
TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE
AUTONOMOUS CAR PROJECT
VEHICLE AUTOMATIC CONTROL SYSTEM
Implementation: Group 4
Guiding lecturer: PhD. Lê Thanh Phúc

OUR TEAM MEMBERS

Le Ho Minh Khoa
ID: 21145019
(Leader)

Nguyen Dang Quoc Khanh
ID: 21145584

Nguyen Anh Quoc
ID: 21145606

Nguyen Gia Bao
ID: 21145576

MAIN CONTENTS

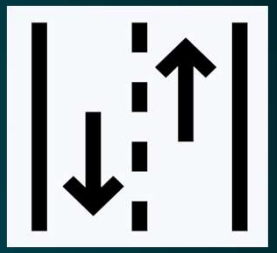
01

OVERVEIW

1.1. The reason for choosing this topic

Technology Advancement

Enhance Road Safety



Optimize Traffic Flow

Increase Mobility

Reduce Carbon Emissions

1.2.1.The objective of the project

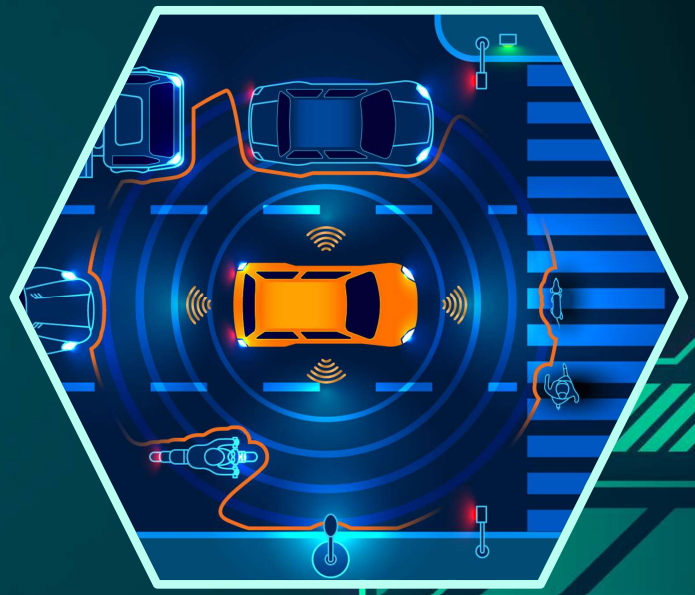
Efficiency

Safety

Advanced Sensors

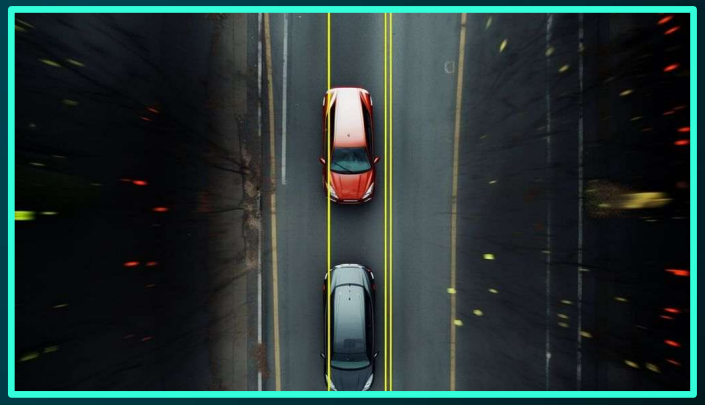
Navigation

Public Trust



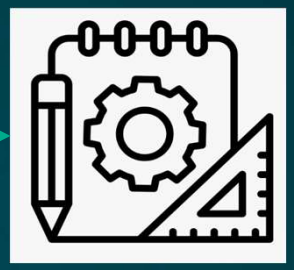
1.2.2. The scope of the project

- Develop autonomous driving and speed control in lab settings
- Focus on basic functions: line detection and road tracking
- Research algorithms for automatic line recognition and tracking
- Develop control systems for lane adherence and safety maintenance
- Strive for reliability through rigorous simulation and testing

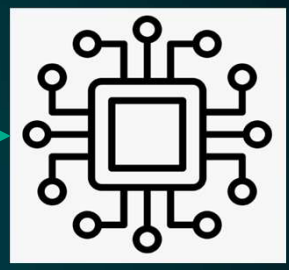


1.3. The methodology of the project

Analyze requirements



Design system



Integrate Hardware

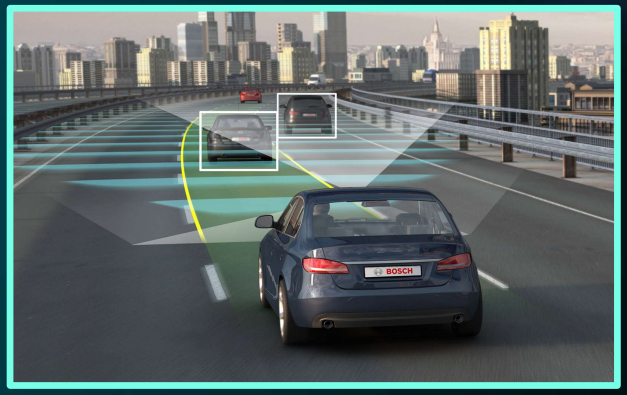
Optimize Performance

02

LITERATURE REVIEW

2.1. Previous theoretical foundations on autonomous vehicles

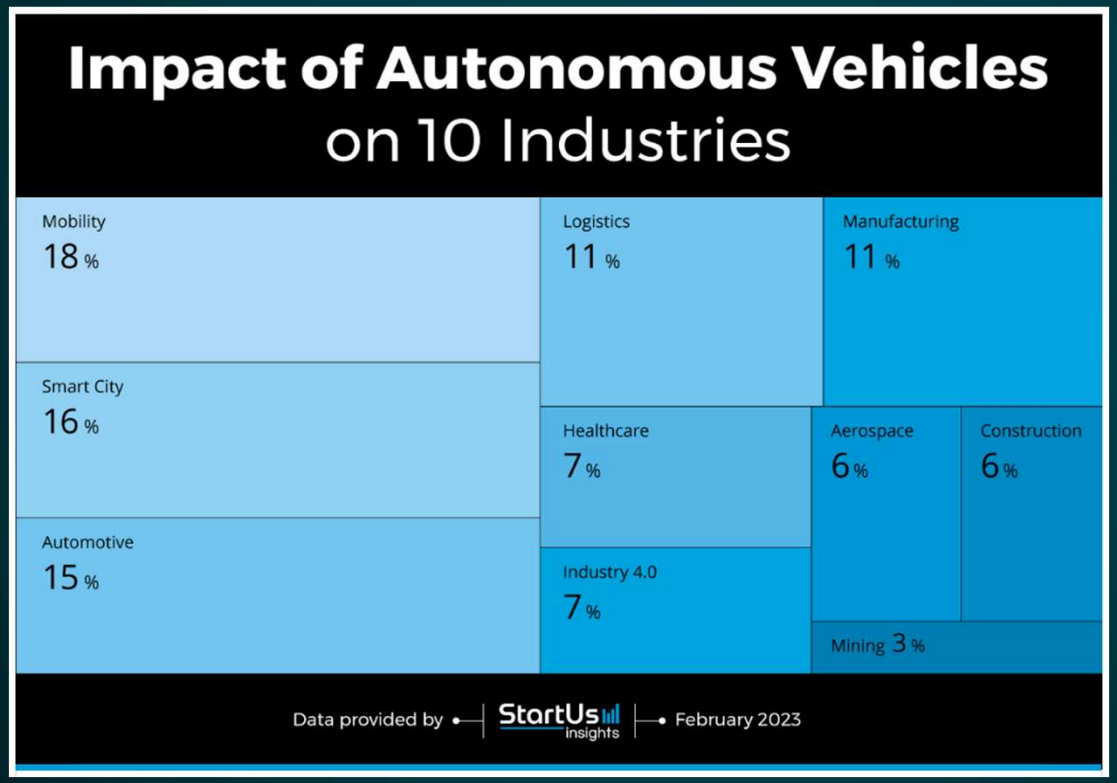
- Previous foundations combine control theory, computer vision, and robotics
- Python and Arduino use real-time image processing for line detection
- Arduino interfaces with sensors for feedback loop adjustment
- Python and Arduino offer a versatile platform for line-following systems



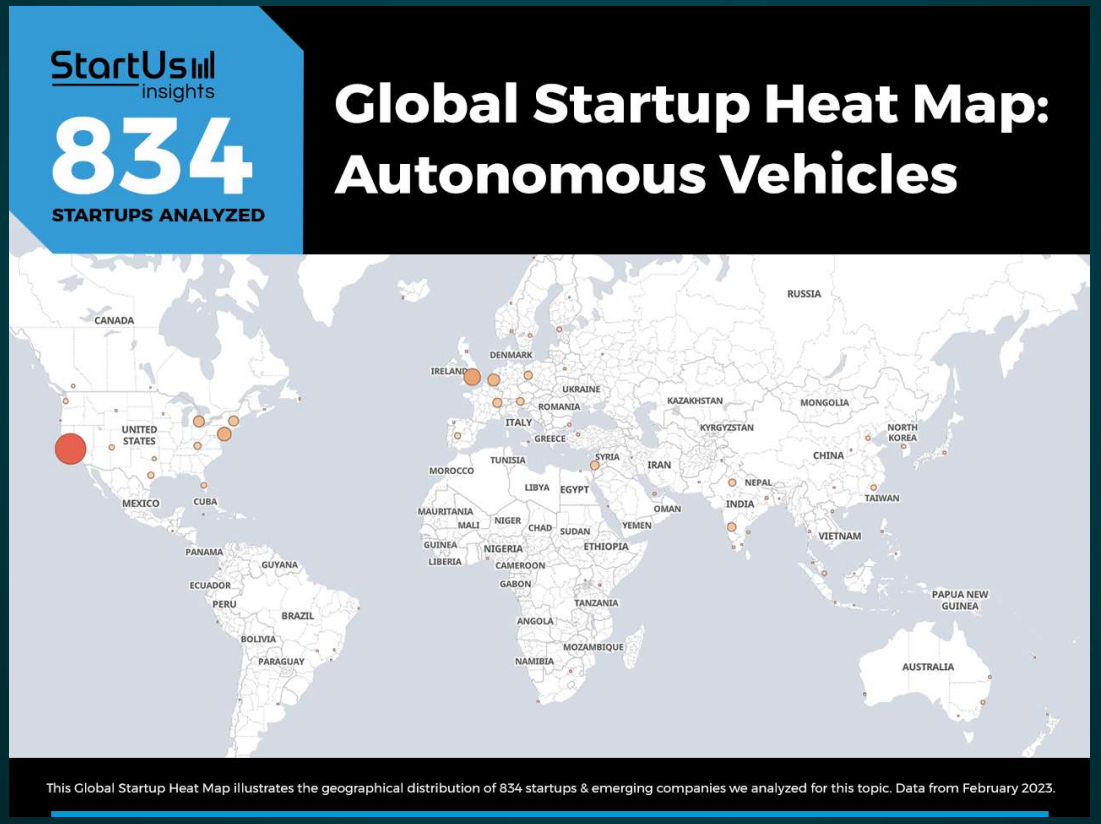
2.2. The global state of autonomous vehicles

- Autonomous vehicles advance industries.
- Advanced sensors ensure precise navigation.
- AI algorithms optimize routes dynamically.
- Reports highlight tech progress.

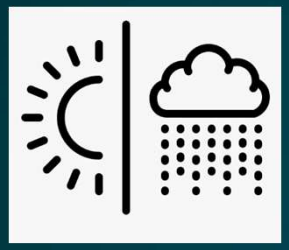
2.2. The global state of autonomous vehicles



2.2. The global state of autonomous vehicles

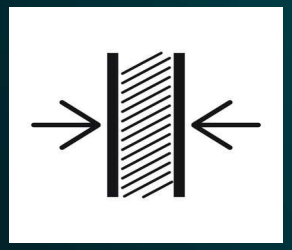


2.3. The challenges of autonomous vehicles

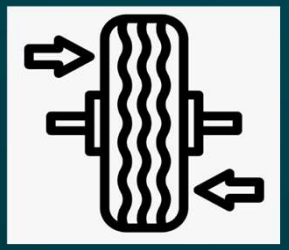


Changing Conditions

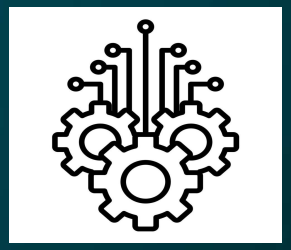
Rapid Sensor Data



Various Line Thickness



Sharp Turns Balance

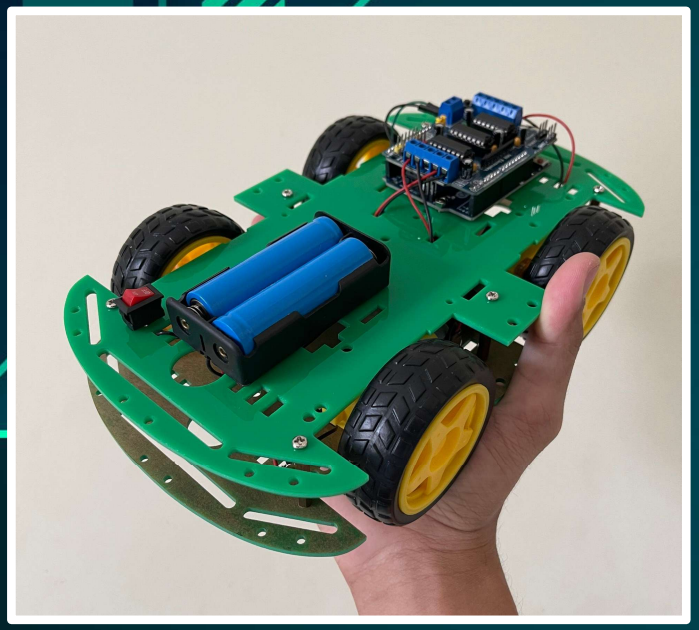


Technical Integration

2.4. Software applications for autonomous vehicles

THE PROCESS OF CREATING AUTONOMOUS VEHICLE

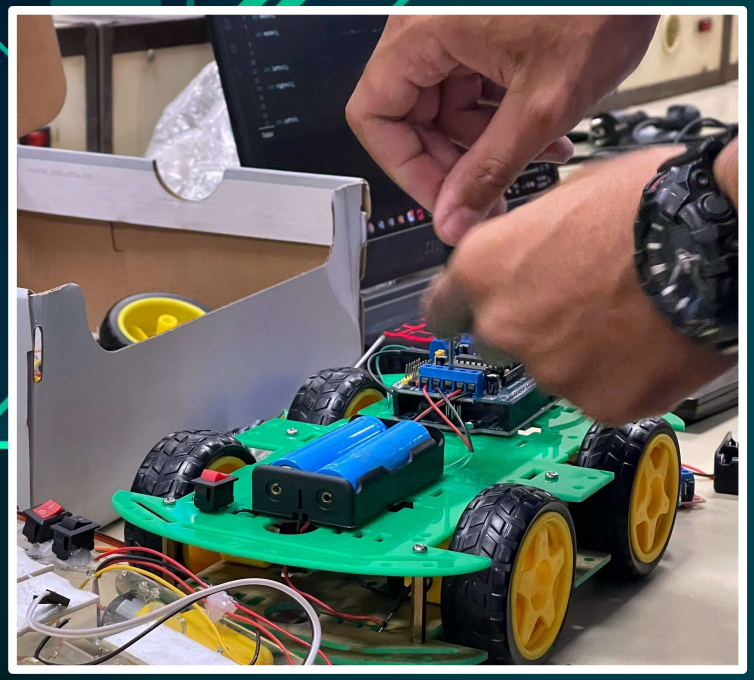
03



3.1. Algorithms of autonomous vehicle

3.1. Algorithms of autonomous vehicle

Lane detection algorithm of autonomous vehicle	Control algorithm of autonomous vehicle	Decision making algorithm of autonomous vehicle	Decision making algorithm of autonomous vehicle
Uses Python Open CV library	PID Control	Lane Detection	UART communication
Hough Transform	Speed consistency	Sensor Inputs	real-time responsiveness



3.2. The designing process of the autonomous vehicle

3.2.1. The components list

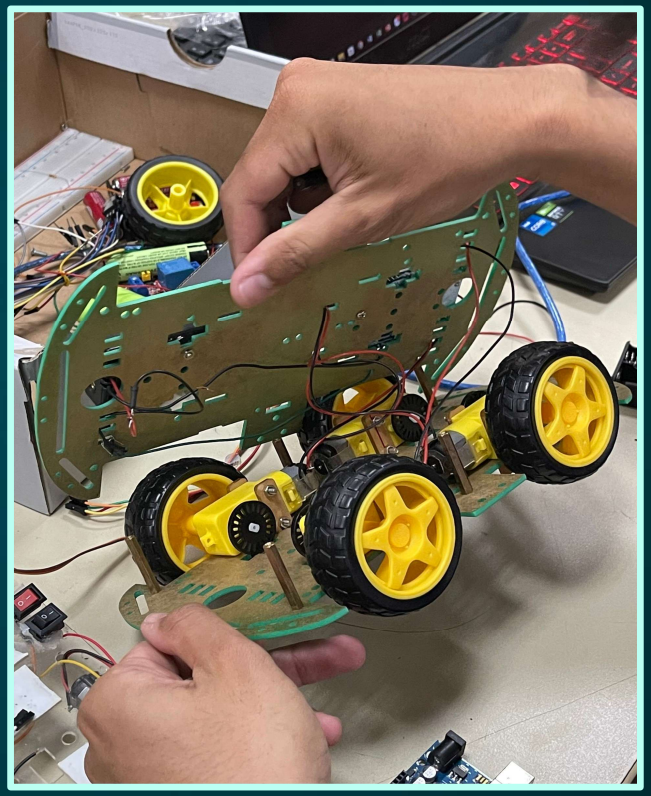
Component name	Component illustration
Arduino Uno	
L293D Shield	
Lithium battery case	
Lithium batteries	
Arduino - USB cable	
Dual Shaft Plastic Geared Motor	
Plastic Wheel 65mm	
Acrylic chassis frame	
Plastic wheel - motor mount plates	
Brass standoff	
Nuts	
8mm bolts	
ON/OFF Switch	
Male – male Wires	
Logitech C920e webcam	
Lego Technic Camera frame	

3.2.2. The chassis assemble process

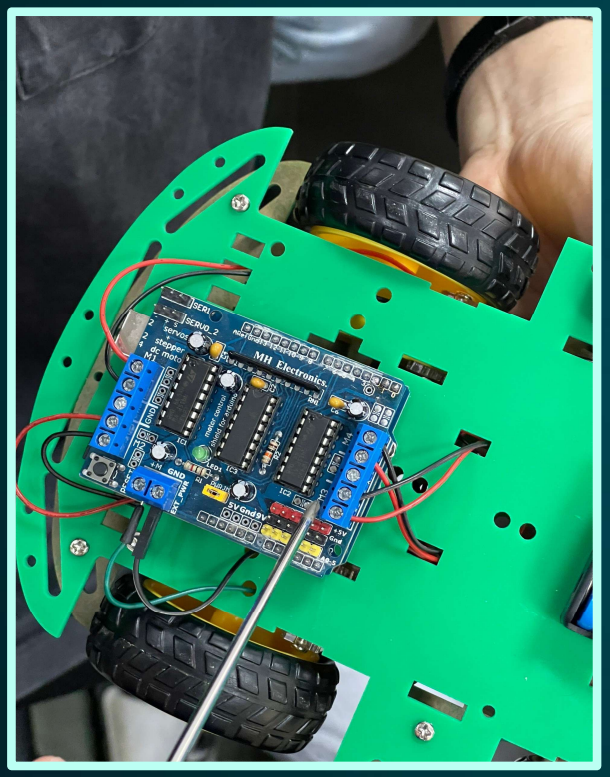
- Attached Arduino UNO and connected it with motors.

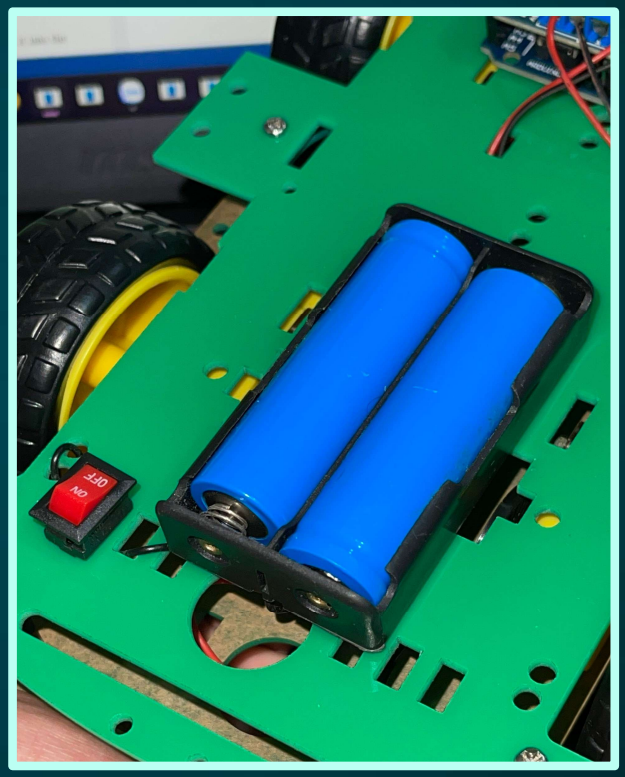
- Linked motors 1, 2, 3, 4 to Arduino UNO through M1, M2, M3, and M4 gates.

- Created electrical pathways.

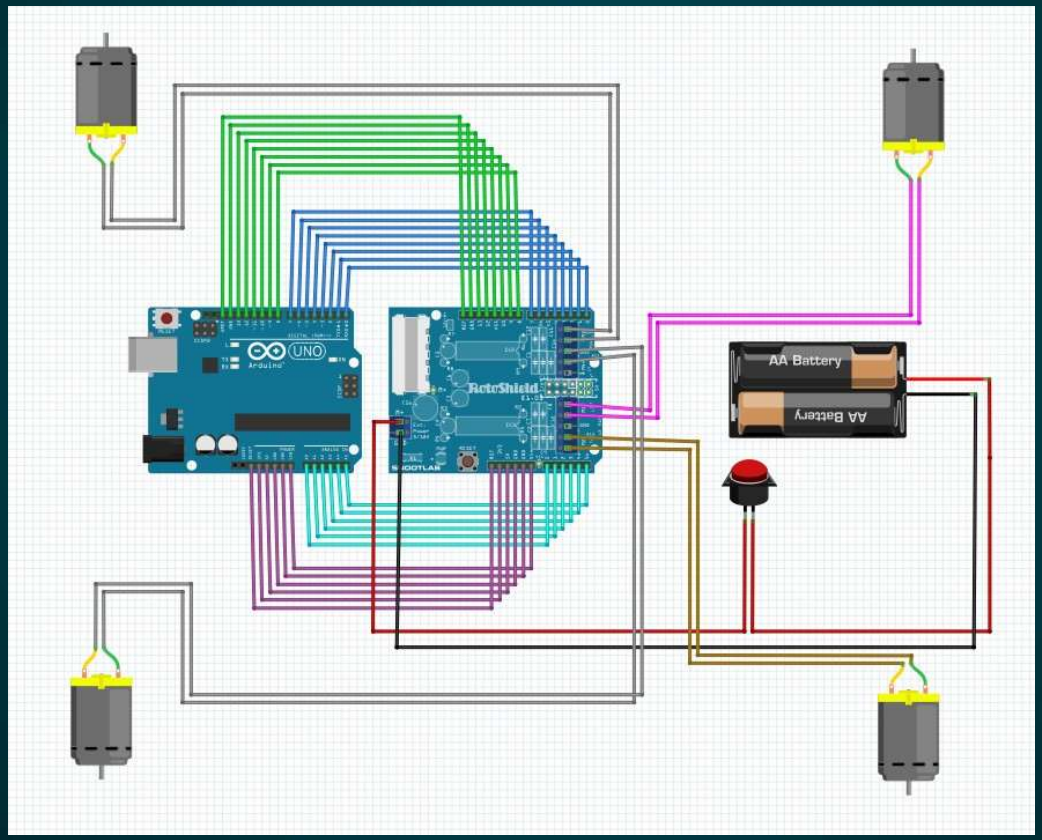


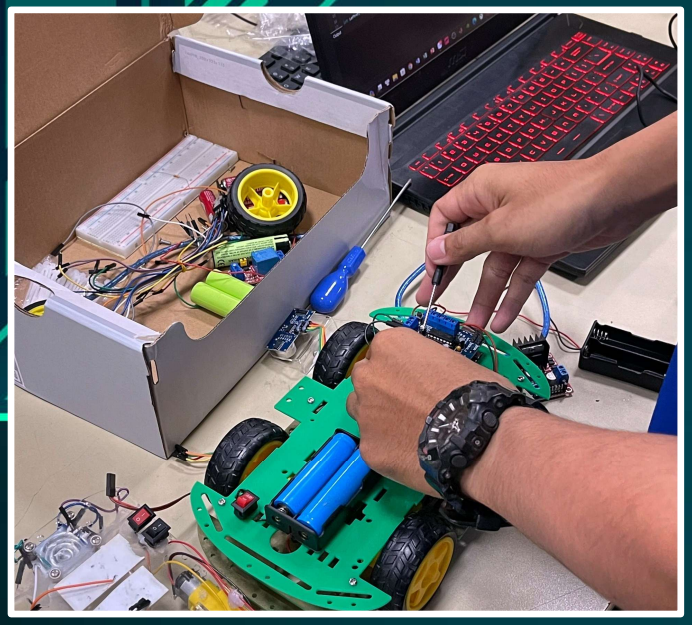
3.2.2. The chassis assemble process





The wiring diagram of the chassis





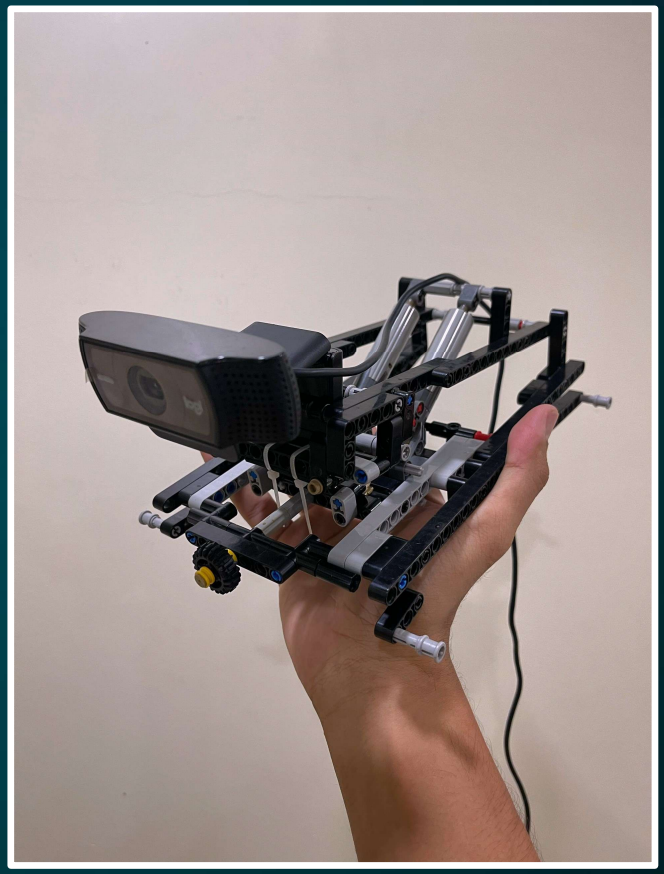
3.2.3. The camera frame adding process

3.2.3. The camera frame adding process

- Address low chassis obstructing camera view.

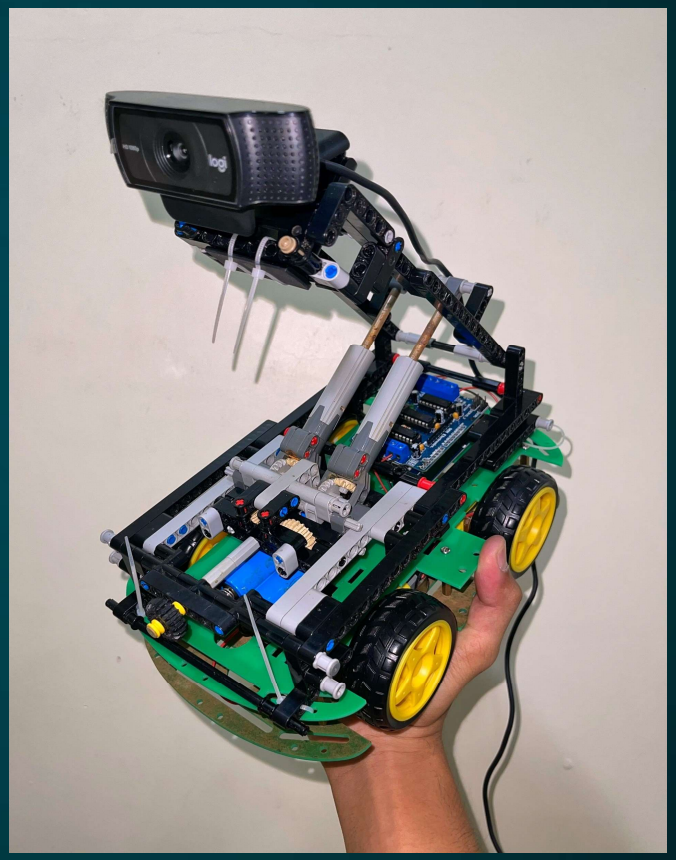
- Utilized Lego Technic for durability and adjustability.

- Insuring improved vision and easy adjustment.



3.2.4. The camera attaching process

3.2.4. The camera attaching process



THE CODING PROCESS 04

4.1. Building the Python code

1. Importing Libraries:

```
import cv2
import numpy as np
import math
import serial
import time
```

2. Setting Parameters:

```
threshold1 = 90      #Parameters for Canny edge detection.
threshold2 = 150
theta = 0
r_width = 640        #Resize the frame
r_height = 480
minLineLength = 100 #Parameters for Hough Line Transform.
maxLineGap = 10
k_width = 5          #Kernel size for Gaussian Blur.
k_height = 5
max_slider = 10 #Max slider value of Hough line detection.
```

4.1. Building the Python code

3. Defining ROI Function:

```
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
# This function is used to define a ROI
# in the frame where lane detection will perform.
# Creates mask based on the edges detected in image.
```

4.1. Building the Python code

4. Initializing Variables:

```
frame_count = 0
send_count = 0
# These variables are used for counting frames
# and controlling the frequency of data to Arduino.
```

5. Setting up Video Capture and Serial Communication:

```
cap = cv2.VideoCapture(0)
ser = serial.Serial('COM7', 115200)
# cap is initialized to capture video frames camera
# ser is initialized for Arduino serial communication
```

6. Main Loop:

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
# This loop continuously captures frames from the camera
# It runs until the user interrupts or closes the program.
```

4.1. Building the Python code

7. Processing the frame:

```
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)

#Limmited region
crop_canny=region_of_interest(edged)
```

4.1. Building the Python code

8. Detecting Lane Lines:

```
# Detect points that form a line
lines = cv2.HoughLinesP(crop_canny, 1, np.pi/180, max_slider, minLineLength, maxLineGap)
```

9. Analyzing Lane Lines:

```
    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
                if abs(angle) < 45:
                    continue
                cv2.line(frame, (x1, y1), (x2, y2), (255, 255, 0), 3)
                theta += math.atan2((y2 - y1), (x2 - x1))
# The code iterates through the detected lines and calculates their angles.
# Lines with angles close to horizontal are ignored
# The angle of the remaining lines is accumulated in the theta variable.
```

4.1. Building the Python code

10. Sending Commands to Arduino:

```
    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

# Based on the accumulated angle (theta),
# Appropriate commands are sent to Arduino to steer the vehicle.
```

4.1. Building the Python code

11. Displaying Output:

```
    cv2.imshow("Frame", frame)
    cv2.imshow("Edged", edged)
    cv2.imshow('cropped pic',crop_canny)

# The original frame, the edges detected,
# and the cropped ROI are displayed
```

12. Exiting and closing the Program:

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

cap.release()
cv2.destroyAllWindows()
# The loop exits if the user presses 'q'.
# The video capture is released
# OpenCV windows are closed.
```

4.1. 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;
```

1. Libraries and Global Variables

```

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);
}
```

2. Setup function

4.1. Building the Arduino code

```
22  void loop() {
23    if (Serial.available() > 0) {
24      // Read data from Serial
25      char command = Serial.read();
26  // Control motors based
27  // on data received from Python
28      if (command == 'S') {
29  // Run all motors at maximum speed,
30  // but the direction of rotation is BACKWARD
31        motor1.run(BACKWARD);
32        motor2.run(BACKWARD);
33        motor3.run(BACKWARD);
34        motor4.run(BACKWARD);
35  // Reset counter and status variables
36        goLeftActive = false;
37        goRightActive = false;
38        goLeftCount = 0;
39        goRightCount = 0;
40      } else if (command == 'F') {
41  // Run all motors at maximum speed
42        motor1.run(FORWARD);
43        motor2.run(FORWARD);
44        motor3.run(FORWARD);
45        motor4.run(FORWARD);
46  // Reset counter and status variables
47        goLeftActive = false;
48        goRightActive = false;
49        goLeftCount = 0;
```

```
50        goRightCount = 0;
51      } else if (command == 'L') {
52  // Pause motor 1 and motor
53  // 3 if not already paused
54        if (!goLeftActive) {
55          motor1.run(RELEASE);
56          motor3.run(RELEASE);
57          goLeftActive = true;
58        }
59        goLeftCount++;
60      } else if (command == 'R') {
61  // Pause motor 2 and motor 4
62  // if not already paused
63        if (!goRightActive) {
64          motor2.run(RELEASE);
65          motor4.run(RELEASE);
66          goRightActive = true;
67        }
68        goRightCount++;
69      }
70  // Check if Python has stopped
71  // sending the "Go left" signal
72      if (goLeftActive && command != 'L') {
73        goLeftActive = false;
74  // Reactivate motor 1 and motor 3
75  // if the "Go left" signal
76  // has been received enough times
```

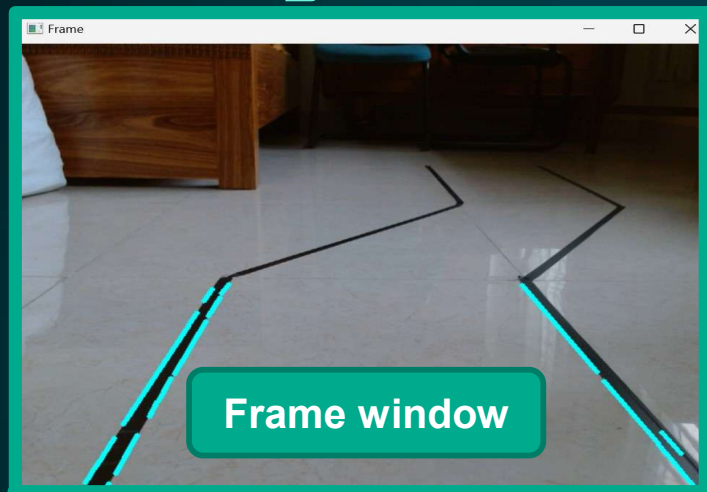
```
77        if (goLeftCount >= 3) {
78          motor1.run(FORWARD);
79          motor3.run(FORWARD);
80          goLeftCount = 0;
81        }
82      }
83
84  // Check if Python has stopped
85  // sending the "Go right" signal
86      if (goRightActive && command != 'R') {
87        goRightActive = false;
88  // Reactivate motor 2 and motor 4
89  // if the "Go right" signal has
90  // been received enough times.
91        if (goRightCount >= 3) {
92          motor2.run(FORWARD);
93          motor4.run(FORWARD);
94          goRightCount = 0;
95        }
96      }
97    }
98  }
99
```

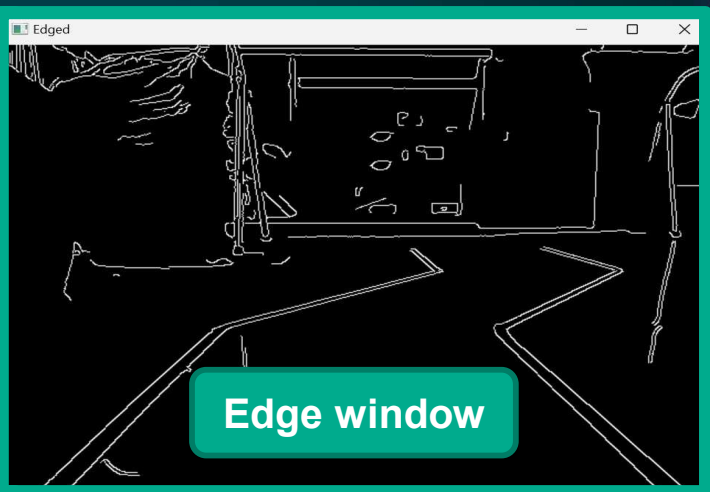
3. Loop Function:

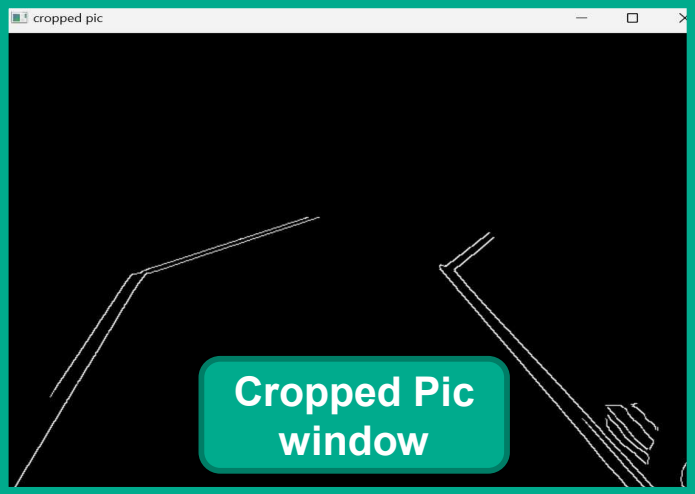
05

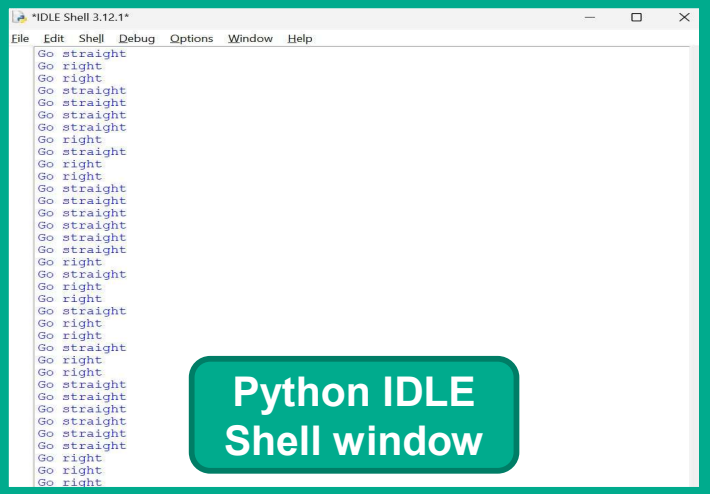
THE TESTING PROCESS

5.1. Testing the vehicle code

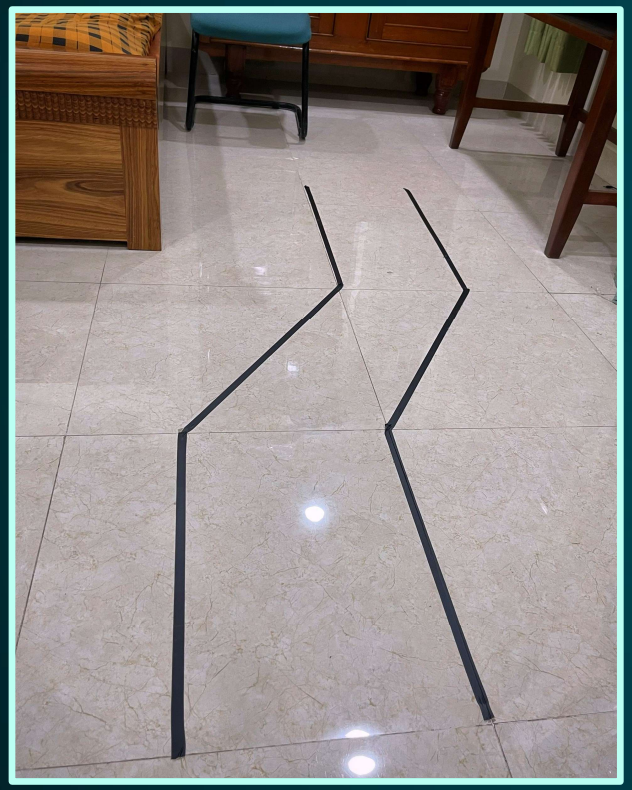








5.2. Testing the vehicle in practical



Tape lane

Practical Experimenting

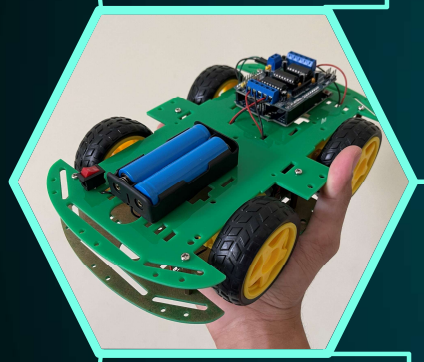
06

CONCLUSION

What have we learned from this project?

Wiring Diagram

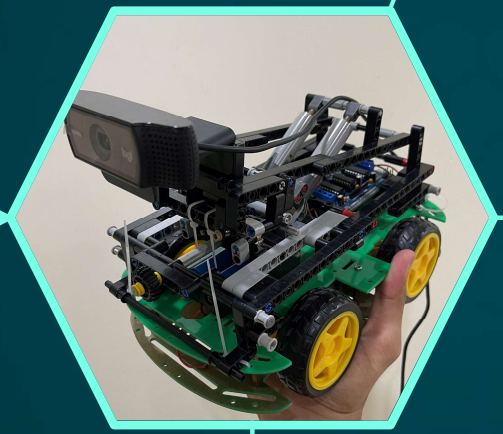
L293D Shield



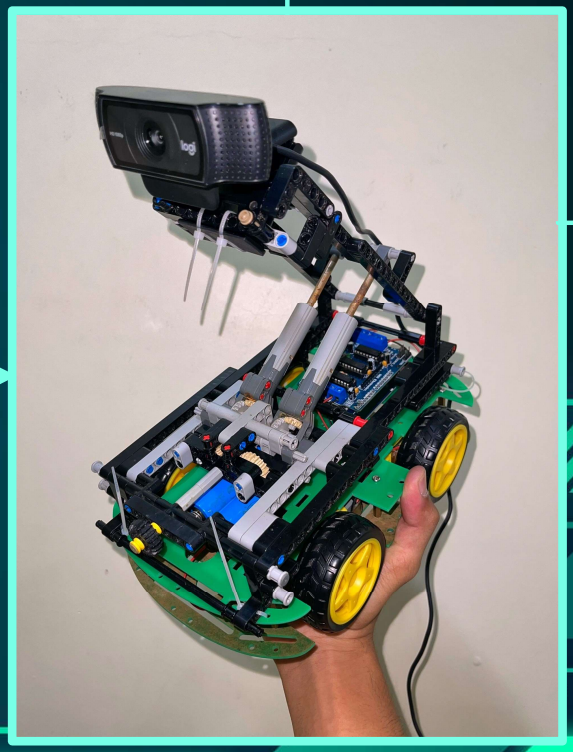
Chassis Frame

Arduino Codes

Lego Technic Frame



Logi C920e webcam



Python codes

Lane Detection

THANKS FOR WATCHING!

VEHICLE AUTOMATIC CONTROL SYSTEM

AUTONOMOUS CAR PROJECT