

**HO CHI MINH CITY UNIVERSITY OF
TECHNOLOGY AND EDUCATION
FACULTY OF INTERNATIONAL EDUCATION**



AUTONOMOUS CAR PROJECT
VEHICLE AUTOMATIC CONTROL SYSTEM

SUBJECT ID: VACS330333E_23_2_01FIE
SEMESTER 2 – ACADEMIC YEAR 2023-2024
Implementation: Group 4
Guiding lecturer: PhD. Lê Thanh Phúc

Ho Chi Minh City, May, 2024

**GROUP PROJECT FOR
VEHICLE AUTOMATIC CONTROL SYSTEM
SEMESTER 2 - ACADEMIC YEAR 2023-2024**

1. Subject ID: VACS330333E_23_2_01FIE

2. Guiding lecturer: PhD. Lê Thanh Phúc

3. List of members attending:

No.	Student's name	Student ID	Attempt %	Sign
1	Lê Hồ Minh Khoa	21145019	100%	
2	Nguyễn Đăng Quốc Khánh	21145584	100%	
3	Nguyễn Anh Quốc	21145606	100%	
4	Nguyễn Gia Bảo	21145576	100%	

- Attempt % = 100%

- Leader: Lê Hồ Minh Khoa

Feedback of Guiding lecturer

.....
.....
.....

May, 2024

GUIDING LECTURER

CONTENT

CHAPTER 1. INTRODUCTION.....	1
1.1. The reason for choosing this topic.....	1
1.2. The objective and the scope of the project	2
1.3. The methodology of the project	3
1.4. The task assignment	4
CHAPTER 2. LITERATURE REVIEW	5
2.1. Previous theoretical foundations on autonomous vehicles	5
2.2. The global state of autonomous vehicles.....	5
2.3. The challenges of autonomous vehicles	7
2.4. Software applications for autonomous vehicles	7
CHAPTER 3. THE PROCESS OF CREATING AUTONOMOUS VEHICLE	8
3.1. Algorithms of autonomous vehicle	8
3.1.1. Lane detection algorithm of autonomous vehicle.....	8
3.1.2. Control algorithm of autonomous vehicle	8
3.1.3. Decision making algorithm of autonomous vehicle	8
3.1.4. Integration algorithm of autonomous vehicle	9
3.2. Design of autonomous vehicle	9
3.2.1. The components preparing process.....	9
3.2.2. The chassis assemble process.....	12
3.2.3. The camera frame adding process.....	14
3.2.4. The camera attaching process	15
3.3. Programming process of autonomous vehicle	17
3.3.1. Building the Python code	17
3.3.2. Building the Arduino code	23
CHAPTER 4. AUTONOMOUS VEHICLE IN PRACTICAL EXPERIMENTATION	26
4.1. Testing the vehicle code	26
4.1.1. Testing the Python code	26

4.1.2. Testing the Arduino code	28
4.2. Experimenting the vehicle in practical	28
CHAPTER 5. CONCLUSION	30
5.1. Summarizing the accomplishments.....	30
5.2. Forward vision and future plans	30
REFERENCES	32

CHAPTER 1. INTRODUCTION

1.1. The reason for choosing this topic

Over the years, there have been significant advancements in artificial intelligence, machine learning, sensors, and computing power. These technological breakthroughs have paved the way for the development of autonomous cars. One of the primary motivations for autonomous cars is the potential to enhance road safety. Human error is a leading cause of accidents, and autonomous cars can mitigate this risk by minimizing human intervention and relying on advanced sensors and algorithms to make driving decisions.

Autonomous cars have the potential to optimize traffic flow and reduce congestion. With interconnected autonomous vehicles communicating with each other and with traffic infrastructure, they can coordinate movements and make more efficient use of road space, leading to smoother traffic flow. Autonomous cars have the potential to provide increased mobility options for individuals who cannot drive, such as the elderly or people with disabilities. They offer the possibility of independent transportation, improving quality of life for many individuals. They contribute to reducing carbon emissions and improving air quality. With the rise of electric autonomous vehicles, it becomes possible to reduce reliance on fossil fuels, transition to cleaner energy sources. The development and adoption of autonomous cars have significant economic implications. It lead to job creation in the technology and automotive sectors and changes in industries such as transportation, logistics, and ride-sharing services.

These reasons highlight the potential transformative impact of autonomous cars on various aspects of our lives. Therefore, this is a wise choice for researchers and students interested in this field.

1.2. The objective and scope of the project

1.2.1. The objective of the project

The primary objective is to create a fully functional autonomous driving system that can navigate and operate a vehicle safely and efficiently in various driving conditions.

A key objective is to prioritize safety and reliability in the autonomous driving system. This involves rigorous testing, validation, and adherence to safety standards to minimize the risk of accidents or malfunctions.

The project aim to improve the vehicle's perception of its environment through advanced sensor technologies such as cameras and other relevant sensors. Additionally, the decision-making algorithms should be optimized to make accurate and efficient driving decisions.

Developing a robust mapping and localization system is crucial for autonomous cars. The objective is to create a mapping system that can accurately identify and locate the vehicle's position in real-time, enabling precise navigation and path planning.

Building public acceptance and trust in autonomous cars is crucial for their widespread adoption. An objective of the project could be to address public concerns, educate stakeholders, and showcase the safety and benefits of autonomous driving technology.

1.2.2. The scope of the project

Our autonomous car research project will focus on developing the driving system and controlling the vehicle's speed within the laboratory. In this scope, we will focus on basic functions such as line detection and road tracking.

For line detection, we will research and develop algorithms and methods for vehicles to automatically recognize and track lines on the road surface. This will allow the car to automatically maintain the correct lane and avoid going over the limit.

For road tracking, we will research, develop algorithms and control systems so that the vehicle can stick to the correct lane and maintain a safe position on the road. This requires continuously determining and monitoring the vehicle's position and adjusting speed and steering direction to maintain stability and safety.

Although the scope of the research will be limited to the laboratory environment, we will strive to achieve the reliability and efficiency of the vehicle's steering and speed control system under carefully simulated and tested conditions. The results of this study can provide important information for developing and improving future driving automation systems.

1.3. The methodology of the project

The methodology for developing an Autonomous Car focused on line following using Arduino and Python programming involves a structured approach to system design, hardware integration, and software development. Initially, a comprehensive requirement analysis is conducted to define the project objectives, level of autonomy desired, and hardware constraints. Following this, the system architecture is designed, delineating the integration of Arduino microcontrollers for low-level control and Python for higher-level logic and decision-making.

Hardware components, including Arduino boards, motor drivers, and line-detecting sensors, are carefully selected and integrated into a functional prototype. Arduino programming involves implementing basic control functionalities and interfacing with sensors, while Python programming focuses on receiving sensor data, executing the line-following algorithm, and coordinating overall vehicle behavior. Integration and testing phases ensure the seamless interaction between Arduino and Python components, while documentation and presentations communicate the project's methodologies, achievements, and outcomes effectively.

Throughout the process, iterative development and collaboration among team members are emphasized to address challenges, optimize performance, and achieve project objectives efficiently.

1.4. The task assignment

<i>Member</i>	<i>Tasks</i>
Lê Hồ Minh Khoa (leader)	Research and write Chapter 1, 2 and 4 Prepare the microcontroller boards Bring tools and electronic components Programming control systems and sensors Prepare the Powerpoint Present the research
Nguyễn Đăng Quốc Khánh	Research and write Chapter 2 and 5 Bring tools and equipment Research on sensors Model testing and evaluation Present the research
Nguyễn Anh Quốc	Research and write Chapter 1, 3 and 4 Buy electronic devices and vehicle frames Calculate model design Research image processing, line detection Present the research
Nguyễn Gia Bảo	Research and write Chapter 2 and 5 Support for model assembly Model testing and evaluation Present the research

CHAPTER 2. LITERATURE REVIEW

2.1. Previous theoretical foundations on autonomous vehicles

Previous theoretical foundations on line-following autonomous vehicles have often relied on a combination of principles from control theory, computer vision, and robotics. In the context of Python and Arduino, these foundations typically involve algorithms for real-time image processing to detect and track the line using techniques such as color segmentation, edge detection, or template matching. Control strategies such as proportional-integral-derivative (PID) control are commonly employed to regulate the vehicle's speed and direction based on the detected error between the desired path and the actual position of the line. Additionally, the integration of Arduino microcontrollers allows for interfacing with sensors and actuators, facilitating the implementation of feedback loops to adjust the vehicle's behavior in response to environmental changes or disturbances. Overall, the convergence of Python for high-level programming and Arduino for low-level hardware control offers a versatile platform for developing efficient and robust line-following autonomous systems.

2.2. The global state of autonomous vehicles

As of the latest reports, autonomous vehicles continue to advance at a rapid pace, showcasing significant strides in both technology and application. Across various industries including logistics, manufacturing, and transportation, these vehicles have become integral for automating repetitive tasks and enhancing efficiency. Advanced sensors, such as lidar and computer vision systems, enable precise navigation and obstacle detection, facilitating seamless movement even in complex environments. Moreover, developments in artificial intelligence and machine learning algorithms have empowered these vehicles to adapt to dynamic surroundings and optimize their routes for maximum productivity. With ongoing research and innovation, the global landscape for line-following self-propelled

vehicles promises continued growth, offering increasingly sophisticated solutions for diverse operational challenges.

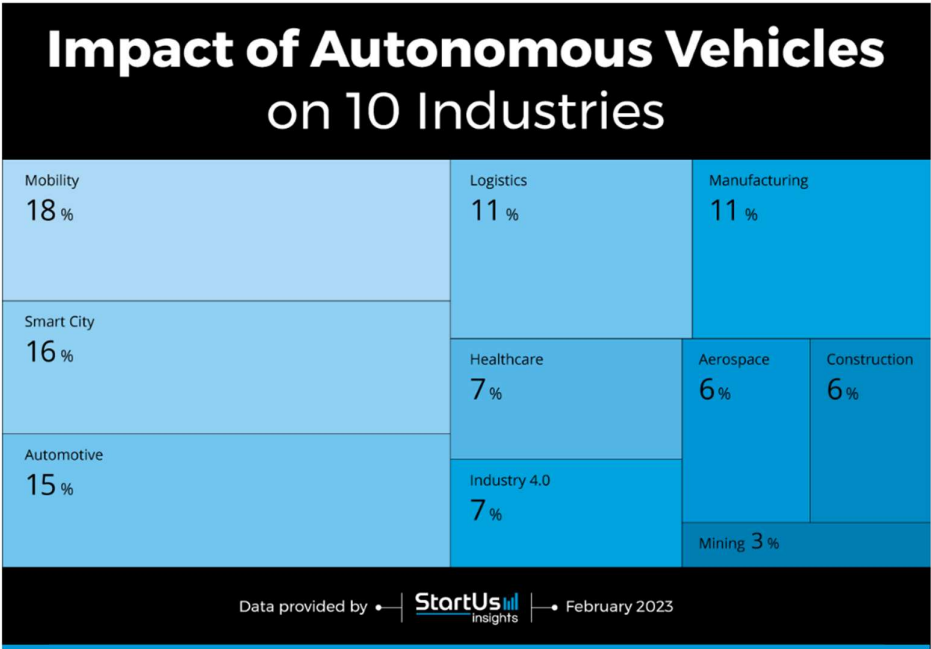


Figure 2.1. Top autonomous vehicle applications across 10 industries in 2023

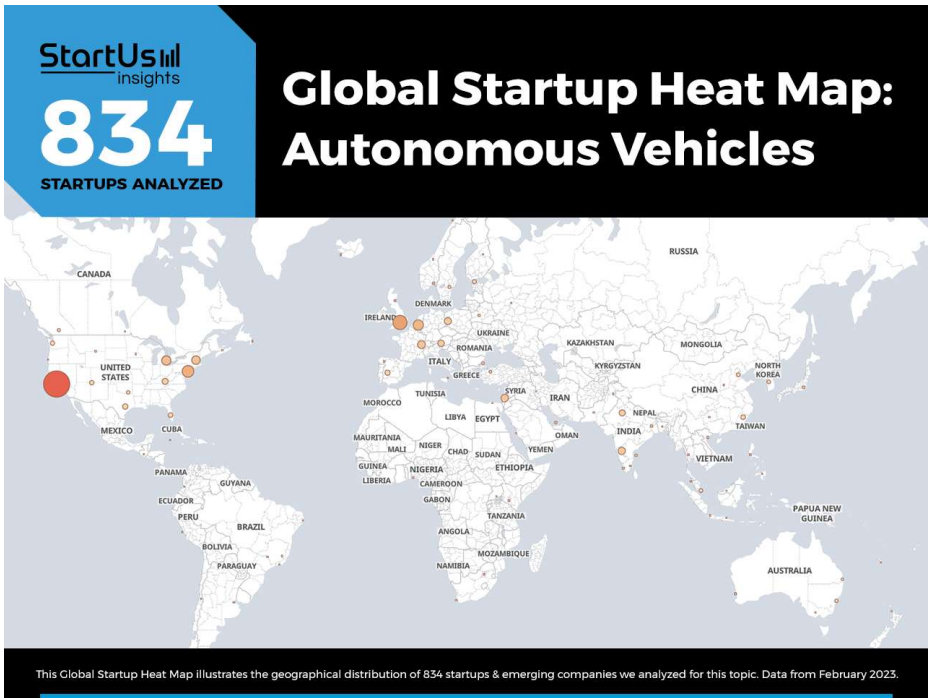


Figure 2.2. Global Startup Heat Map of Autonomous Vehicles

2.3. The challenges of autonomous vehicles

Autonomous vehicles face the intricate challenge of maintaining precise navigation along designated paths while dynamically responding to changing environmental conditions. The primary hurdle lies in developing algorithms that enable the vehicle to interpret sensor data effectively, making rapid decisions to adjust speed and direction accordingly. Furthermore, these vehicles must contend with variations in line thickness, color, and ambient lighting, necessitating robust sensor fusion techniques and adaptive control mechanisms. Additionally, the design must balance the trade-offs between agility and stability, ensuring the vehicle can swiftly navigate sharp turns without sacrificing accuracy or risking destabilization. Overcoming these challenges requires a multidisciplinary approach that integrates advancements in computer vision, machine learning, and robotics, ultimately culminating in autonomous systems capable of reliably traversing complex terrains with minimal human intervention.

2.4. Software applications for autonomous vehicles

To build our autonomous vehicle, we used 2 main software applications which are Arduino IDE and Python because they offer complementary strengths in hardware interfacing and high-level programming, allowing us to efficiently manage both low-level control systems and complex decision-making algorithms. Python, on a connected computer, acts as the brain, handling high-level control and decision-making. Python employs algorithms like PID control to calculate the necessary motor adjustments. Python then transmits these commands back to the Arduino, which translates them into real-time motor actions, steering the vehicle along the lane. This collaborative approach leverages Python's strength in data analysis and complex logic, while capitalizing on Arduino's proficiency in real-time communication and low-level motor control.

CHAPTER 3. THE PROCESS OF CREATING AUTONOMOUS VEHICLE

3.1. Algorithms of autonomous vehicle

3.1.1. Lane detection algorithm of autonomous vehicle

In terms of image processing, Python libraries like OpenCV can be employed to process the video feed from the webcam. Techniques such as edge detection (e.g., Canny edge detector), color thresholding, and morphological operations can be used to identify lane markings. Once the lanes are detected, algorithms like Hough Transform can be utilized to extract lines representing the lanes from the processed image.

3.1.2. Control algorithm of autonomous vehicle

Here we used PID Control (Proportional-Integral-Derivative), which PID controllers can be implemented in Arduino to regulate vehicle steering based on the deviation of detected lane positions from the desired trajectory. Algorithm for maintaining a consistent speed can be implemented, possibly considering inputs like distance to the vehicle ahead or the desired speed set by the user.

3.1.3. Decision making algorithm of autonomous vehicle

In the behavioral algorithms, Python code can incorporate decision-making algorithms that determine actions based on the lane detection results and other sensor inputs. This could involve strategies for lane changing, obstacle avoidance, and following traffic rules. Algorithms for planning a safe and efficient path to the destination, considering factors like lane changes, traffic conditions, and obstacles.


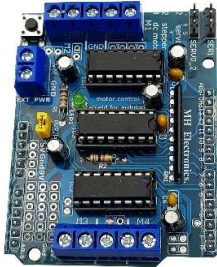

3.1.4. Integration algorithm of autonomous vehicle

In serial communication, implementing communication protocols like UART between the Arduino and Python code enables seamless data exchange. Arduino can send sensor data such as steering angle and vehicle speed to Python, while Python can send control commands back to Arduino. This can ensure that synchronization between the control loop running on Arduino and the decision-making process in Python is crucial for real-time responsiveness.

3.2. The designing process of the autonomous vehicle

3.2.1. The components preparing process

In order to successfully construct and assemble this autonomous vehicle, our team made various research, and purchased the essential components. The list below includes all of the necessary components.

Component name	Component illustration	Quantity
Arduino Uno		1
L293D Shield		1
Lithium battery case		1

Lithium batteries		2
Arduino - USB cable		1
Dual Shaft Plastic Geared Motor		4
Plastic Wheel 65mm		4
Acrylic chassis frame		2
Plastic wheel - motor mount plates		4

Brass standoff		6
Nuts		10
8mm bolts		10
ON/OFF Switch		1
Male – male Wires		10
Logitech C920e webcam		1

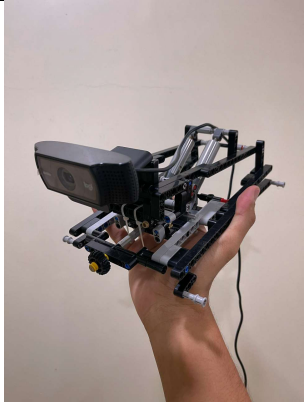
Lego Technic Camera frame		1
---------------------------	--	---

Figure 3.1. Table of components

3.2.2. The chassis assemble process

First, as part of the initial phase, we built the chassis by adding standoffs and fastened the nuts to the acrylic chassis frame. Subsequently, we connected 4 motors into the frame by using 4 plastic wheel – motor mount plates to ensure optimal stability and precise alignment within the overall structure. Then we added wheels to the motor, ensuring a secure and reliable connection.

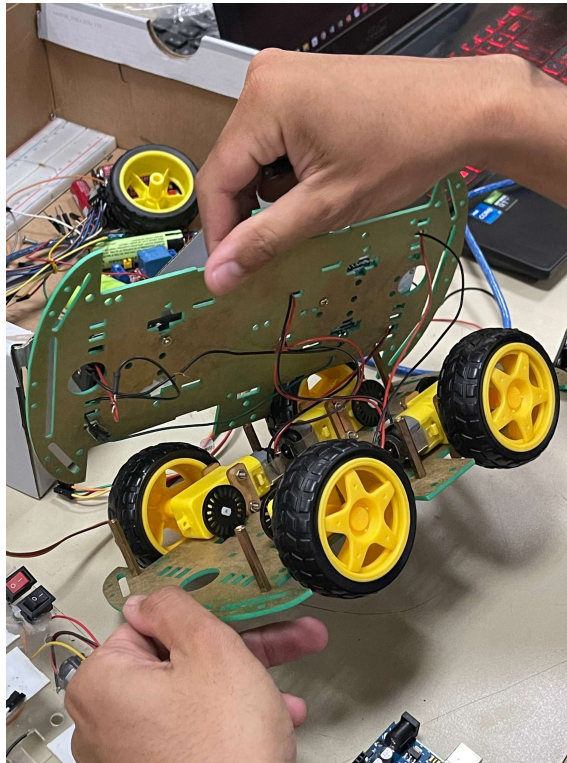


Figure 3.2. Our vehicle chassis in the building process

In the second stage, we attached the Arduino UNO onto the vehicle and connected it with motors. The motors 1, 2, 3 and 4 were linked into the Arduino UNO unit through gates M1, M2, M3 and M4 respectively. Thereby forging an intricate network of electrical pathways that facilitate the seamless transmission of commands and instructions from the microcontroller to the motors.

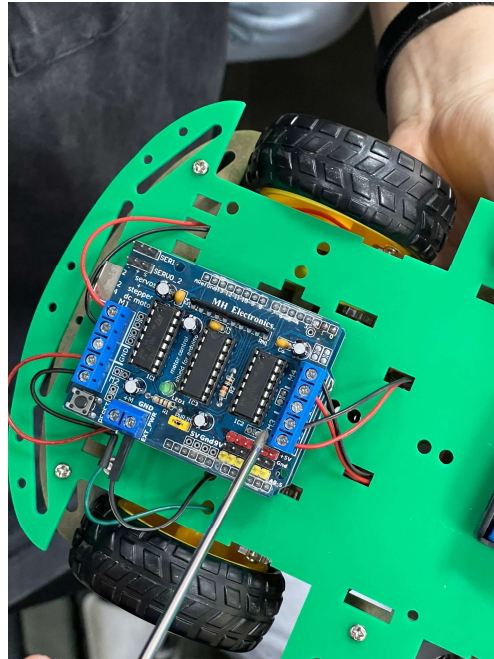


Figure 3.3. Connecting the motor wires into the Arduino unit

In the third stage, we add the switch on the frame and connect it into the Arduino unit we established a vital electrical connection by linking the switch to the Arduino unit, thereby enabling convenient control and activation of the entire system. In addition, we put the Lithium batteries and battery case in the upper part of the chassis frame. With careful consideration of weight distribution and optimal utilization of available space, we securely positioned the Lithium batteries within a purpose-built battery case.

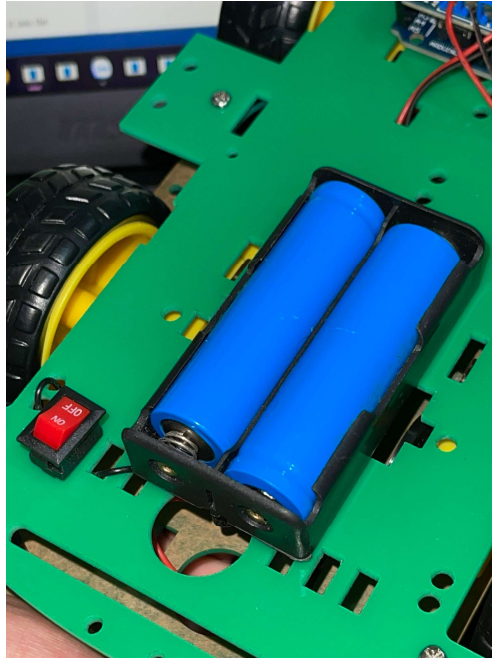


Figure 3.4. Adding the switch and batteries onto the chassis

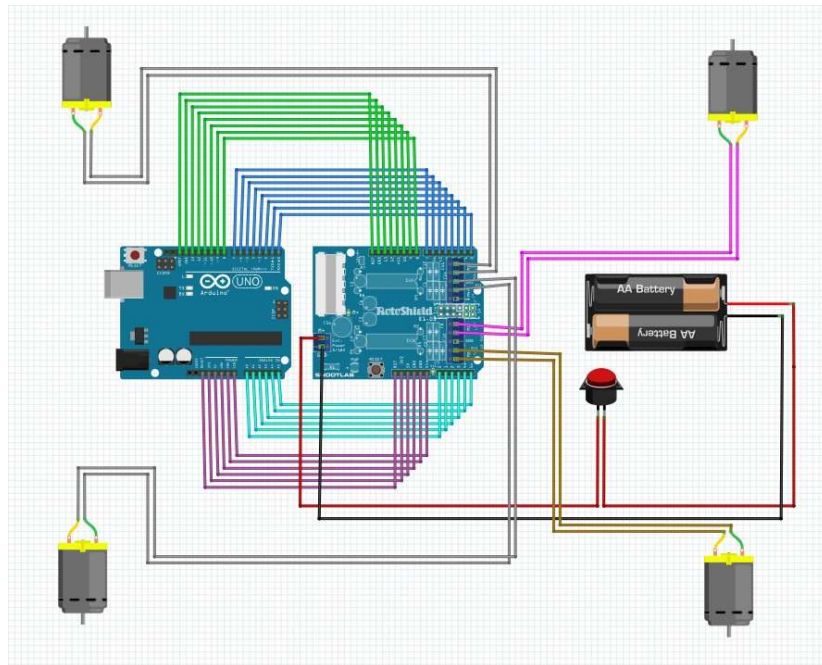


Figure 3.5. The wiring diagram of the chassis

3.2.3. The camera frame adding process

To address the challenge posed by the low chassis obstructing the camera's view, we implemented a solution involving the addition of a frame. Recognizing

the need for both durability and adjustability, we opted for a construction method utilizing Lego Technic components. Leveraging Lego Technic gears, axles, and functional pins, we engineered a robust frame capable of providing the necessary elevation and stability for optimal camera performance. This approach not only ensured the camera's high viewpoint and improved vision but also facilitated easy adjustment, allowing for fine-tuning as needed. By harnessing the versatility and reliability of Lego Technic elements, we achieved a practical and efficient solution that seamlessly integrated with our existing setup, enhancing the overall functionality of the vehicle.

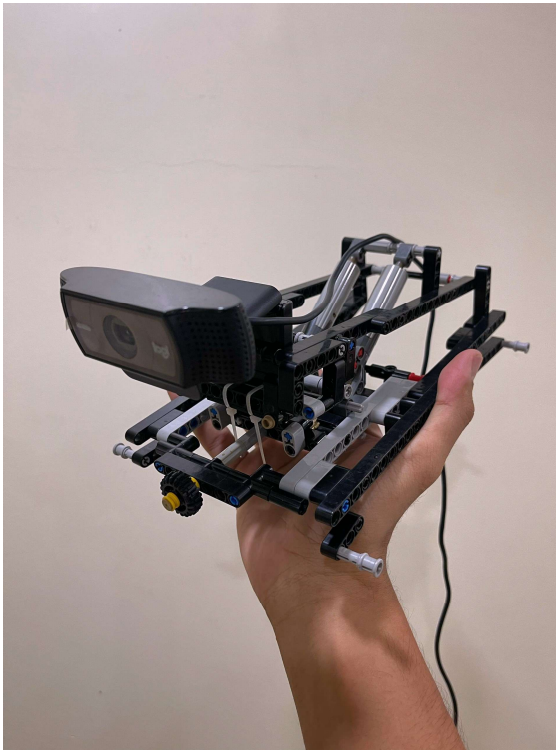


Figure 3.6. Lego Technic Camera frame

3.2.4. The camera attaching process

In order to support the vehicle's capabilities in executing intricate image processing tasks, we have thoughtfully integrated a cutting-edge Logitech C920e webcam onto the front section of the vehicle. This high-performance camera, renowned for its exceptional image quality and advanced functionalities.

To facilitate the seamless transmission of visual data captured by the camera, we have established a reliable connection between the camera and our laptop. By using the camera's capabilities and the computing power of our laptop, we process the captured data efficiently.

The laptop, acting as the central hub for data processing, establishes a vital connection with the Arduino UNO residing within the vehicle. This pivotal connection enables the exchange of processed data, commands, and instructions between the laptop and the Arduino UNO, thereby empowering the vehicle with the intelligence and decision-making capabilities necessary for executing complex tasks based on the analyzed visual data.



Figure 3.7. Our webcam Logitech C920e

After meticulously combining all the components, we have reached the culmination of our efforts. With every component meticulously assembled and every system meticulously calibrated, we stand ready to unleash the full potential of our creation. This milestone represents not only the realization of our initial

vision but also the beginning of a new chapter, where our vehicle's capabilities will be put to the test in real-world applications and challenges.

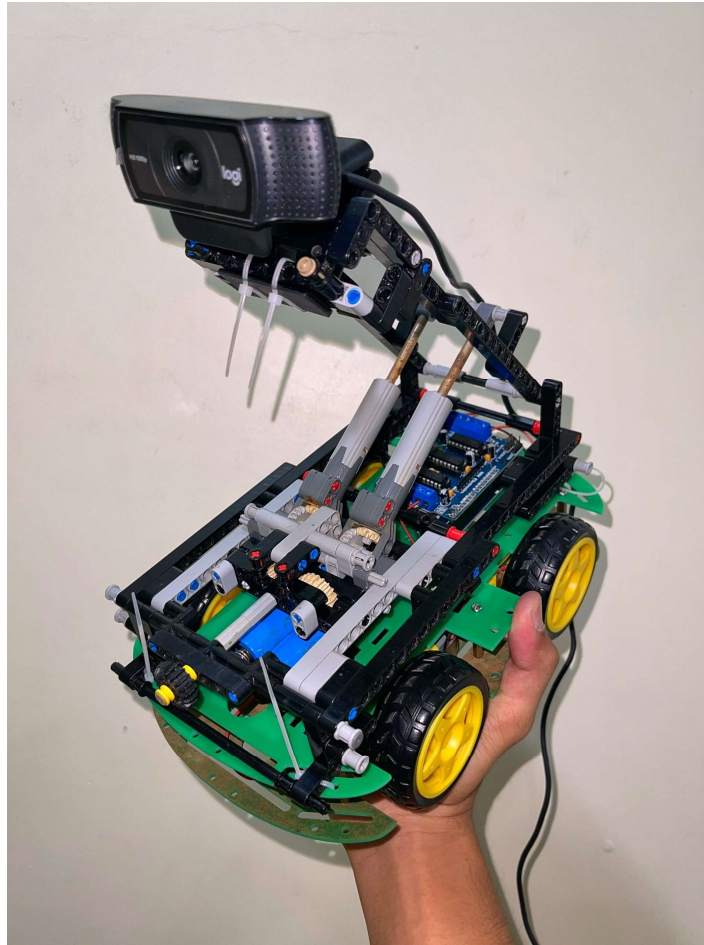


Figure 3.8. The final version of our autonomous vehicle

3.3. Programming process of autonomous vehicle

3.3.1. Building the Python code

```
#VEHICLE AUTOMATIC CONTROL SYSTEM  
#VACS330333E_23_2_01FIE-HCMUTE  
#GROUP_4  
import cv2  
import numpy as np  
import math  
import serial
```

```

import time

threshold1 = 90 #Reduce Canny threshold to enhance edge detection
threshold2 = 150 #Reduce Canny threshold to enhance edge detection
theta = 0
r_width = 640 #Reduce frame size
r_height = 480 #Reduce frame size
minLineLength = 100
maxLineGap = 10
k_width = 5
k_height = 5
max_slider = 10

def region_of_interest(edges):
    height = frame.shape[0]
    width = frame.shape[1]
    mask = np.zeros_like(edges)
    triangle = np.array([[
        (0,height-5), #Left_Bottom
        (width*(0.20), height*(0.4)),#Left_Top
        (width*(0.67), height*(0.4)),#Right_Top
        (width, height-5 ),]], np.int32) #Right_Bottom
    cv2.fillPoly(mask, triangle, 255)
    masked_image = cv2.bitwise_and(edges, mask)
    return masked_image

frame_count = 0 #Frame count variable
send_count = 0 #Counter variable sends data

```

```

cap = cv2.VideoCapture(0)
ser = serial.Serial('COM7', 115200)

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

    frame_count += 1
    if frame_count % 3 != 0: #Only process frames every 3 times
        continue

    # Resize width=400 height=225
    frame = cv2.resize(frame, (r_width, r_height))

    # Convert the image to gray-scale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur
    blurred = cv2.GaussianBlur(gray, (k_width, k_height), 0)

    # Find the edges in the ROI using Canny detector
    edged = cv2.Canny(blurred, threshold1, threshold2, apertureSize = 3)

    #Limited region
    crop_canny=region_of_interest(edged)

    # Detect points that form a line

```

```
lines = cv2.HoughLinesP(crop_canny, 1, np.pi/180, max_slider,
minLineLength, maxLineGap)
```

```
if lines is not None:
```

```
    for line in lines:
```

```
        for x1, y1, x2, y2 in line:
```

```
            angle = np.arctan2(y2 - y1, x2 - x1) * 180 / np.pi #Calculate the
angle of the line
```

```
            if abs(angle) < 45: #Remove lines that are close to horizontal
```

```
                continue
```

```
            cv2.line(frame, (x1, y1), (x2, y2), (255, 255, 0), 3) #Redraw the
line on the original frame
```

```
            theta += math.atan2((y2 - y1), (x2 - x1))
```

```
threshold = 5
```

```
if theta > threshold:
```

```
    print("Go left")
```

```
    send_count += 1
```

```
    if send_count % 3 == 0: #Send data every 3 times
```

```
        ser.write(b'L') #Send left signal to Arduino
```

```
elif theta < -threshold:
```

```
    print("Go right")
```

```
    send_count += 1
```

```
    if send_count % 3 == 0: #Send data every 3 times
```

```
        ser.write(b'R') #Send signal right to Arduino
```

```
else:
```

```
    print("Go straight")
```

```
    send_count += 1
```

```
    if send_count % 3 == 0: #Send data every 3 times
```



```
ser.write(b'S') #Send stop signal to Arduino
```

```
theta = 0
```

```
cv2.imshow("Frame", frame)
```

```
cv2.imshow("Edged", edged)
```

```
cv2.imshow('cropped pic',crop_canny)
```

```
if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
    break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

The code begins by importing necessary libraries such as OpenCV for image processing, NumPy for numerical operations, Math for mathematical calculations, Serial for serial communication, and Time for time-related functions. These libraries provide essential tools handling image data, performing mathematical computations, establishing communication with external devices.

Following the imports, the code initializes various parameters that will be used throughout the program. These parameters include thresholds for edge detection, frame size, line detection parameters, and others. Proper parameter initialization is crucial for fine-tuning the performance of the computer vision algorithms used in the subsequent steps.

A key function defined in the code is “region_of_interest”, which creates a mask to focus on a specific area of interest in the frame, presumably the area where the road is located. This function helps in reducing the computational load by limiting the processing to relevant parts of the image, enhancing the efficiency of subsequent operations.

The main loop of the program continuously reads frames from the video capture device, downsizes them for faster processing, converts them to grayscale, and applies Gaussian blur to reduce noise. It then detects edges in the region of interest using the Canny edge detector and applies the Hough transform to detect lines in the edges. By filtering out horizontal lines and calculating the average angle of the remaining lines, the program determines the appropriate steering action for the vehicle.

Based on the calculated angle, the program decides whether the vehicle should go left, right, or straight and sends corresponding control signals ('L' for left, 'R' for right, 'S' for straight) to an Arduino connected via serial communication. This mechanism allows the vehicle to autonomously navigate based on the detected lane's angle, making it suitable for applications such as lane-keeping assistance or autonomous driving.

Finally, the program cleans up by releasing the video capture object and closing all OpenCV windows. This ensures proper resource management and terminates the execution of the program in an organized manner. Overall, the code implements a basic automatic control system for a vehicle using computer vision techniques, enabling it to detect lanes in real-time video feed and adjust its steering accordingly for autonomous navigation.

3.3.2. Building the Arduino code

```
#include <AFMotor.h>

AF_DCMotor motor1(1); // Declare motor1 at pin 1 of Shield
AF_DCMotor motor2(2); // Declare motor2 at pin 2 of Shield
AF_DCMotor motor3(3); // Declare motor3 at pin 3 of Shield
AF_DCMotor motor4(4); // Declare motor4 at pin 4 of Shield

bool goLeftActive = false;
bool goRightActive = false;
int goLeftCount = 0;
int goRightCount = 0;

void setup() {
  Serial.begin(115200); // Start Serial communication with a baud rate of
115200
  // Start the motors
  motor1.setSpeed(255); // Set speed (range from 0 to 255)
  motor2.setSpeed(255);
  motor3.setSpeed(255);
  motor4.setSpeed(255);
}

void loop() {
  if (Serial.available() > 0) {
    // Read data from Serial
    char command = Serial.read();
    // Control motors based on data received from Python
    if (command == 'S') {
      // Run all motors at maximum speed, but the direction of rotation is
BACKWARD instead of FORWARD
      motor1.run(BACKWARD);
      motor2.run(BACKWARD);
      motor3.run(BACKWARD);
      motor4.run(BACKWARD);
      // Reset counter and status variables
```

```

    goLeftActive = false;
    goRightActive = false;
    goLeftCount = 0;
    goRightCount = 0;
} else if (command == 'F') {
    // Run all motors at maximum speed
    motor1.run(FORWARD);
    motor2.run(FORWARD);
    motor3.run(FORWARD);
    motor4.run(FORWARD);
    // Reset counter and status variables
    goLeftActive = false;
    goRightActive = false;
    goLeftCount = 0;
    goRightCount = 0;
} else if (command == 'L') {
    // Pause motor 1 and motor 3 if not already paused
    if (!goLeftActive) {
        motor1.run(RELEASE);
        motor3.run(RELEASE);
        goLeftActive = true;
    }
    goLeftCount++;
} else if (command == 'R') {
    // Pause motor 2 and motor 4 if not already paused
    if (!goRightActive) {
        motor2.run(RELEASE);
        motor4.run(RELEASE);
        goRightActive = true;
    }
    goRightCount++;
}

// Check if Python has stopped sending the "Go left" signal
if (goLeftActive && command != 'L') {
    goLeftActive = false;
    // Reactivate motor 1 and motor 3 if the "Go left" signal has been
received enough times
    if (goLeftCount >= 3) {
        motor1.run(FORWARD);
        motor3.run(FORWARD);
        goLeftCount = 0;
    }
}

// Check if Python has stopped sending the "Go right" signal
if (goRightActive && command != 'R') {
    goRightActive = false;

```

```

    // Reactivate motor 2 and motor 4 if the "Go right" signal has been
    received enough times.
    if (goRightCount >= 3) {
        motor2.run(FORWARD);
        motor4.run(FORWARD);
        goRightCount = 0;
    }
}
}
}
}

```

This Arduino code enables the control of a vehicle's movement using commands sent over serial communication from a Python script. It initializes four DC motors connected to specific pins on an Adafruit Motor Shield and sets their speed to maximum. The main loop continuously listens for incoming commands, interpreting each command to dictate the vehicle's behavior. Commands include stopping all motors ('S'), moving forward ('F'), turning left ('L'), and turning right ('R'). Upon receiving a turn command, the code selectively releases the appropriate motors to enable turning, keeping track of how many times each turn command is received to ensure the vehicle completes the turn adequately before resuming straight movement.

This code implements a robust control system for a vehicle, allowing for precise movement control through straightforward serial commands. By interfacing with the Adafruit Motor Shield, it translates incoming commands into motor actions, enabling functionalities such as forward motion, stopping, and turning. This versatility makes it suitable for various applications requiring remote or automated vehicle navigation, offering a foundation for more complex control strategies and autonomous operation.

CHAPTER 4. AUTONOMOUS VEHICLE IN PRACTICAL EXPERIMENTATION

4.1. Testing the vehicle code

4.1.1. Testing the Python code

Reaching the final version of our code involved navigating through multiple setbacks and failures. However, despite these challenges, we persisted and ultimately succeeded in completing it within the designated timeframe. Leveraging Python as our programming language of choice, we orchestrated the code to execute seamlessly, albeit with an operational time of approximately 1.5 minutes post-activation.

During runtime, the interface presented 4 distinct windows on the screen: the Frame window, providing real-time camera feed visualization; the Edged window, displaying the detected edges extracted from the video capture; the Cropped Pic window, showcasing the filtered edges representing the detected lanes; and the Python IDE Shell window, serving as the conduit for issuing commands to the Arduino module, thereby orchestrating the vehicle's movements and interactions.

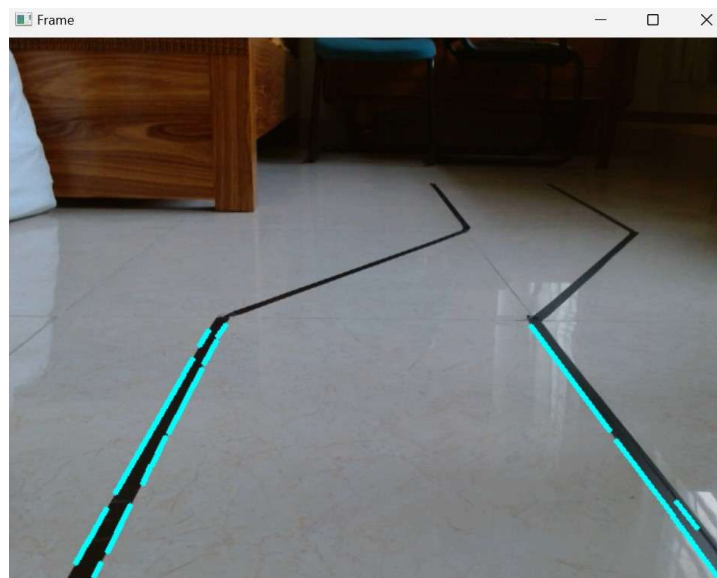


Figure 4.1. The Frame window



Figure 4.2. The Edge window

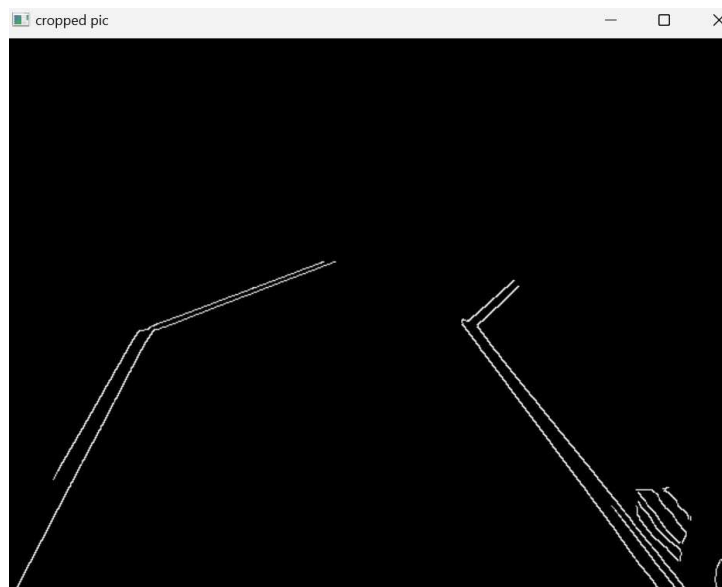


Figure 4.3. The Cropped Pic window

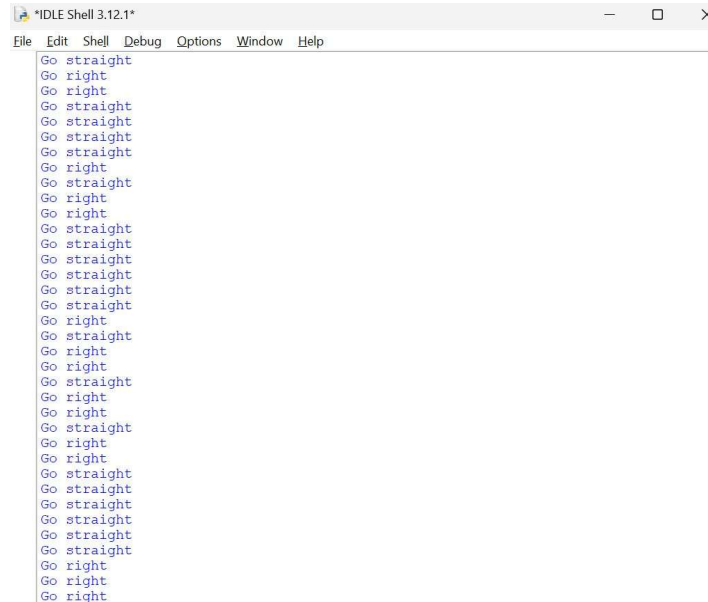


Figure 4.4. The Python IDE Shell window

4.1.2. Testing the Arduino code

Developing the code consumed over a week of dedicated effort before achieving its final form. Upon integration with the vehicle, the code demonstrated remarkable effectiveness, facilitating smooth operation throughout various maneuvers. Notably, during turning maneuvers, the vehicle's steering mechanism operated according to the principles of compass theory, a technique ensuring precise handling around bends. This approach involved halting the rotation of the inner wheels when deemed necessary, optimizing the vehicle's ability to navigate sharp turns and negotiate challenging corners with enhanced stability and control.

4.2. Experimenting the vehicle in practical

The process began with the setup of a basic lane using back tape, providing a clear visual marker for the vehicle's detection system. Placing the vehicle in front of the lane, we meticulously adjusted the webcam's height and angle to ensure optimal data capture, vital for accurate lane detection. With the camera cable and Arduino cable connected to the laptop, we initiated the vehicle's operation, initiating a series of testing runs to record performance metrics.



Figure 4.5. Our tape lane for vehicle testing

Through iterative testing and refinement, we scrutinized the results, identifying errors and inconsistencies to address. This iterative process involved repeated runs, each revealing new insights and opportunities for improvement. By diligently analyzing each test run and iteratively adjusting parameters, we steadily honed the system's functionality until achieving the desired practical results. This meticulous approach to testing and refinement was crucial in fine-tuning the system's performance, ultimately culminating in a reliable and effective lane detection system ready.

CHAPTER 5. CONCLUSION

5.1. Summarizing the accomplishments

In achieving our project goals, we've completed pivotal tasks such as chassis assembly, meticulous wiring connections, Python code development utilizing OpenCV for webcam integration, and Arduino programming for vehicle control. The successful execution of our vehicle's functionality, particularly its ability to detect lanes and navigate paths autonomously, marks a significant milestone, showcasing the integration of mechanical, electrical, and software components into a cohesive system. Yet, beyond these tangible achievements, the project has served as a rich learning experience, deepening our understanding of robotics and automation. Each challenge overcome and problem solved has contributed to our growth as engineers, highlighting the value of hands-on experimentation and interdisciplinary collaboration. Ultimately, while the project's outcomes are notable, it is the knowledge and skills gained throughout the process that underscore its true success.

5.2. Forward vision and future plans

Looking ahead, our focus extends to enhancing and refining our lane-detecting vehicle. Building upon the foundation established by this project, we aim to delve deeper into its capabilities, exploring avenues for optimization and performance enhancement. Additionally, we aspire to venture into new realms of autonomous vehicle systems, integrating advanced technologies such as Lidar, radar, ultrasonic sensors, GPS, IMU, and wheel encoders. These cutting-edge technologies promise to elevate the vehicle's capabilities, enabling more robust and precise navigation in diverse environments. Furthermore, we envision incorporating solar panels into the vehicle's design, aligning with our commitment to sustainability and environmental responsibility. By harnessing renewable

energy sources, we aim to reduce our ecological footprint while enhancing the vehicle's autonomy and longevity. Overall, our ambitions extend beyond the confines of this project, driving us towards innovation and excellence in the realm of autonomous vehicles.

REFERENCES

- [1] Autonomous Vehicle Market Size, Share & Covid 19 Impact Analysis, By Level (L1, L2, & L3 and L4 & L5), By Vehicle Type (Passenger Cars and Commercial Vehicles), and Regional Forecast, 2023 – 203, Fortune Business Insights (updated April 29, 2024)
- [2] Top 10 Applications of Autonomous Vehicles in 2023 & 2024, StarUs insights
- [3] Car Vision: Lane detection and Following, Chalmers University of Technology, Aditya Subramanian, Bachelor's Thesis in Electrical Engineering, Gothenburg, Sweden 2017
- [4] Implementation of lane detection algorithm for self-driving car on toll road cipularang using Python language, Mochamad Vicky Ghani Aziz, Bandung Institute of Technology, October 2017
- [5] Designing Autonomous Car using OpenCV and Machine Learning, International Research Journal of Engineering and Technology (IRJET), R. Hemanth Kumar, T. Varun Sai Krishna, S. Durga Prasad, P. Varuna Sai, 04, Apr 2023
- [6] Driving DC Motors with L293D and Arduino, Robocraze, India Most Trusted Robotics and DIY Store, October 2017
- [7] ArtiBot: A Bluetooth-Controlled Drawing Robot Car Powered by Arduino, Journal of Emerging Technologies and Innovative Research, Aaraisha Zehra, Kratika Jain, Amrah Maryam, Savita Gautam, Salma Shaheen, Volume 10 Issue 9 September-2023