ideal for smart parking systems with vertical lifts or conveyor-based systems.
  + Potential Enhancements:
    - Add position sensors for accurate control.
    - Use encoders for speed and displacement feedback.
    - Integrate shock absorbers or spring-loaded rollers for gentler contact with vehicle tires.

6.2. Webcam Mounting Bracket for License Plate Recognition

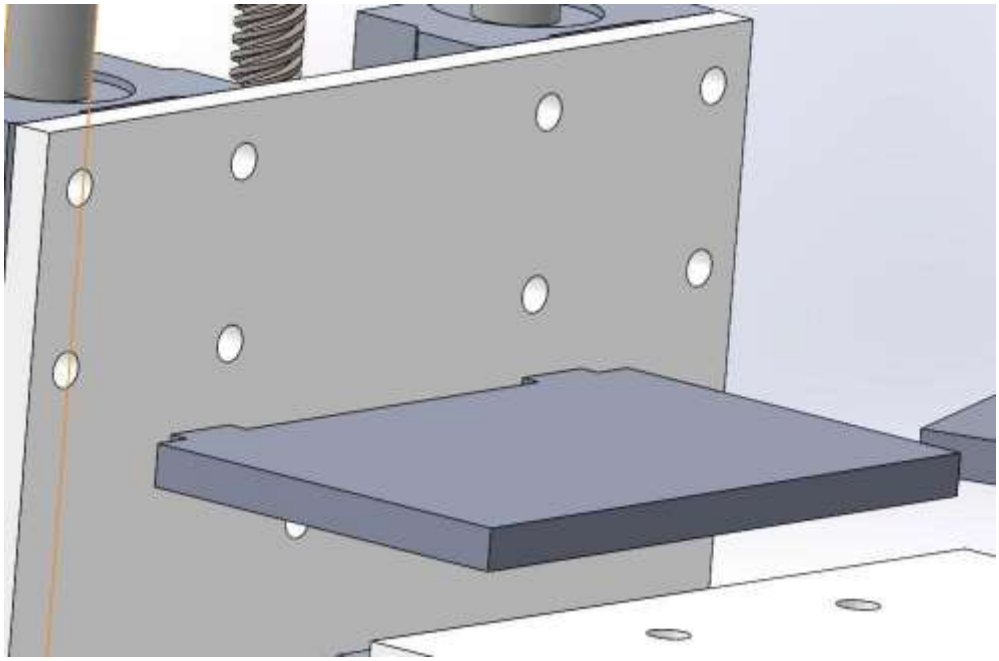

Figure 3.7 Webcam Mounting Bracket

- Function and Placement:
  + This component is a webcam mounting bracket, designed specifically for holding a camera used in automatic license plate recognition (ALPR). It is strategically positioned directly in front of the vehicle's license plate, ensuring clear and direct visibility for accurate image capture.
  + Location: Installed on the main structure of the vehicle retrieval mechanism.
  + Orientation: The bracket holds the camera facing forward, perpendicular to the vehicle's orientation, to get a frontal view of the license plate.
- Design and Construction:
  + Bracket Shape
    - The bracket features a flat horizontal platform, which acts as a base for attaching the webcam or camera housing.
    - A small notch or recess at the back of the bracket provides cable management or camera alignment support.

- The shape is rectangular with optimized dimensions for compact mounting within the limited space of the elevator carriage.

+ Material
  - Likely made from aluminum or steel sheet metal, providing:
  - Sufficient rigidity to prevent camera vibration.
  - Resistance to deformation due to temperature or minor impact.
  - Lightweight construction to avoid overloading the lifting mechanism.
+ Mounting Interface
  - The bracket is fixed to the vertical face of the main structure using screws or bolts.
  - Multiple circular holes are visible on the mounting surface, allowing for:
  - Flexible height adjustment.
  - Compatibility with different camera types or models.
  - Strong and stable attachment using threaded fasteners.

– Integration with Smart Parking System:
  + Purpose: Enables the system to automatically detect and identify license plates during vehicle retrieval or parking.
  + Timing: The webcam activates when the vehicle is fully loaded on the platform or when it arrives at the central lift.
  + Signal Routing: The bracket design leaves space underneath for camera power and data cables, allowing neat and safe routing to the main control unit.
– Benefits of the Bracket Design:
  + Compact and unobtrusive: Does not interfere with vehicle movement.
  + Precision alignment: Ensures the webcam stays correctly positioned at all times.
  + Ease of maintenance: Camera can be quickly removed or replaced by unscrewing a few fasteners.
  + Reliable data capture: Thanks to stable mounting and vibration reduction.

6.3. Lifting mechanism design

– General Introduction:

The analyzed structure is a vertical linear lifting mechanism (Z-axis) that utilizes a ball screw as its main transmission component. The system is driven by a single stepper motor located at the top of the structure. This mechanism is commonly applied in automated storage systems, smart parking solutions, robotic manipulators, or automation equipment that requires vertical motion.

Its core structure comprises a rigid guide frame, a ball screw shaft paired with a recirculating ball nut, a moving platform (mounting plate) that travels vertically, and a pair of linear guide rods. The design prioritizes accuracy, structural stability, simplicity of control, and ease of maintenance.

Figure 3.8 Lifting mechanism design

- Detailed Component Description:
  + Upper Housing Unit (Head Section)
    - Located at the topmost position of the mechanism, this unit serves as the foundation for mounting and protecting the motor and transmission components.
    - Structure: The housing is a closed rectangular box, potentially made of extruded aluminum, cast aluminum, or steel plate. Inside, a stepper motor is mounted, which connects directly or via a flexible coupling to the ball screw shaft. On both sides of the housing are circular openings, likely intended for assembly access or passive ventilation — no active cooling fan is present.
    - Function: The housing provides a stable mount and protective enclosure for the stepper motor. It transmits the motor's rotational motion to the ball screw, thereby generating linear motion in the system. Additionally, this unit may also serve as a mount for control circuitry, limit switches, or cable routing components.
  + Ball Screw Shaft
    - This is the primary component responsible for generating linear vertical movement within the system.
    - Structure: The ball screw features a helical groove along which a recirculating ball nut travels. The ends of the screw shaft are supported by bearings or bushings to ensure alignment and smooth rotation.
    - Function: When the screw is rotated by the stepper motor, the ball nut translates along its axis. This nut is mechanically connected to the moving platform, enabling it to move vertically up or down depending on the direction of rotation.
  + Pair of Parallel Linear Guide Rods
    - These are the linear guiding elements that ensure the moving platform travels smoothly along the Z-axis without tilt or misalignment.

- Structure: Two polished cylindrical rods are arranged parallel to the ball screw. The moving platform incorporates linear bushings or slide bearings that travel along these guide rods.
- Function: They maintain the linearity and stability of the platform's motion, preventing lateral shifts or angular deviations. They also help bear off-center loads or moments caused by external forces acting on the end effector.

+ Moving Platform (Mounting Plate)
  - This intermediate structure carries the payload or end effector and moves along the Z-axis.
  - Structure: A flat metal plate rigidly fixed to the ball nut. The plate features threaded holes, mounting slots, or modular fixtures for attaching tools or devices. Linear bushings are integrated on both sides for smooth motion along the guide rods.
  - Function: It transfers the linear motion generated by the ball screw to the mounted equipment (e.g., grippers, lifting modules). It also acts as a modular interface for integrating various automation devices or sensors.
+ Bottom Frame (Base Support)
  - This is the fixed structural base at the bottom of the entire mechanism.
  - Structure: A precision-machined rectangular block, typically made of aluminum or steel. It houses the lower-end supports for both the ball screw and guide rods using bearings or bushings.
  - Function: It provides structural anchoring and accurate positioning for the screw shaft and guide rods. No motor or actuator is housed in this unit. It also serves as the base to securely attach the entire lifting system to the main machine frame or ground structure.

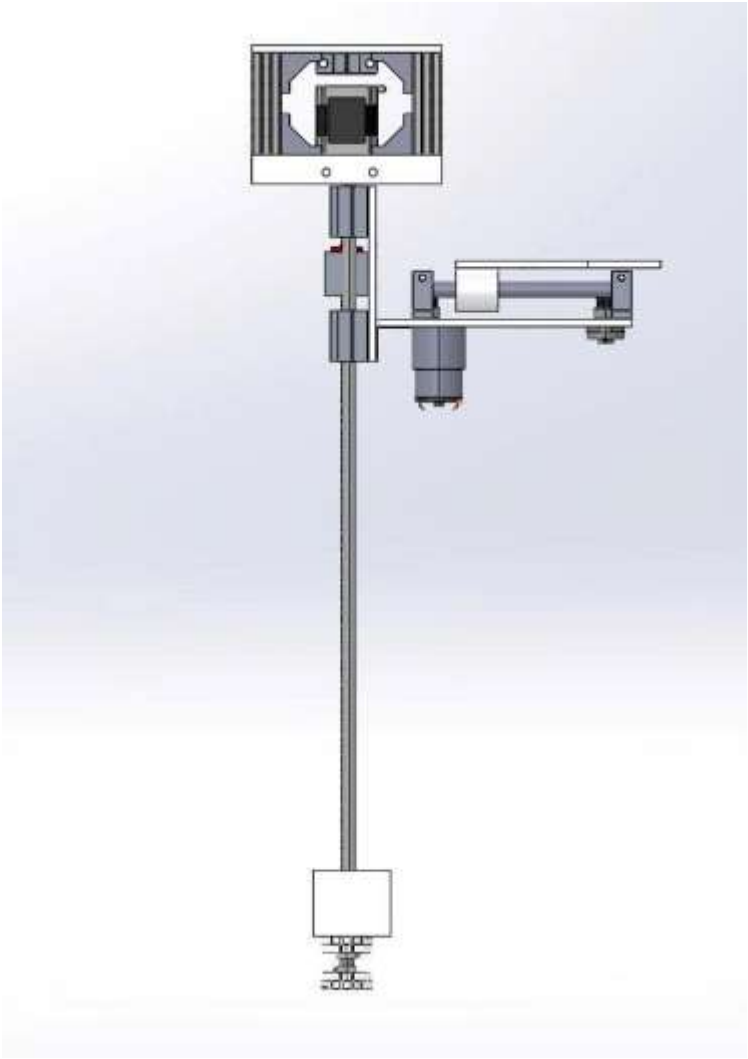

Figure 3.9 Lifting mechanism design

- Working Principle:

  The system operates by converting rotary motion from the stepper motor into linear motion through the ball screw mechanism. As the motor rotates the screw, the ball nut translates vertically along the screw's axis. The moving platform, attached to the nut, follows this motion, thereby raising or lowering any attached payload.

  The twin guide rods ensure that the platform maintains a straight, vibration-free vertical path. This design allows for high-precision, repeatable movements and can endure long operational hours with minimal wear and maintenance.

- Advantages of the Design:
  + Uses only a single motor, simplifying control and reducing cost.
  + Ball screw mechanism provides high efficiency, low friction, and excellent positional accuracy.
  + Dual guide rods maintain stable vertical motion and prevent twisting or wobble.
  + The drive system is isolated from dust and vibration, improving durability.
  + The overall structure is compact, modular, and easy to maintain or integrate with other automation subsystems.

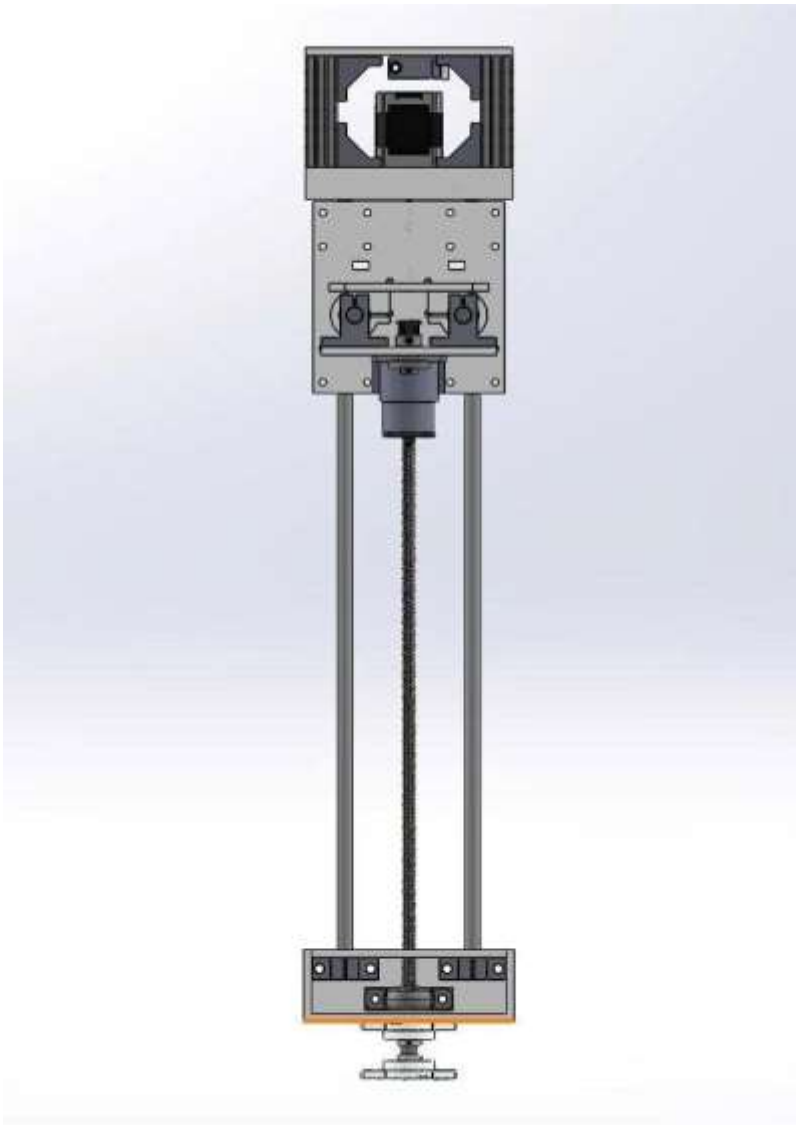

Figure 3.10 Lifting mechanism design

6.4. Rotating mechanism design

- In the context of rapidly growing urban areas, the availability of parking space is becoming increasingly limited. As a result, the development and implementation of compact, intelligent, and automated parking systems is an essential trend. One effective solution is the circular rotary parking system, where vehicles are parked around a central rotating structure. In such systems, a precise rotating mechanism is required to position the lifting carriage at various parking slots arranged along the circumference. To accomplish this accurately and reliably, we have designed a rotation mechanism driven by a stepper motor through a timing belt transmission.

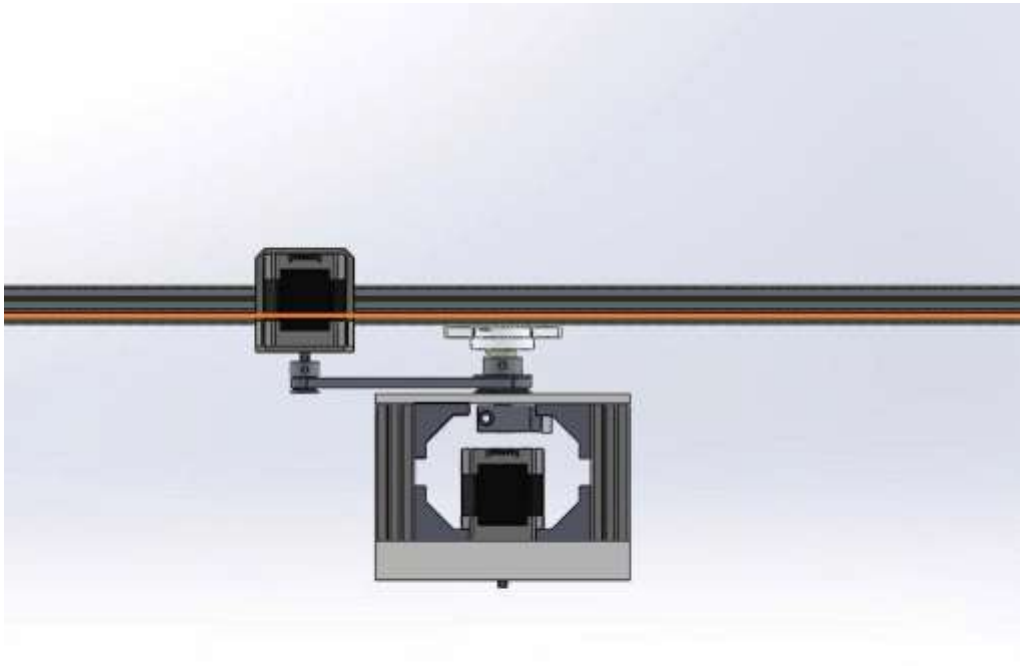

Figure 3.11 Rotating mechanism design

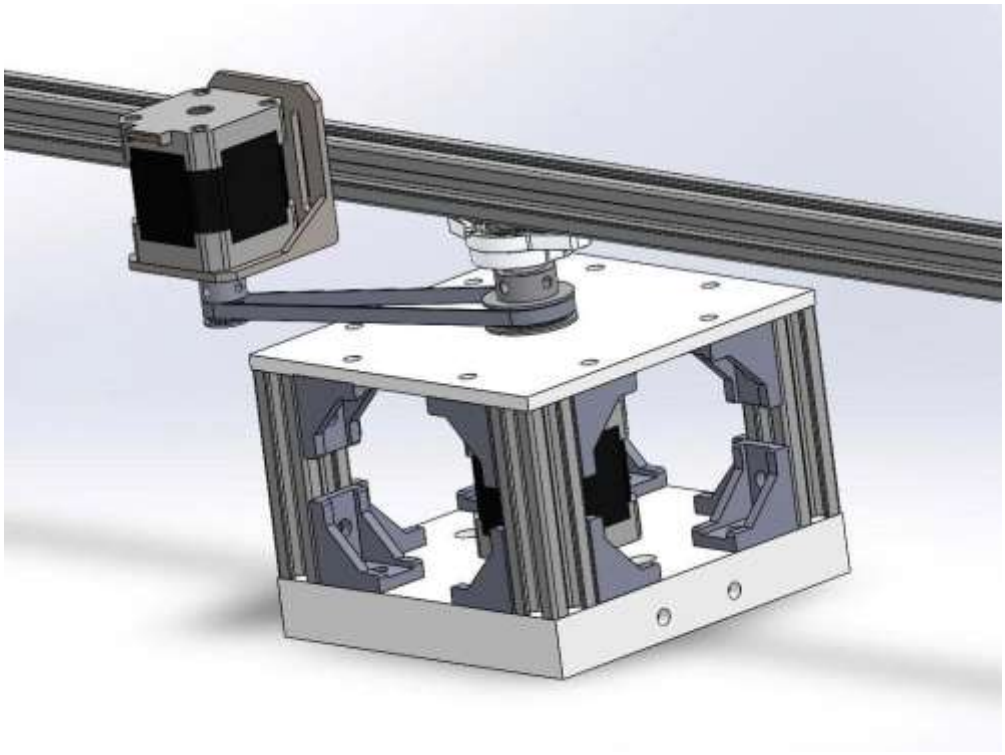

Figure 3.12 Rotating mechanism design

- Role and Location of the Rotating Mechanism:
  + The rotating mechanism is located at the center of the entire system, acting as the main axis that supports and rotates the entire structure including the leadscrew, the lifting carriage, and the linear guide rails. As the mechanism rotates, it positions the carriage precisely at each designated parking slot around the system.
  + This mechanism:
    - Precisely orients the carriage to the correct parking position.
    - Transmits rotational motion from the stepper motor to the central shaft.
    - Supports the entire lifting structure and ensures smooth, stable, and safe rotation.
  + Notably, this rotation is independent of the vertical movement of the carriage (lifting/lowering), enabling two separate yet coordinated motions.

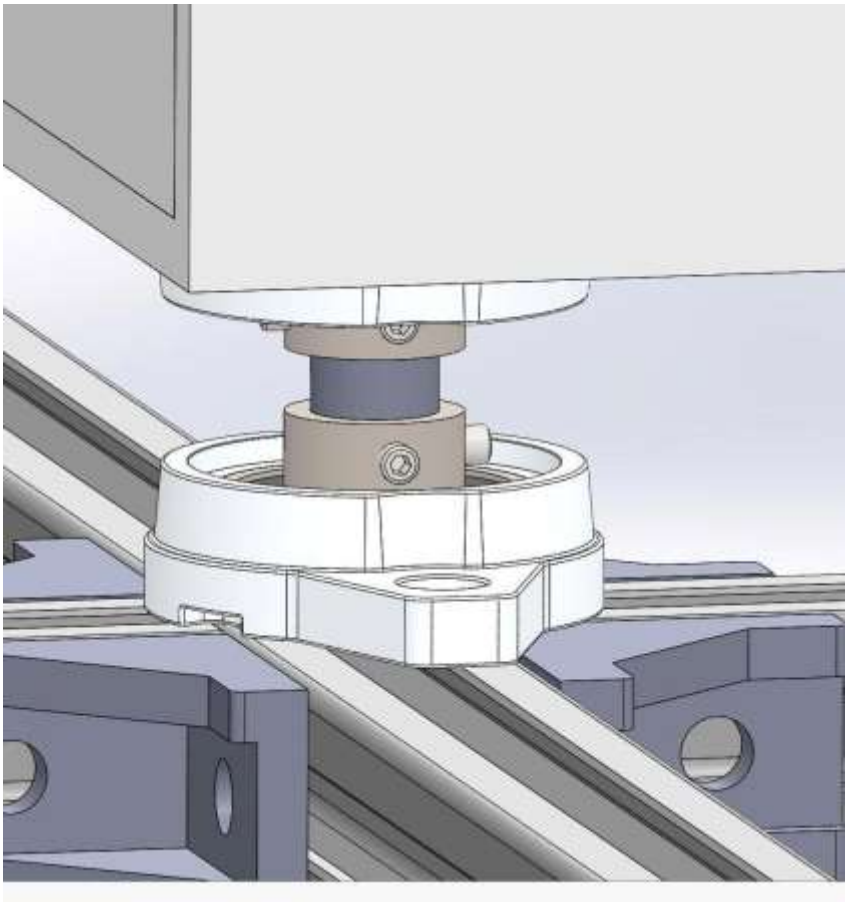

Figure 3.13 Rotating mechanism design

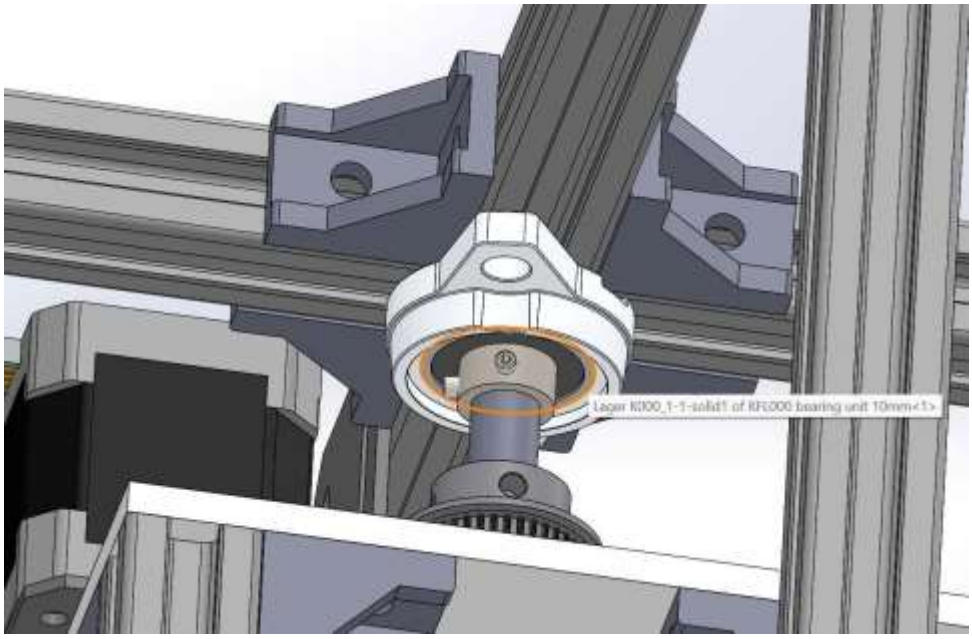

Figure 3.14 Rotating mechanism design

- General Working Principle:
  + A stepper motor is mounted horizontally onto the frame near the base of the system.
  + The motor rotates a small pulley, which transmits motion via a GT2 timing belt to a larger pulley fixed on the central vertical shaft.
  + As the central shaft rotates, the entire leadscrew, carriage, and guide rail assembly rotates with it around the vertical axis.
  + Once the carriage reaches the desired angular position (e.g., parking spot No. 3), the stepper motor halts, and then another motor initiates the vertical movement via the leadscrew to lift or lower the vehicle.
  + Thanks to the stepper motor, angular positioning is highly accurate even without feedback sensors (open-loop control). The number of steps or microsteps is pre-calculated based on the total number of parking spots required.
- Detailed Mechanical Structure:
  + Central Rotating Shaft
    - This is the core mechanical element of the system. It is a vertically mounted steel or stainless steel shaft, precisely machined to transmit torque and support the entire load.
    - Functions:
    - Transmits torque from the pulley to the entire lifting assembly.
    - Supports the total load including the leadscrew, guide rails, carriage, and the vehicle weight.
    - Recommended diameter: 10mm–15mm, depending on the load.
    - Both ends of the shaft are supported by bearing housings (KFL000) to ensure smooth rotation and alignment.
  + Bearing Housings (KFL000)
    - These are self-aligning flange bearings fixed onto the aluminum frame (30x30 or 40x40 profiles).

- The bearings allow free rotation while preventing lateral movement of the shaft.
- They are mounted with bolts and allow for easy installation and maintenance.
- Two bearings: one on the top, one at the bottom, to ensure stability and minimize radial run-out.

+ Stepper Motor
  - Type: NEMA 17 or NEMA 23, commonly used, easy to control, and cost-effective.
  - Mounted horizontally using L-shaped brackets or custom plates.
  - Torque rating: 1.2 Nm to 3 Nm, depending on the system's mechanical load and number of parking levels.
  - Controlled via drivers such as A4988, TMC2209, or DM542 using pulse signals from a microcontroller (Arduino, ESP32, or PLC).
+ Timing Belt and Pulley Drive
  - The stepper motor rotates a small pulley, which drives a GT2 or HTD 3M timing belt connected to a larger pulley fixed on the central shaft.
  - Gear ratio: typically 1:3 to 1:5, enhancing torque and reducing rotation speed for smoother motion.
  - The toothed belt ensures zero slip, providing accurate and repeatable rotation steps.
  - This system is easy to assemble, low maintenance, and flexible in terms of design scalability.

– Leadscrew and Carriage – Rotating as a Unit:

+ Leadscrew (Ball Screw or ACME Rod)
  - A vertically mounted threaded rod used to lift or lower the carriage.
  - One end connects to the carriage through a leadscrew nut; the other end is supported or driven by a second motor.

- The entire leadscrew is rigidly fixed to the central shaft, so it rotates along with the shaft but moves independently in the vertical direction when driven.

+ Carriage (Lift Platform)
  - A sliding platform that moves vertically along two linear guide rails.
  - Guide rails (e.g., MGN12 linear rails or SBR round rods) are fixed to the rotating structure and move with it.
  - The carriage supports a small vehicle platform or manipulator that interacts with the parked car.

– Technical Advantages of Stepper Motor + Timing Belt Design:
  + High precision: Each motor step corresponds to a fixed, known rotation angle—ideal for systems with defined stop positions.
  + No slippage: The timing belt ensures synchronized rotation with minimal backlash.
  + Simple mechanics: Easy to fabricate, assemble, and maintain.
  + Scalability: The number of parking spots can be increased by simply changing the program and pulley ratio.
  + Cost-effective: Only one vertical lifting mechanism is needed for the entire structure, reducing redundancy.

– Applications and Future Improvements:
  + Ideal for model-scale or actual full-size rotary parking systems.
  + Can be upgraded by replacing the stepper motor with a servo motor for closed-loop feedback and smoother control.
  + Angular sensors or encoders can be integrated for position correction.
  + Easily integrated into a smart IoT or PLC-based system for full automation.

6.5. Complete model design

- In recent decades, the expansion of urban populations and rapid vehicle ownership have led to increasing pressure on city infrastructure, especially parking availability. Traditional parking systems, which require large horizontal space, are no longer efficient or viable in densely populated urban areas.

  To address this problem, the Smart Multi-Level Parking System (SMLPS) was developed. This system aims to minimize land usage by stacking vehicles vertically and automating the parking and retrieval process through an intelligent electromechanical design. This project presents a scaled-down prototype of such a system using accessible components, including a lead-screw-driven lift, Arduino-based control system, OpenCV-based license plate recognition, and RFID access control.

  This comprehensive description explains the system's architecture in detail, covering mechanical design, electrical and control systems, image processing, and user interaction interface.

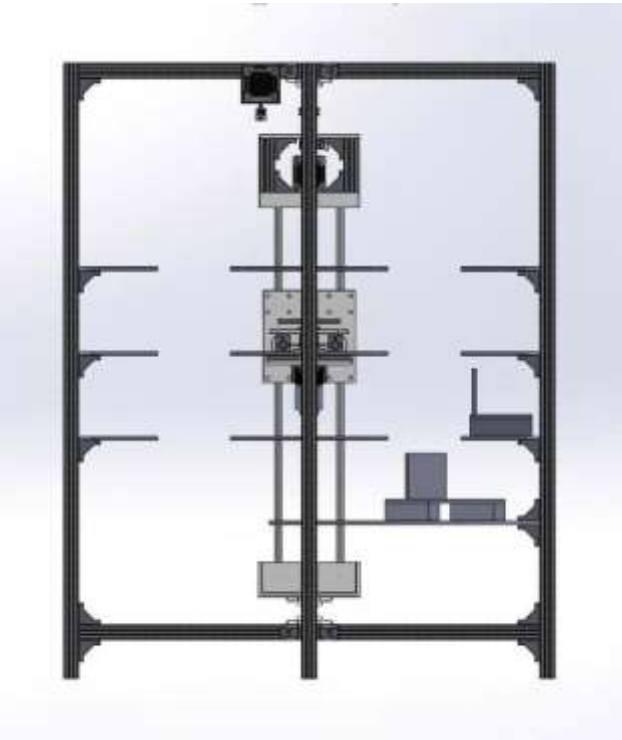

Figure 3.15 Complete model design

- Mechanical design:
  + Structural Framework
    - he physical structure of the system is built from extruded aluminum profiles (20x20 or 40x40 mm), known for their durability, ease of assembly, and modularity. The frame forms a vertical tower that accommodates multiple vehicle slots on different levels, divided symmetrically on both sides of a central lift mechanism.
    - Key Features:
    - Modularity: Allows the system to be scaled easily—more levels or columns can be added.
    - Rigidity: Aluminum profiles offer excellent structural integrity while being lightweight.
    - Customization: Brackets, plates, and sensor mounts can be directly attached to grooves.
    - The base includes crossbars that prevent deformation and distribute load evenly. The vertical columns support trays, rails, and the lead screw assembly.
  + Vehicle Tray Platforms
    - Each vehicle is placed on a platform tray, which can be moved vertically and horizontally. These trays are fabricated from acrylic (mica), aluminum, or steel sheets, depending on the size and load requirements.
    - Shape: Generally rectangular or L-shaped to support car wheels.
    - Support system: Bolted to the structure or suspended via arms or brackets.
    - Tray Guides: Rails and stoppers ensure precise alignment during insertion and retrieval.
    - Each platform corresponds to a fixed parking slot, and the total number of slots can be adjusted depending on system height and user demand.

- Vertical Motion System (Elevator):
  + A critical component of the system is the car elevator, which transports cars vertically between levels. It uses a lead screw mechanism, which converts rotary motion into linear motion.
  + Components:
    - Lead Screw (Acme or Ball Screw): Ensures smooth and precise motion.
    - Sliding Nut: Moves along the screw as the motor rotates.
    - Guide Rails: Two or more linear rods provide lateral stability.
    - Lift Platform: A rigid base with a pusher/puller mechanism or rails to accommodate the car tray.
  + Motor:
    - Stepper Motor (e.g., NEMA 23) or a DC Gear Motor drives the lead screw.
    - The motor is mounted on the top or bottom and coupled with the screw using a shaft coupler.
    - A limit switch system prevents over-travel and ensures safety.
- Horizontal Tray Transfer:
  + Once at the correct floor, the vehicle tray must be transferred from the lift platform into the appropriate parking bay.
  + Transfer Mechanisms:
    - Rack and Pinion or Belt Drive: Moves the tray linearly into the slot.
    - Linear Actuators: Controlled via relay or motor driver.
    - Rollers or Conveyor Belts: Optional for smooth entry/exit of trays.
  + Limit switches or optical sensors determine the exact tray position during insertion and retrieval.

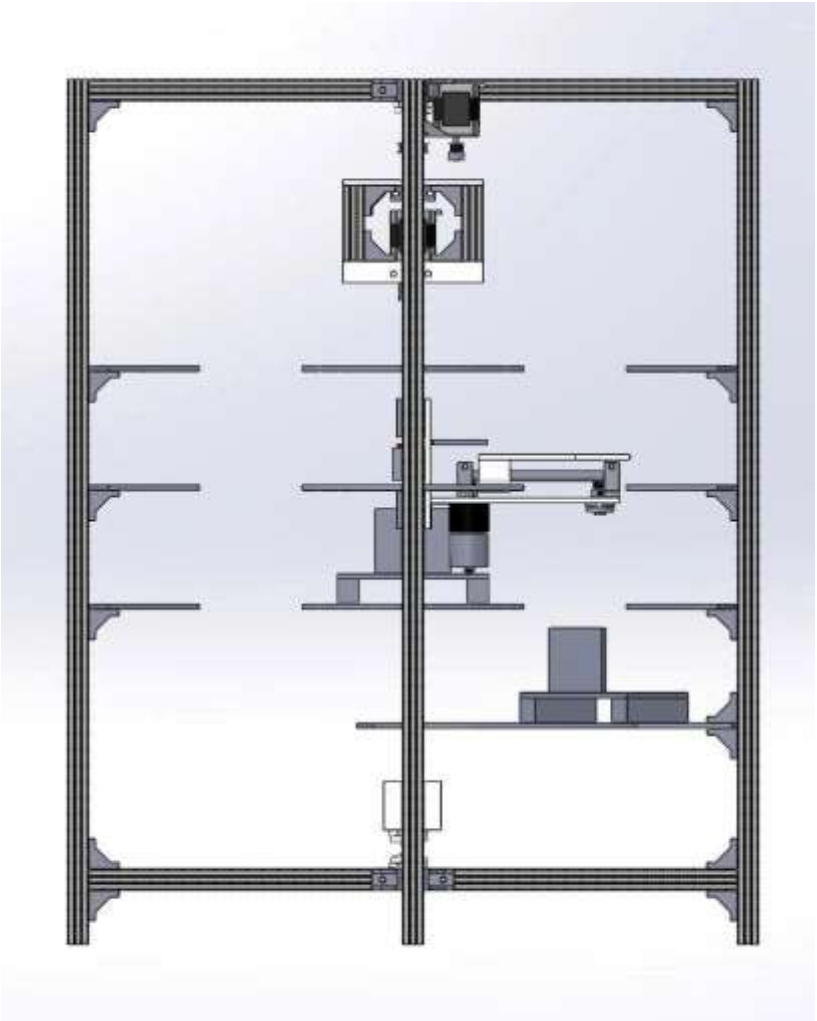

Figure 3.16 Complete model design

- Central Controller: Arduino:
  + The brain of the system is an Arduino Mega (or Uno for simpler versions). It controls all electrical components:
    - Motors (via motor drivers)
    - Sensors (IR, ultrasonic, limit switches)
    - RFID reader and serial input
    - Communicates with PC software or display interface
  + The Arduino is programmed in C/C++ (Arduino IDE) and handles real-time task scheduling using state machines or timing interrupts.

- Sensors and Feedback:
  + IR/Optical Sensors: Detect tray position and car presence.
  + Limit Switches: Prevent over-travel at top/bottom of lift shaft.
  + Ultrasonic Sensors: Measure distance to avoid collision.
  + Hall Effect Sensors: Detect rotation position of motors.
  + RFID Reader (RC522): Recognizes user cards and maps them to parking slots.
- Power Supply and Protection:
  + Motors are powered via a 12V or 24V power supply, with separate lines for logic and motor power.
  + Protection includes fuses, diodes, and current sensors for overload monitoring.
- Parking a Car:
  + Car arrives at the entry position (on lift platform).
  + User scans RFID or license plate is recognized.
  + System checks for nearest available slot.
  + Elevator raises car to appropriate floor.
  + Tray is inserted horizontally into the slot.
  + System updates database and confirms completion.
- Retrieving a Car:
  + User requests car via RFID or license plate input.
  + System locates car tray and brings it to ground level.
  + Car is delivered to the user.
  + Tray is marked as empty and ready for next use.
- Safety and fail-safe systems:
  + Emergency Stop Button on physical panel.
  + Current sensors to detect motor stall or overload.
  + Watchdog timer in Arduino to reset in case of code freeze.
  + Mechanical stoppers in tray and lift structure.
  + Software limits to prevent unauthorized access or movement.

- Future expansions and enhancements
  + AI Integration: Use machine learning to predict parking behavior.
  + Solar Power Supply: Increase energy efficiency.
  + Voice Control: Enable speech-based commands.
  + Real-time Cloud Monitoring: Integration with city-wide parking networks.
  + Robotic Arms: For higher precision in transferring trays.

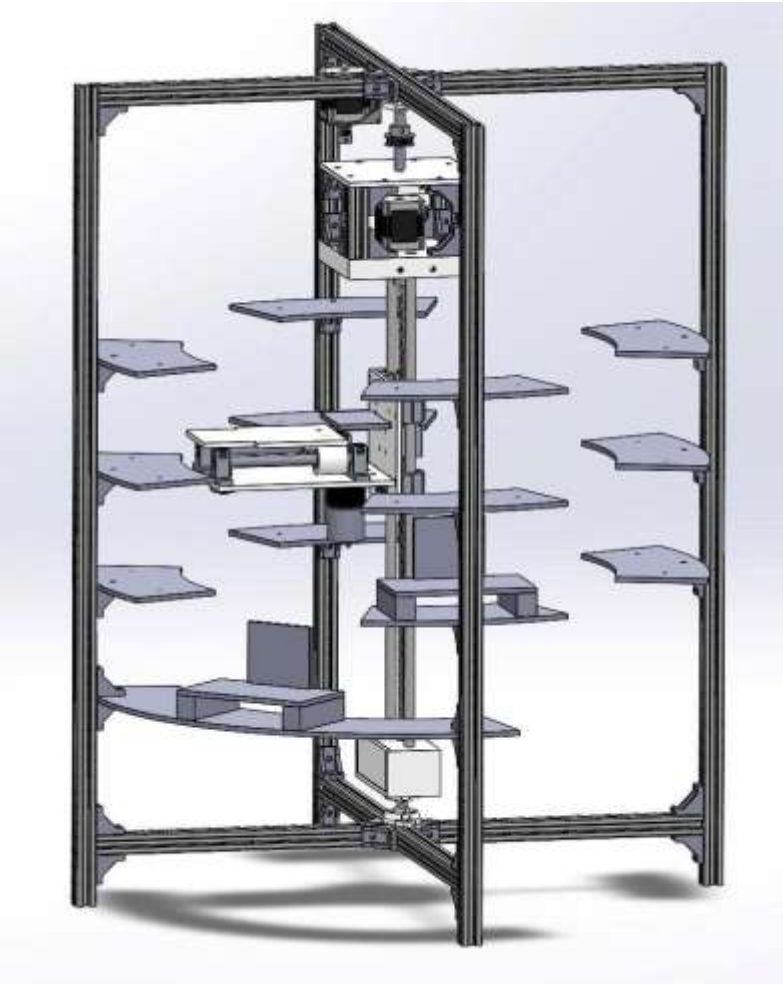

Figure 3.17 Complete model design

CHAPTER 4: CALCULATION AND DESIGN OF CONTROL SYSTEM

1. Block diagram of the system

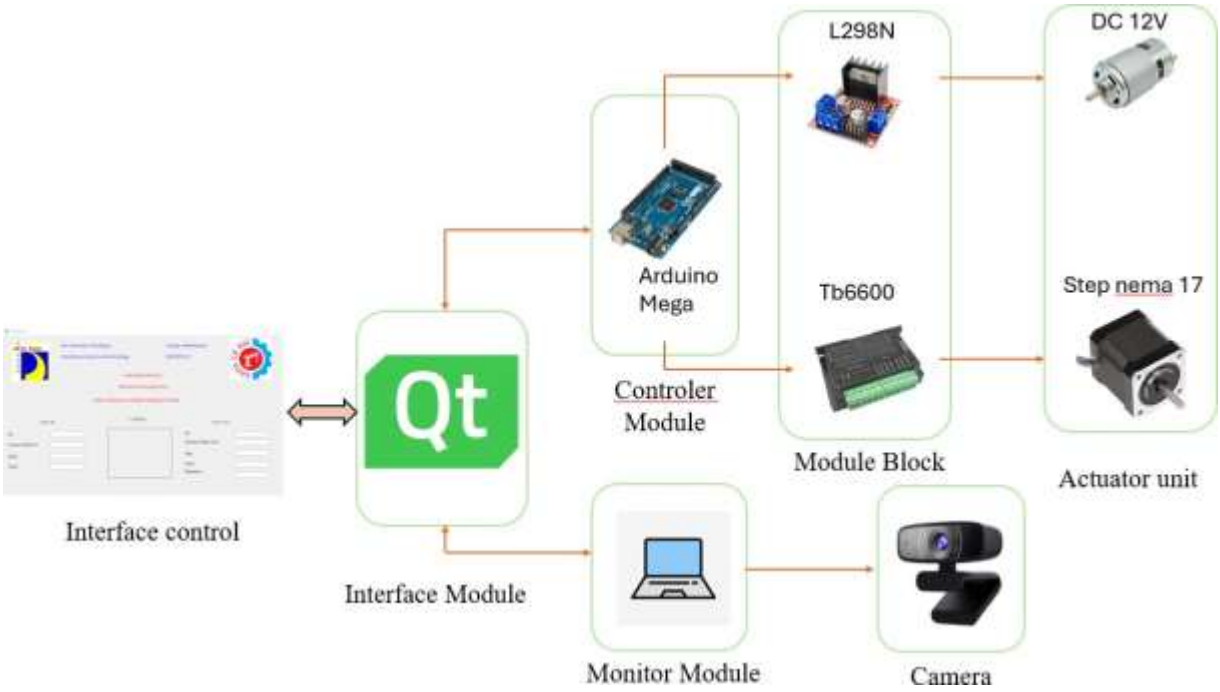

Figure 4.1 Block diagram

2. Arduino

2.1. What is Arduino?

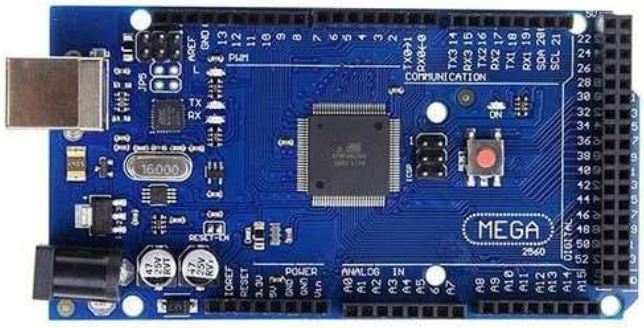

Figure 4.2 Arduino

- Arduino is a microcontroller board designed by a group of Italian professors and students, first introduced in 2005. Arduino boards are used to sense and control various objects. They can perform multiple tasks, such as receiving signals from sensors and controlling lights, motors, and other objects. Additionally, the board can interface with various modules like RFID readers, Ethernet shields, SIM900A, etc., to enhance its applicability. [4]
- The hardware consists of an open-source circuit board built on an AVR Atmel 8-bit microcontroller platform, or ARM, Atmel 32-bit, among others. There are currently six versions of Arduino hardware, with Arduino Uno and Arduino Mega being the most commonly used. [4]
- The software for programming Arduino boards is the Arduino IDE. [4]

2.2. Applications of Arduino

Arduino has many applications in life and in creating high-quality electronic devices. Some applications include [5]:

+ Robot programming: Arduino plays a critical role in the processing center, allowing for robot activity control.
+ Drone programming: This is an application with significant potential for the future.
+ Interactive gaming: Arduino can be used to interact with joysticks, screens, and more to play games such as Tetris, Brick Breaker, Mario, and many creative games.
+ Sensor-based lighting control: Arduino is a critical component in traffic lights and programmable light effects to highlight advertisements.
+ 3D printing and other projects: The use of Arduino depends on the user's creativity.

2.3. Arduino IDE Programming Software

Figure 4.3 Arduino IDE

Arduino offers an open-source integrated programming environment (IDE) that helps users write code and upload it to the Arduino board. This cross-platform IDE supports a variety of Arduino boards and unique features. Its interface is streamlined, making it suitable for both professional and amateur users.

3. Stepper Motor Driver TB6600

3.1. Introduction

The TB6600 stepper motor driver uses the TB6600HQ/HG IC, designed for stepper motors such as 42/57/86 2-phase or 4-wire motors with a load current of 4A/42VDC. It is applied in machines like CNC, laser cutters, or other automatic equipment. [6]

Figure 4.4 TB6600

3.2. Specifications of the TB6600 Stepper Motor Driver

- Input voltage: 9V– 42V.
- Maximum supply current: 4A.
- High-speed optically isolated input.
- Integrated overcurrent and overvoltage protection.
- Weight: 200g.
- Dimensions: 96 × 71 × 37 mm.

3.3. Installation and Connection

- DC+: Connect to a power source of 9–40VDC.
- DC-: Negative (-) terminal of the power source.
- A+ and A-: Connect to one coil of the stepper motor.
- B+ and B-: Connect to the other coil of the stepper motor.
- PUL+: Speed control pulse signal (+5V) from the BOB to the M6600.
- PUL-: Speed control pulse signal (-) from the BOB to the M6600.
- DIR+: Direction control pulse signal (+5V) from the BOB to the M6600.
- DIR-: Direction control pulse signal (-) from the BOB to the M6600.
- ENA+ and ENA-: When signals are supplied to this pair, the motor will lose its holding torque and stop spinning.

4. Nema 17 Stepper Motor

4.1. Introduction

The Nema 17 stepper motor (42mm, 48mm) is commonly used in 3D printers, mini CNC machines, and often paired with a GT2 pulley.

Figure 4.5 Nema 17 stepper motor

4.2. Product Information

- Shaft diameter: 5mm. The shaft has a flat surface, ensuring secure mounting of pulleys/couplers during operation.
- Rated current: 1.5A, holding torque: 0.55 Nm, step angle: 1.8°.
- Wire length: 1m, XH2.54 standard connector, compatible with stepper motor outputs on RAMPS 1.4 or CNC shield V3 boards.
- Suitable power capacity for 3D printers and mini CNC machines.
- Low heat generation, smooth operation.
- Weight: 400g.

5. 12V DC

Figure 4.6 12V DC

- Voltage: 12V DC.
- Power: 3W.
- Weight: 250g.
- Motor speed: 3000 RPM.
- Gearbox output speed: 45 RPM.
- Gear ratio: 30:1.

6. 24V - 5A Power

Figure 4.7 24V – 5A power

- The 24V - 5A power utilizes switching power technology and is widely used for powering surveillance camera systems, LED displays, peripheral devices, and testing setups.
- Specifications of the 24V - 5A Power:
  + Input Voltage: AC 220V.
  + Output Voltage: DC 24V – current up to 5A (MAX).
  + Power: 48W.
  + Operating Temperature: -40°C to 65°C.
  + Storage Temperature: -20°C to 60°C.
  + Protection: Overvoltage and overload protection, short-circuit protection.
  + Material: Metal protective casing.

7. L298N Motor Driver Module

Figure 4.8 L298N

7.1. Introduce

The L298N is a motor driver module used for controlling DC motors and stepper motors. The module includes an L298 motor driver IC and a 5V voltage regulator (78M05). The L298N module can control up to 4 DC motors or 2 DC motors with the capability to control both direction and speed. [7]

7.2. Pin Configuration Diagram of L298N

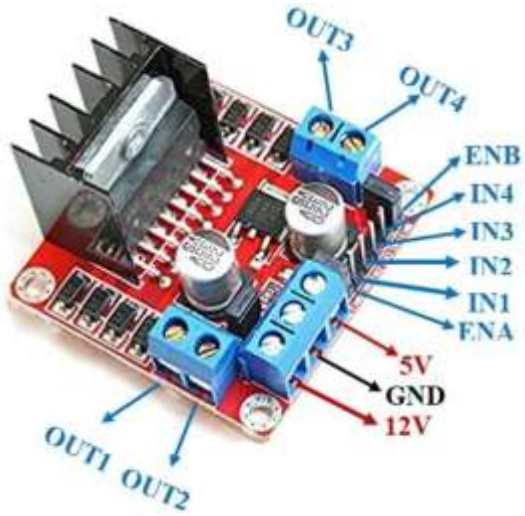

Figure 4.9 Pin Configuration Diagram of L298N [7]

Table 3. Pinout Diagram of L298N [7]

Pin Name	Description
IN1 & IN2	Input pins to control the rotation direction of Motor A
IN3 & IN4	Input pins to control the rotation direction of Motor B
ENA	Enables PWM signal for Motor A
ENB	Enables PWM signal for Motor B
OUT1 & OUT2	Output pins for Motor A
OUT3 & OUT4	Output pins for Motor B
12V	12V power input
5V	Supplies voltage for the internal logic circuit of the L298N IC
GND	Ground pin

7.3. Features and Technical Specifications of the L298 Module:

- Motor Supply Voltage (Max): 46V
- Motor Drive Current (Max): 2A
- Logic Voltage: 5V
- Operating Voltage of IC: 5–35V
- Operating Current of IC: 2A
- Logic Current: 0–36mA
- Maximum Power Output: 25W
- Current Sensing for Each Motor
- Equipped with a Heat Sink for Better Performance
- Power LED Indicator Included

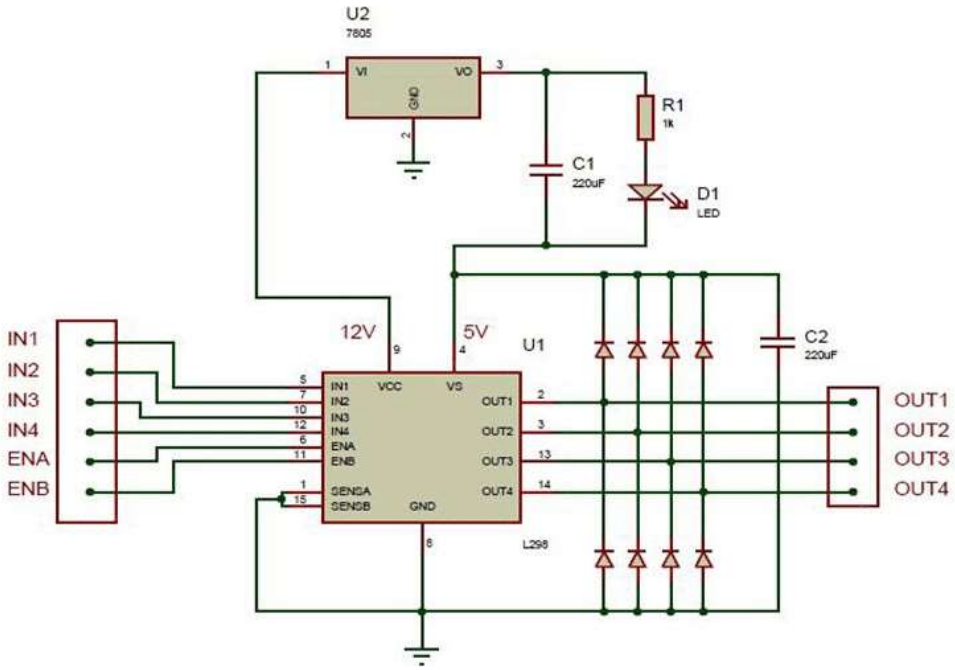

Figure 4.10 Sơ đồ mạch bên trong L298N

8. PyCharm Software

8.1. What is PyCharm?

– PyCharm is a hybrid platform developed by JetBrains as an Integrated Development Environment (IDE) for Python. It is commonly used for Python application development. Some major tech companies such as Twitter, Facebook, Amazon, and Pinterest also use PyCharm as their Python IDE of choice. [8]

Figure 4.11 PyCharm Software

- PyCharm can be run on Windows, Linux, or macOS. Additionally, it includes modules and packages that help developers build Python applications more efficiently, saving time and effort. Moreover, it can also be customized according to the specific needs of developers.

8.2. Features of PyCharm

a. Intelligent Code Editor

- Helps us write higher-quality code.
- It includes color schemes for keywords, classes, and functions, which improves code readability and understanding.
- Assists in identifying errors easily.
- Offers auto-completion and suggestions to help complete code efficiently.

b. Code Navigation

- It helps developers edit and improve code with less effort and time.
- With code navigation, programmers can easily jump to a function, class, or file.
- A developer can quickly locate an element, symbol, or variable in the source code.
- Moreover, by using the lens mode, developers can thoroughly inspect and debug the entire codebase.

c. Refactoring

- It allows for efficient and quick changes to both local and global variables.
- Refactoring in PyCharm enables developers to improve internal code structure without affecting its external behavior.
- PyCharm also helps better split classes and extended functions using the extract method.

d. Support for Various Web Technologies

- It helps developers create web applications using Python.
- It supports popular web technologies such as HTML, CSS, and JavaScript.
- Developers have the option to perform inline editing with this IDE. At the same time, they can preview the updated/created webpage.

- Developers can track changes directly in the web browser.
- PyCharm also supports AngularJS and NodeJS for developing web applications.

e. Support for Popular Python Web Frameworks

- PyCharm supports web frameworks such as Django.
- It provides auto-completion and suggestions for Django parameters.
- Helps debug Django code.
- Supports popular web frameworks like web2py and Pyramid.

f. Support for Python Scientific Libraries

- PyCharm supports scientific Python libraries such as Matplotlib, NumPy, and Anaconda.
- These scientific libraries assist in building projects related to Data Science and Machine Learning.
- Supports interactive charts that help developers better understand data.
- It can integrate with various tools like IPython, Django, and Pytest, promoting innovative solutions.

8.3. Advantages and Disadvantages of Using PyCharm

a. Advantages

- PyCharm is very easy to install.
- PyCharm is a user-friendly IDE.
- It has many useful plugins and shortcuts.
- PyCharm integrates library and IDE features like auto-completion and syntax highlighting.
- Allows one-click source code navigation.
- Saves software development time.
- Error highlighting features further improve the development process.
- Has a large Python developer community, making it easy to resolve doubts and questions.

b. Disadvantages

- PyCharm is not free, and its Professional version is quite expensive.
- Auto-completion might not be very effective for beginner programmers.
- It can cause issues when fixing tools such as virtual environments (venv).

9. Visual Studio Code Software

9.1. What is Visual Studio Code?

Figure 4.12 Visual Studio Code

- Visual Studio Code (VS Code) is a free source code editor available on three platforms: Windows, macOS, and Linux, developed by Microsoft. Visual Studio Code is highly regarded by IT professionals as it represents a perfect blend between an IDE and a code editor. [9]
- Visual Studio Code also supports platforms such as JavaScript, TypeScript, and Node.js. Specifically, its role is to provide a rich new ecosystem for various programming languages. [9]

9.2. Difference Between Visual Studio Code and Visual Studio

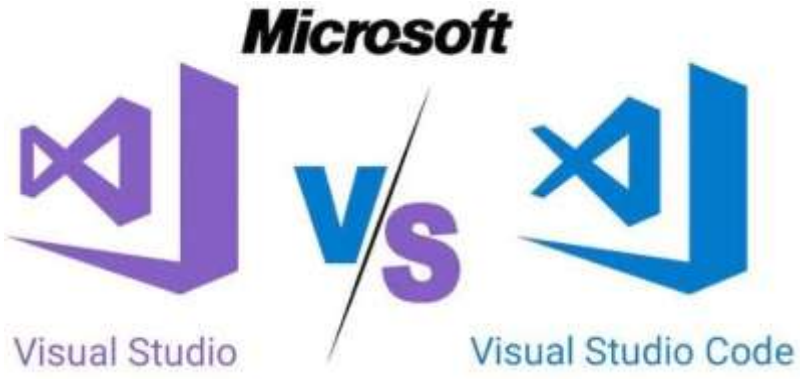

Figure 4.13 Visual Studio Code vs Visual Studio

Many people often confuse Visual Studio Code with Visual Studio because their names sound quite similar. In reality, these two are completely different versions. Specifically [9]:

- Visual Studio is a "full-featured" and "convenient" development environment.
- Visual Studio is a text editor within software.
- Visual Studio Code is a cross-platform text editor supporting Linux, macOS, and Windows, and it can be extended with plugins according to your needs.
- Visual Studio can only run on either Windows or Mac platforms, which are two different operating systems.
- The processing speed of Visual Studio Code is considered faster than that of Visual Studio.
- The file size of Visual Studio Code is smaller than Visual Studio.

9.3. Outstanding Features of Visual Studio Code

a. Support for Multiple Programming Languages

Visual Studio Code easily recognizes programming languages such as CSS, C++, C#, HTML, F#, etc., and provides warnings if your program contains errors. [9]

b. Diverse Feature Support Across Platforms

Usually, code editors can only be used on a single platform such as Windows, Mac, or Linux. However, Visual Studio Code can operate simultaneously on all three platforms. [9]

c. Secure Data Storage

Data security is highly demanded, especially in IT programming systems. Users can trust Visual Studio Code as it can easily connect with Git or any existing data repositories. [9]

d. Extension Marketplace

If the developer's programming language is not supported by default, they can install additional extensions. This does not affect software performance because extensions run independently like separate programs. [9]

e. Web Support

Visual Studio Code supports web applications and also includes its own text editor for designing and building websites. [9]

f. Hierarchical Data Storage

Visual Studio Code offers the advantage of organizing important files into folders, as most code files are stored within similar directory structures. [9]

g. Code Assistance

For user convenience, some code snippets can be customized or modified. Visual Studio Code provides suggestions or alternative options for programmers. [9]

h. Integrated Terminal Support

Programmers do not need to switch between multiple windows or return to the root directory during work because Visual Studio Code integrates a built-in terminal. [9]

i. Multi-tasking Screen

Users can open multiple folders and files at the same time, even if they are unrelated.

j. Git Support

Visual Studio Code can directly clone code from GitHub, serving as a replacement and storage platform. [9]

k. Intellisense

Visual Studio Code is more advanced than other code editors because it can detect incomplete code snippets. IntelliSense automatically helps programmers complete missing syntax, even if variables are not declared. [9]

9.4. Reasons to Use Visual Studio Code

- Low file size.
- Strong development across the three major operating systems worldwide. Visual Studio Code is developed by Microsoft, so it has strong compatibility with Windows, and it also supports macOS and Linux.
- User-friendly and intuitive interface design.
- Optimized coding and debugging process. Visual Studio Code helps save time with keyboard shortcuts for opening functions or inserting lines of code, avoiding interruptions while coding. You can also customize or change shortcuts to suit your usage.
- Excellent architecture and extensibility. Visual Studio Code is built on Electron combined with intelligent language technologies such as JavaScript and Node.js, along with its own powerful operation. This creates a wonderful experience for users.
- Large support community. Over the years, Visual Studio Code has secured its position with a large user base worldwide. If you encounter any issues, you can get help from communities such as Microsoft, Reddit, StackOverflow, etc.

9.5. Who Should Use Visual Studio Code?

Microsoft has invested significant effort into Visual Studio Code, integrating it with the most advanced technologies. Visual Studio Code is suitable for a wide range of users, not just professional developers, such as:

- Students majoring in Information Technology.
- Individuals working in software testing and quality assurance, commonly known as Testers.
- Professionals responsible for managing database systems, also known as Data Administrators.

10. Webcam

10.1. Introduce

A webcam is a compact digital imaging device that allows real-time transmission of images and audio to a computer or the internet. Originally developed in 1991 at the University of Cambridge to monitor a coffee pot in a laboratory, webcams have now become an essential part of modern computing devices, including laptops, desktops, and even smartphones.

Figure 4.14 Webcam

10.2. Detailed Structure and Working Principle of a Webcam

- A standard webcam typically consists of the following components:
  + Lens: Captures light and visual information from the environment.
  + Image Sensor: Converts light into electrical signals; commonly uses CMOS or CCD sensors.
  + Image Processor: Processes the electrical signals into digital image data.

+ Memory: Stores image data, either built-in or external (e.g., SD card).
+ Interface: Connects the webcam to the computer or internet via USB or wireless connection.

– Working Principle: When operating, the lens gathers light from the surroundings and focuses it onto the image sensor. The sensor then converts the light into electrical signals. These signals are processed by the image processor and converted into digital data, which is transmitted to a computer or online platform for display, storage, or live streaming.

10.3. Common Applications of Webcams

– Video Calling and Online Conferencing: Webcams allow video calls and virtual meetings via platforms like Zoom, Skype, or Google Meet, enabling real-time communication across distances.
– Gaming and Virtual Reality (VR): Some 3D games use webcams to capture player movements, enabling gesture control or immersive VR experiences.
– Security and Surveillance: Webcams are used in security systems for real-time monitoring, sending live feeds to a control center, or recording for later review.
– Facial Recognition and Login: Many modern laptops use webcams for facial recognition, allowing secure and fast logins.
– Video Recording and Photography: Webcams can be used to record video clips or take photos for both personal and professional use.

11. Flowchart

11.1. Main algorithm flowchart

The flowchart of the main parking system algorithm illustrates the full cycle of vehicle entry and exit management. It starts by detecting whether a vehicle is arriving or departing. For incoming vehicles, the system checks for available parking slots, scans the RFID tag at the chosen slot, and captures the license plate image using a camera. The license plate is linked to the RFID tag and the vehicle is guided to its designated slot. For exiting vehicles, the system reads the RFID tag, identifies the slot,

and re-scans the license plate to match it with the stored data. Upon a successful match, the vehicle is moved out and parking duration and fees are displayed. If mismatches occur, the system reprocesses the input image for verification.

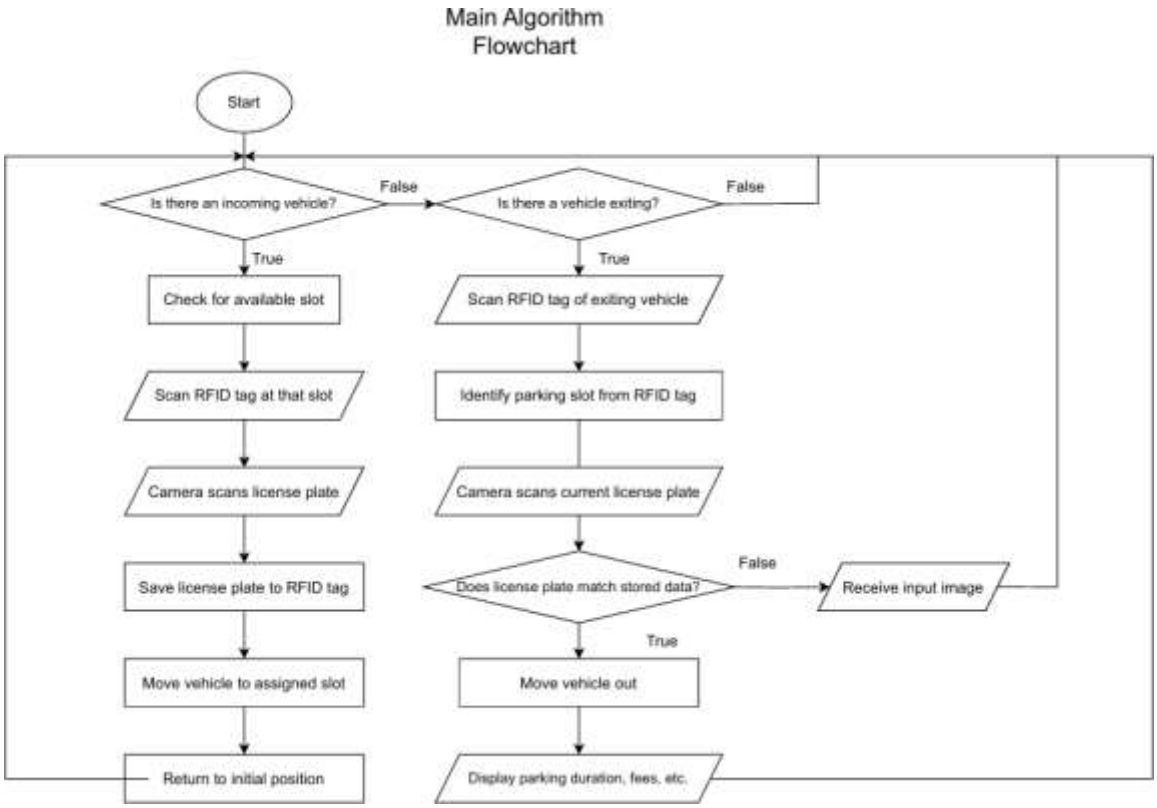

Figure 4.15 Main algorithm flowchart

11.2. Character recognition in license plate flowchart

The character recognition flowchart for license plates describes the process of image processing to extract and recognize characters. The input license plate image is first converted to grayscale, then the algorithm scans through regions containing potential characters, filtering by valid size. Qualified regions are processed and passed to an SVM model for character prediction. The results are sorted from left to right and combined into a string representing the license plate.

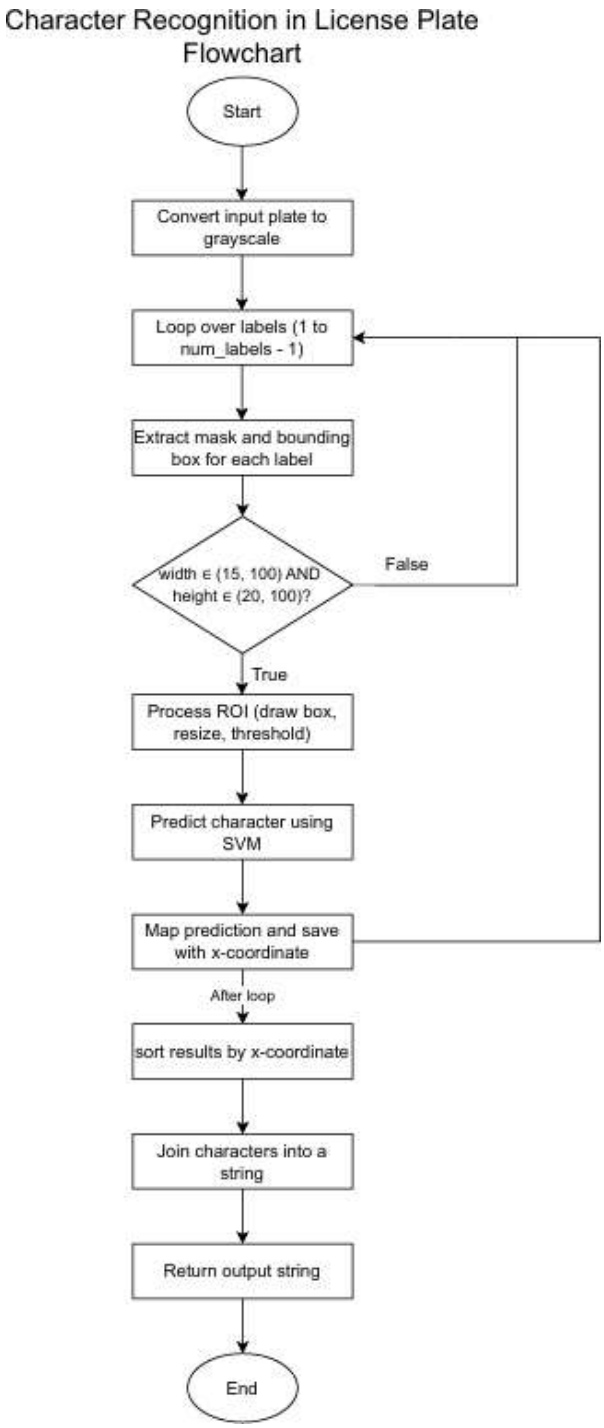

Figure 4.16 Character recognition in license plate flowchart

11.3. License plate detection flowchart

The license plate detection flowchart describes the image processing steps to locate a license plate. The system receives an input image, converts it to grayscale, applies Gaussian blur, and uses Canny edge detection. It then finds and sorts the top 5 contours and checks if any contour has 4 vertices (indicating a rectangular shape). If such a contour is found, it is assigned as the license plate, cropped from the image, and the cropped plate image is returned.

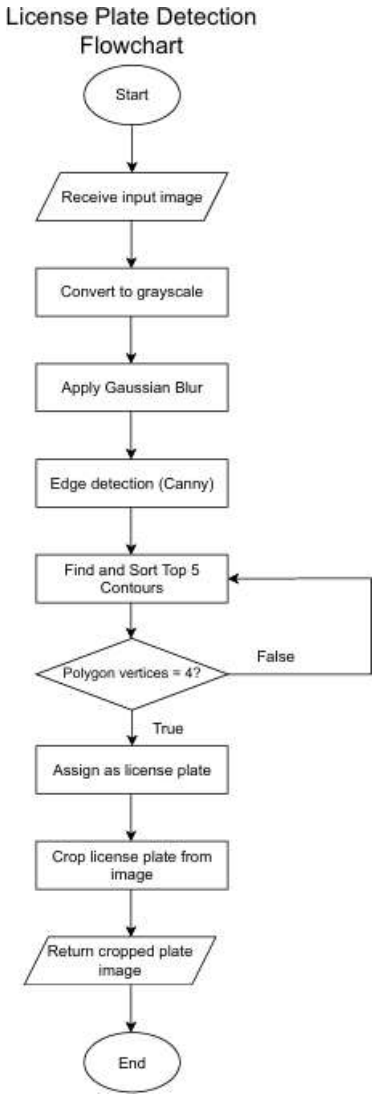

Figure 4.17 License plate detection flowchart

11.4. Serial data processing flowchart

This serial data processing flowchart outlines how a system handles vehicle entry and exit through serial data. If the data starts with 'A', it indicates a vehicle entering: the system extracts the ticket number and license plate, then saves the entry time and data. If the data starts with 'B', it means a vehicle is exiting: it checks the entry record, reads and verifies the license plate, records exit time, and then displays time and fee information if the plates match.

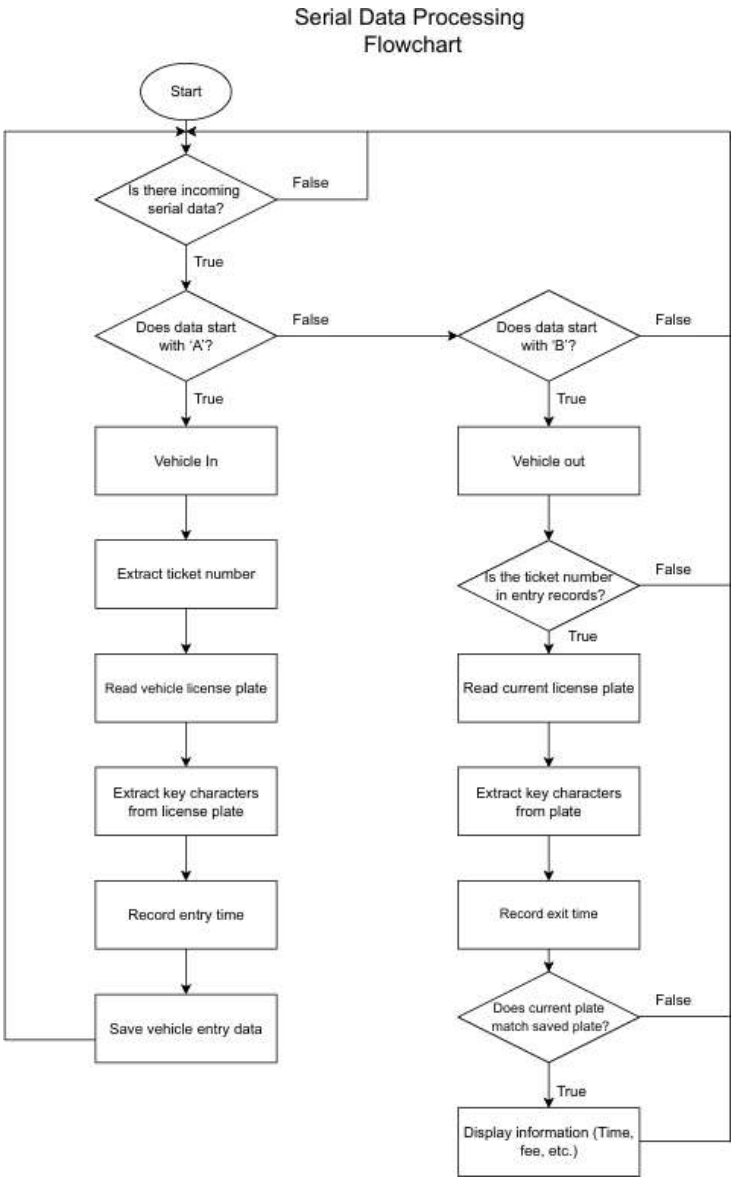

Figure 4.18 Serial data processing flowchart

11.5. Camera processing flowchart

This camera processing flowchart describes the process of capturing and displaying webcam footage. It begins by opening the default webcam. If successful, it starts reading frames and processes serial data. Each frame is captured, saved, converted to QImage format, and displayed in a PyQt GUI. The loop continues until the 'q' key is pressed, after which the webcam is released and all OpenCV windows are closed.

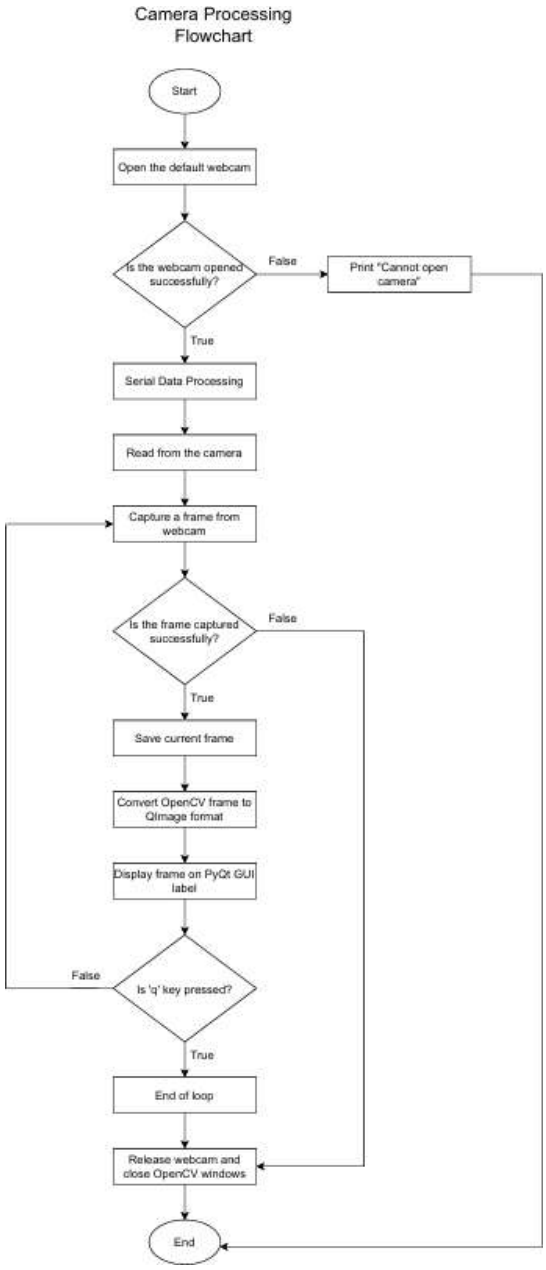

Figure 4.19 Camera processing flowchart

11.6. Position handling flowchart

This position handling flowchart outlines the process of managing a parking slot using RFID cards. It begins by detecting a new card, reads its UID, and compares it to a predefined ID. If matched, it checks the car's status (VT1). If the car is not parked (VT1 = 0), it triggers the "Position 1 In" procedure and updates the status. If the car is already parked (VT1 = 1), it runs the "Position 1 Out" procedure and resets the status. This process loops continuously.

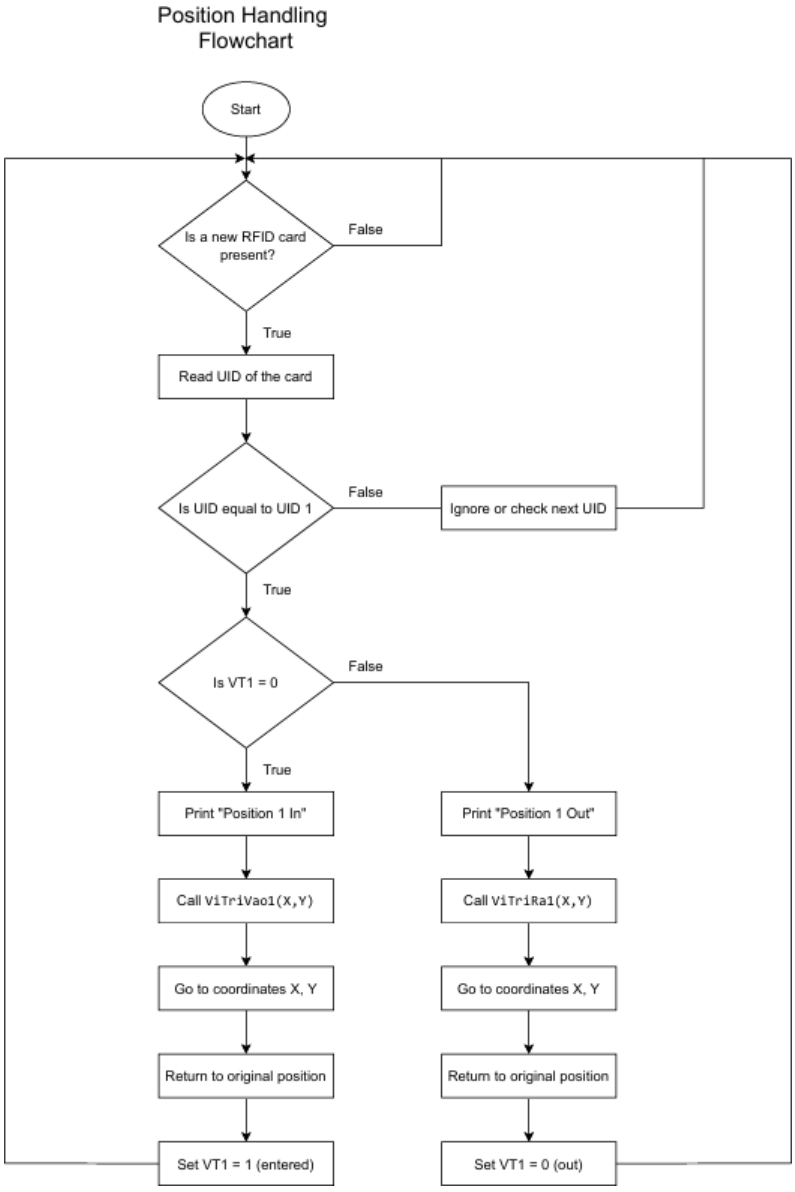

Figure 4.20 Position handling flowchart

CHAPTER 5: CONCLUSION

1. Summary

In this project, the team researched, designed, and built a model of a multi-level smart parking system. The system uses sensors to detect vehicle positions, microcontrollers to process data, and a screw mechanism to control the lifting and lowering motion of the parking levels. Additionally, the team integrated several user-support features such as displaying available parking spots and automatically managing the vehicle arrangement process.

Figure 5.1 Realistic model

Figure 5.2 System electrical box

Figure 5.3 Control interface

The interface shown is the central control panel of the smart parking system, designed to help users monitor and manage the vehicle entry and exit process in an intuitive and efficient manner. It is divided into three main sections: GATE IN (on the left) displays information when a vehicle enters the parking area, such as the RFID card ID, license plate number, date, and entry time; CAMERA (center) shows the live feed from the webcam, which is used for automatic license plate recognition via OpenCV; and GATE OUT (on the right) displays information when a vehicle exits, including the RFID card ID, license plate number, exit time, and calculated payment. This interface not only ensures ease of use and clarity, but also acts as a bridge between the image processing software, motor control system, and the human operator.

2. Evaluate

– Initial parameters assumed

Parameter	Assumed Value
Model car weight	0.1 kg
Number of levels in the model	3 levels
Vertical height per level	0.2 m
Average lifting speed	0.1 m/s
Lifting time per level	2 seconds
Number of parking slots	12 slots
RFID reading accuracy	100%
License plate recognition accuracy	100%
Total time for 1 cycle	15 - 30 seconds

- After testing the model, the team has drawn the following evaluations:
  + The cycle time at the nearest position is 20 seconds, which is slower than the initially expected 15 seconds.
  + The cycle time at the farthest position is 45 seconds, compared to the expected 30 seconds.
  + After 6 test runs, the RFID scanning success rate was 100%, but the license plate recognition rate was 4 out of 6, equivalent to 66.67%. This may be due to the webcam's insufficient resolution and inadequate lighting conditions, which caused it to fail in recognizing the license plates.

3. Achievements

- Designed a 3D model using SolidWorks
- Calculated and selected appropriate types of motors
- Completed mechanical drawings
- Developed a system algorithm flowchart
- Built the control system
- Processed images for license plate recognition
- The model has been completed and operates stably according to the initial requirements. The system can detect vehicle entry/exit, display available parking spots, control level transitions flexibly, and ensure vehicle safety during movement. The project has achieved its technical and operational goals for the prototype model.
- Optimized parking space usage under limited area conditions
- Reduced operational labor through automation

4. Unachieved Goals

- The model is still relatively simple and has not yet integrated advanced technologies such as remote control via smartphone.
- Processing speed and accuracy are still limited by the hardware capabilities.
- The actual performance has not been evaluated under large-scale operating conditions.

5. Future Work

- Integrate a license plate recognition system using cameras and image processing.
- Develop parking control and management software on a web platform or mobile application.
- Upgrade hardware to improve stability, expand the model's scale, and apply it in real-world scenarios.
- Connect the system to the IoT network to enhance remote monitoring and control capabilities.

REFERENCES

[1] N. V. L. Nguyen Trong Hiep, Machine Detail Design, Hanoi: Education Publishing House, 1999.

[2] C. M. L. Luu Duc Binh, Mechanical Measurement Engineering, Vietnam Education Publishing House, 2015.

[3] Nguyễn Thúc An, Nguyễn Đình Chiểu and Khổng Doãn Điền, Giáo trình cơ học lý thuyết, NXB Xây dựng, 2007.

[4] Hour Of Code Việt Nam, "Giới thiệu về lập trình Arduino," 2025. [Online]. Available: https://hourofcode.vn/gioi-thieu-ve-lap-trinh-arduino/. [Accessed 5 2025].

[5] H. Đàm, "Lập trình Arduino – Những ứng dụng hữu ích trong cuộc sống," 22 10 2023. [Online]. Available: https://teky.edu.vn/blog/lap-trinh-arduino/. [Accessed 10 6 2025].

[6] CNCVietPro, "Driver điều khiển động cơ bước TB6600," [Online]. Available: https://cncvietpro.com/san-pham/driver-dieu-khien-dong-co-buoc-tb6600/. [Accessed 10 6 2025].

[7] Mecsu.vn, "Module điều khiển động cơ L298N," 1 8 2023. [Online]. Available: https://mecsu.vn/ho-tro-ky-thuat/module-dieu-khien-dong-co-l298n.lGG. [Accessed 10 6 2025].

[8] LANIT, "PyCharm là gì? Tính Năng, Lợi ích của IDE Python," 2025. [Online]. Available: https://lanit.com.vn/pycharm-la-gi.html. [Accessed 2025].

[9] CloudFly.vn, "Visual Studio Code Là Gì? Tính Năng Vượt Trội Của Visual Studio Code," 6 8 2024. [Online]. Available: https://cloudfly.vn/blog/visual-studio-code-la-gi-tinh-nang-vuot-troi-cua-visual-studio-code. [Accessed 2025].

[10] L. D. Binh, Textbook of Machine Manufacturing Technology.

[11] N. P. Hai, SolidWorks Mechanical Design Textbook 2018.

APPENDIX

CODE PYCHARM

```
import sys
import cv2
import numpy as np
from PyQt6 import QtCore, QtGui, QtWidgets
from datetime import datetime
import serial
import threading

# Các biến và hàm nhận diện biển số
vehicle_in_info = {}
vehicle_out_info = {}
ticket_counter_in = 1
ticket_counter_out = 1
frame = None  # Biến lưu trữ frame webcam


def laybienso(image):
    grayscale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(grayscale, (5, 5), 0)
    edged = cv2.Canny(blurred, 50, 150)
    contours, _ = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=cv2.contourArea, reverse=True)[:5]
    for c in contours:
        perimeter = cv2.arcLength(c, True)
        approximation = cv2.approxPolyDP(c, 0.02 * perimeter, True)
```

```
        if len(approximation) == 4:
            number_plate_shape = approximation
            break
    x, y, w, h = cv2.boundingRect(number_plate_shape)
    number_plate = image[y:y + h, x:x + w]
    return number_plate


def laykytubiendai(number_plate):
    gray_plate = cv2.cvtColor(number_plate, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray_plate, 50, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)
    num_labels, labels = cv2.connectedComponents(thresh)
    results = []
    for label in range(1, num_labels):
        mask = np.uint8(labels == label) * 255
        contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        x, y, w, h = cv2.boundingRect(contours[0])
        if 15 < w < 100 and 20 < h < 100:
            cv2.rectangle(number_plate, (x, y), (x + w, y + h), (0, 255, 0), 2)
            roi = gray_plate[y:y + h, x:x + w]
            roi_resized = cv2.resize(roi, (30, 60))
            _, thresh1 = cv2.threshold(roi_resized, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)
            svm = cv2.ml.SVM_load('svm.xml')
            image_flattened = thresh1.flatten().astype(np.float32)
            result = svm.predict(image_flattened.reshape(1, -1))[1]
            mapping = {0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9: '9', 65: 'A',
```

```
66: 'B'}
            predicted_label = mapping.get(int(result[0, 0]), 'Unknown')
            results.append((x, predicted_label))
    sorted_results = sorted(results, key=lambda x: x[0])
    output_string = ''.join(label for _, label in sorted_results)
    return output_string



# Hàm xử lý dữ liệu từ cổng serial
def handle_serial_input(ser, ui):
    global ticket_counter_in, ticket_counter_out, vehicle_in_info, vehicle_out_info

    while True:
        if ser.in_waiting > 0:
            serial_data = ser.readline().decode('utf-8').strip()

            if serial_data.startswith('A'):
                ticket_number = serial_data[1:]
                number_plate = laybienso(frame)
                kytu = laykytubiendai(number_plate)
                time_in = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

                # Cập nhật giao diện
                ui.BienSoVao.setText(kytu)
                ui.NgayVao.setText(time_in.split()[0])
                ui.Giovao.setText(time_in.split()[1])

                vehicle_in_info[ticket_number] = {'number_plate': kytu, 'time': time_in}
                ticket_counter_in += 1
```

```
            ser.write(b'v')

        elif serial_data.startswith('B'):
            ticket_number = serial_data[1:]

            if ticket_number not in vehicle_in_info:
                continue

            number_plate = laybienso(frame)
            kytu = laykytubiendai(number_plate)
            time_out = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

            if vehicle_in_info[ticket_number]['number_plate'] != kytu:
                ser.write(b'e')
                continue

            time_in = datetime.strptime(vehicle_in_info[ticket_number]['time'], '%Y-%m-
%d %H:%M:%S')
            time_out = datetime.strptime(time_out, '%Y-%m-%d %H:%M:%S')
            duration = (time_out - time_in).total_seconds()
            cost = int(duration // 5) * 1000

            # Cập nhật giao diện cho xe ra
            ui.BienSoRa.setText(kytu)
            ui.NgayRa.setText(time_out.split()[0])
            ui.GioRa.setText(time_out.split()[1])
            ui.CanThanhToan.setText(f"{cost} VND")

            vehicle_out_info[ticket_number] = {'number_plate': kytu, 'time': time_out,
```

```
'cost': cost}
            ticket_counter_out += 1
            ser.write(b'r')



# Hàm xử lý webcam và cập nhật giao diện
def process_camera(window_name, ui):
    global frame
    cap = cv2.VideoCapture(0)  # Chỉ sử dụng webcam mặc định

    if not cap.isOpened():
        print("Không thể mở camera")
        return

    ser = serial.Serial('COM3', 9600, timeout=1)  # Cổng serial (kiểm tra lại cổng của bạn)
    serial_thread = threading.Thread(target=handle_serial_input, args=(ser, ui))
    serial_thread.start()

    while True:
        ret, img = cap.read()
        if not ret:
            break

        frame = img  # Lưu ảnh hiện tại

        # Chuyển đổi ảnh OpenCV thành QImage và cập nhật QLabel trong PyQt
        height, width, channels = img.shape
        bytes_per_line = channels * width
        qimg = QtGui.QImage(img.data, width, height, bytes_per_line,
```

```
QtGui.QImage.Format.Format_BGR888)

        ui.Camera.setPixmap(QtGui.QPixmap.fromImage(qimg))  # Cập nhật video lên
QLabel

        key = cv2.waitKey(50)
        if key == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
    ser.close()



# Giao diện PyQt
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1048, 368)
        self.centralwidget = QtWidgets.QWidget(parent=MainWindow)
        self.centralwidget.setObjectName("centralwidget")

        # Các thành phần giao diện như bạn đã làm
        self.BienSoVao = QtWidgets.QLineEdit(parent=self.centralwidget)
        self.BienSoVao.setGeometry(QtCore.QRect(190, 170, 131, 20))
        self.BienSoVao.setObjectName("BienSoVao")

        self.Camera = QtWidgets.QLabel(parent=self.centralwidget)
        self.Camera.setGeometry(QtCore.QRect(10, 10, 640, 360))
```

```
        self.Camera.setObjectName("Camera")

        self.NgayVao = QtWidgets.QLineEdit(parent=self.centralwidget)
        self.NgayVao.setGeometry(QtCore.QRect(190, 220, 131, 20))

        self.Giovao = QtWidgets.QLineEdit(parent=self.centralwidget)
        self.Giovao.setGeometry(QtCore.QRect(190, 270, 131, 20))

        self.BienSoRa = QtWidgets.QLineEdit(parent=self.centralwidget)
        self.BienSoRa.setGeometry(QtCore.QRect(190, 320, 131, 20))

        self.NgayRa = QtWidgets.QLineEdit(parent=self.centralwidget)
        self.NgayRa.setGeometry(QtCore.QRect(190, 370, 131, 20))

        self.GioRa = QtWidgets.QLineEdit(parent=self.centralwidget)
        self.GioRa.setGeometry(QtCore.QRect(190, 420, 131, 20))

        self.CanThanhToan = QtWidgets.QLineEdit(parent=self.centralwidget)
        self.CanThanhToan.setGeometry(QtCore.QRect(190, 470, 131, 20))

        # Khởi tạo giao diện
        MainWindow.setCentralWidget(self.centralwidget)
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Xe Vào Xe Ra"))
```

```
if __name___== "__main__":
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)

    # Chạy thread xử lý camera và cổng serial
    threading.Thread(target=process_camera, args=("Camera", ui)).start()

    MainWindow.show()
    sys.exit(app.exec())
```

CODE ARDUINO

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>  // Thêm thư viện Servo để điều khiển servo
const int RL = 2;
// Định nghĩa các chân cho RFID1
#define SDA1 8
#define RST1 4

// Định nghĩa các chân cho RFID2
#define SDA2 10
#define RST2 9

// Khai báo đối tượng MFRC522 cho hai module RFID
```

```
MFRC522 mfrc522_1(SDA1, RST1); // RFID 1
MFRC522 mfrc522_2(SDA2, RST2); // RFID 2

// Khai báo các biến string để lưu UID của các thẻ RFID dưới dạng DEC (thập phân)
String mathe1 = "95 213 50 40";
String mathe2 = "34 226 89 52";
String mathe3 = "163 98 231 167";
String mathe4 = "115 155 226 167";
String mathe5 = "83 17 124 167";
String mathe6 = "227 136 66 167";
String mathe7 = "99 121 126 146";
String mathe8 = "195 227 138 145";

// Khai báo các đối tượng Servo
Servo servo1;
Servo servo2;
const int CB = A0;
#define dirPin 7
#define stepPin 3

void Xoay()
{
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(1200);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(1200);
```

```
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(1200);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(1200);
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(1200);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(1200);
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(1200);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(1200);
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(1200);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(1200);
}
int t=0;
void Vitri(int i)
{
  while(i!=t)
  {
    if(digitalRead(CB)==LOW)
    {
      while(digitalRead(CB)==LOW)Xoay();
      t=t+1;
```

```
    if(t==9)
    t=1;
   }
   else
   Xoay();
  }
}
void setup() {
  // Khởi động giao tiếp SPI
  Serial.begin(9600);
  SPI.begin(); // Khởi động SPI

  // Khởi động RFID1
  mfrc522_1.PCD_Init();
  Serial.println("RFID 1 ready!");

  // Khởi động RFID2
  mfrc522_2.PCD_Init();
  Serial.println("RFID 2 ready!");

  // Gắn servo vào các chân 5 và 6
  servo1.attach(5);
  servo2.attach(6);

  // Đảm bảo cả 2 servo bắt đầu ở góc 0
  servo1.write(0);
```

```
  servo2.write(0);
  Serial.println("Khoi Dong");
  pinMode(RL,OUTPUT);
     pinMode(CB,INPUT);
  // Declare pins as output:
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);

  // Set the spinning direction CW/CCW:
  digitalWrite(dirPin, LOW);
  while(digitalRead(CB)==HIGH)
  Xoay();
}
int k=1;
void loop() {
  // Kiểm tra RFID 1
  if (mfrc522_1.PICC_IsNewCardPresent()) {
    digitalWrite(RL,HIGH);
    delay(600);
    digitalWrite(RL,LOW);
    if (mfrc522_1.PICC_ReadCardSerial()) {
      // Lấy UID của thẻ RFID1 dưới dạng DEC
      String rfid1UID = "";
      for (byte i = 0; i < mfrc522_1.uid.size; i++) {
        rfid1UID += String(mfrc522_1.uid.uidByte[i], DEC); // Chuyển đổi mỗi byte sang
DEC (thập phân)
        rfid1UID += " "; // Thêm khoảng trắng giữa các giá trị byte
```

```
}
rfid1UID.trim(); // Loại bỏ khoảng trắng thừa ở đầu và cuối chuỗi

// Kiểm tra và hiển thị mã thẻ tương ứng với UID đã lưu
if (rfid1UID == mathe1) {
  Serial.println("A1");
  k=1;
  for(int u = 0;u<3;u++)
  {
    Xoay();
  }
} else if (rfid1UID == mathe2) {
  Serial.println("A2");
  k=2;
  for(int u = 0;u<3;u++)
  {
    Xoay();
  }
} else if (rfid1UID == mathe3) {
  Serial.println("A3");
  k=3;
  for(int u = 0;u<3;u++)
  {
    Xoay();
  }
} else if (rfid1UID == mathe4) {
```

```
  Serial.println("A4");
  k=4;
  for(int u = 0;u<3;u++)
  {
    Xoay();
  }
}
else if (rfid1UID == mathe5) {
  Serial.println("A5");
  k=5;
  for(int u = 0;u<3;u++)
  {
    Xoay();
  }
}
else if (rfid1UID == mathe6) {
  Serial.println("A6");
  k=6;
  for(int u = 0;u<3;u++)
  {
    Xoay();
  }
}
else if (rfid1UID == mathe7) {
  Serial.println("A7");
  k=7;
```

```
      for(int u = 0;u<3;u++)
      {
        Xoay();
      }
    }
    else if (rfid1UID == mathe8) {
      Serial.println("A8");
      k=8;
      for(int u = 0;u<3;u++)
      {
        Xoay();
      }
    }else {
      Serial.println("A0");
    }

    mfrc522_1.PICC_HaltA(); // Dừng thẻ
  }
}

// Kiểm tra RFID 2
if (mfrc522_2.PICC_IsNewCardPresent()) {
  digitalWrite(RL,HIGH);
  delay(600);
  digitalWrite(RL,LOW);
  if (mfrc522_2.PICC_ReadCardSerial()) {
```

```
  // Lấy UID của thẻ RFID2 dưới dạng DEC
  String rfid2UID = "";
  for (byte i = 0; i < mfrc522_2.uid.size; i++) {
    rfid2UID += String(mfrc522_2.uid.uidByte[i], DEC); // Chuyển đổi mỗi byte sang DEC (thập phân)
    rfid2UID += " "; // Thêm khoảng trắng giữa các giá trị byte
  }
  rfid2UID.trim(); // Loại bỏ khoảng trắng thừa ở đầu và cuối chuỗi

  // Kiểm tra và hiển thị mã thẻ tương ứng với UID đã lưu
  if (rfid2UID == mathe1) {
    Serial.println("B1");
    k=1;
    for(int u = 0;u<3;u++)
    {
      Xoay();
    }
  } else if (rfid2UID == mathe2) {
    Serial.println("B2");
    k=2;
    for(int u = 0;u<3;u++)
    {
      Xoay();
    }
  } else if (rfid2UID == mathe3) {
    Serial.println("B3");
    k=3;
```

```
  for(int u = 0;u<3;u++)
  {
    Xoay();
  }
 } else if (rfid2UID == mathe4) {
  Serial.println("B4");
  k=4;
  for(int u = 0;u<3;u++)
  {
    Xoay();
  }
 }
 else if (rfid2UID == mathe5) {
  Serial.println("B5");
  k=5;
  for(int u = 0;u<3;u++)
  {
    Xoay();
  }
 }
 else if (rfid2UID == mathe6) {
  Serial.println("B6");
  k=6;
  for(int u = 0;u<3;u++)
  {
    Xoay();
```

```
    }
   }
   else if (rfid2UID == mathe7) {
    Serial.println("B7");
    k=7;
    for(int u = 0;u<3;u++)
    {
     Xoay();
    }
   }
   else if (rfid2UID == mathe8) {
    Serial.println("B8");
    k=8;
    for(int u = 0;u<3;u++)
    {
     Xoay();
    }
   }else {
    Serial.println("B0");
   }

   mfrc522_2.PICC_HaltA(); // Dừng thẻ
  }
}

// Kiểm tra lệnh từ Serial
```

```
  if (Serial.available() > 0) {
    char command = Serial.read();  // Đọc lệnh từ Serial

    if (command == 'v') {
      // Điều khiển servo1 mở góc 90 độ và sau 3 giây quay lại góc 0
      servo1.write(90);
      delay(3000);  // Đợi 3 giây
      servo1.write(0);  // Quay lại 0 độ
    }
    else if (command == 'r') {
      // Điều khiển servo2 mở góc 90 độ và sau 3 giây quay lại góc 0
      servo2.write(90);
      delay(3000);  // Đợi 3 giây
      servo2.write(0);  // Quay lại 0 độ
    }
    else if (command == 'e') {
      // Điều khiển servo2 mở góc 90 độ và sau 3 giây quay lại góc 0
      digitalWrite(RL,HIGH);
      delay(2000);  // Đợi 3 giây
      digitalWrite(RL,LOW);
    }
  }
  Vitri(k);
  //Serial.print(t);
  //Serial.println(k);
}
```