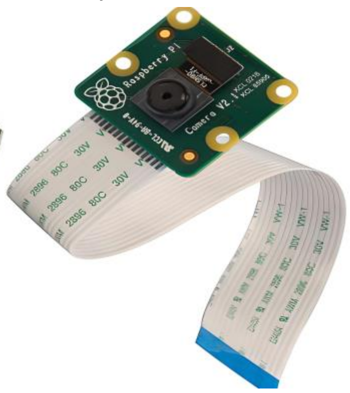
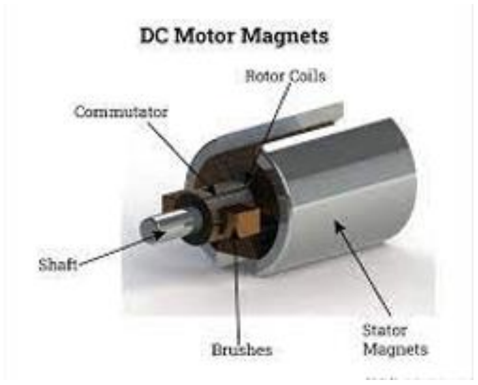
Portfolio

Lane Following Autonomous Car C++

Purpose: This project implements a self-driving vehicle control system using a combination of C++ and Python. The lane detection is performed using OpenCV in **C++ for real-time image processing**, and the control system is implemented in Python, using a PID controller
Links: https://github.com/Ucicek/lane-following-autonomous-car

I built the entire car from scratch. There are 3 important parts to the car. Sensors to obtain information, microcontroller to process the information in real-time, and dc motors to execute necessary actions for the car to stay within lanes.

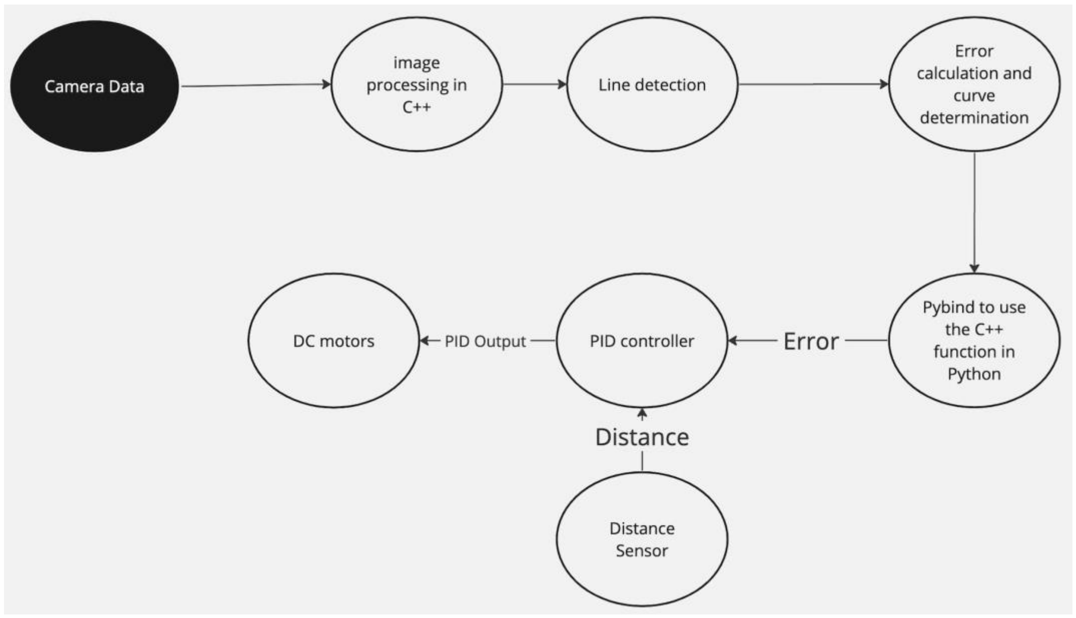




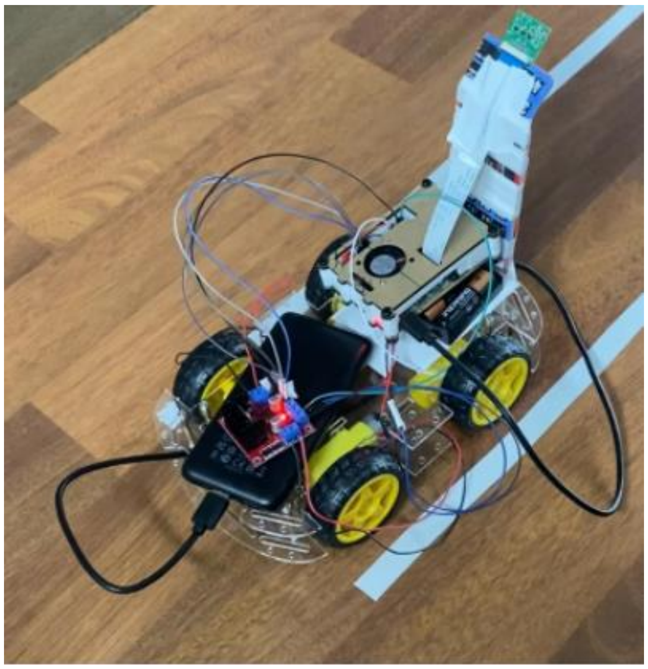
Algorithm highlights:

- **Image Preprocessing**: The input to the function process_frame is a frame captured by the vehicle's camera. This frame is first resized to reduce computation. Several image transformations are applied.
- **Line Detection**: The Hough transform is used on the processed image to detect lines, which are presumed to be lane lines. The lines are then sorted by their y-coordinates and only the top 10 longest lines are retained to account for curves.
- **Error and Curve Determination**: The horizontal distance of this midpoint from the center of the image is calculated and returned as the lane error. Also, the function determines if there's a curve ahead based on the slopes of the lane lines.

Image processing and real-time lane detection was done in **C++ for faster computation**, and the controller was implemented using Python. They were **bridged via Pybind11**.


Camera Data
image processing in C++
Line detection
Error calculation and curve determination
Pybind to use the C++ function in Python
Error
PID controller
PID Output
DC motors
Distance
Distance Sensor

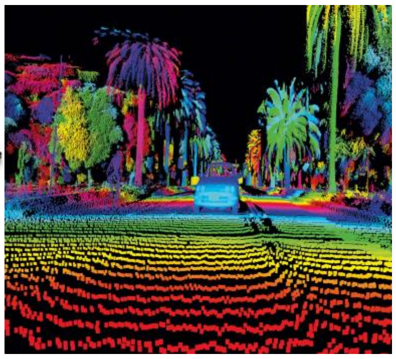
-

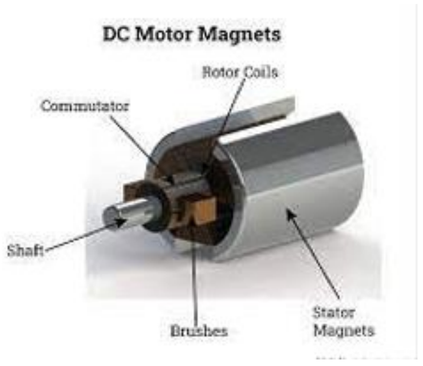


Obstacle Avoiding LIDAR Car

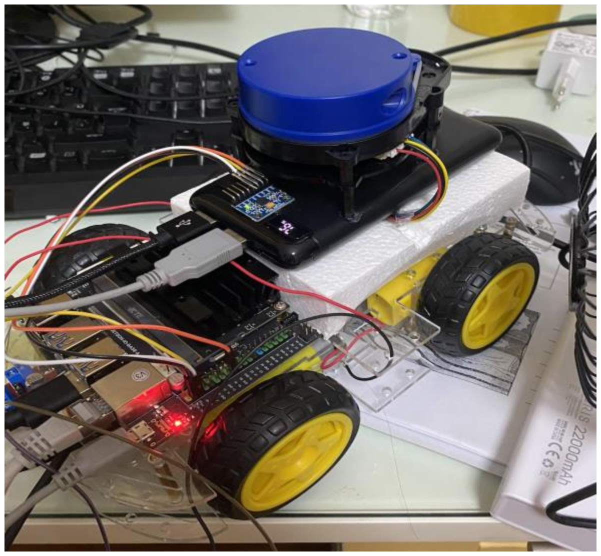
Purpose: This project aims to develop an autonomous car capable of navigating its environment using LIDAR as its primary sensor.
Links: https://github.com/Ucicek/Lidar-car-with-slam

I built the entire car from scratch. There are 3 important parts to the car. **LIDAR** to obtain information, **NVIDIA Jetson Nano** to process the information in **real-time** and create a data cloud, and DC motors to execute necessary actions for the car to stay within lanes.

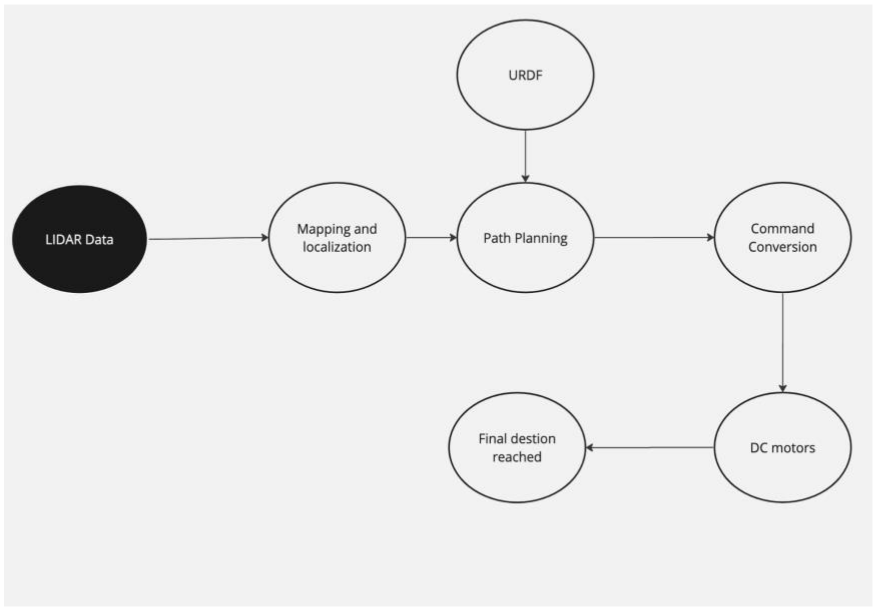




Here is the final look of the car. The car used LIDAR as its main sensor to obtain information from the environment.



Here is a diagram to explain how the car was able to avoid obstacles and sense the environment.



Software highlights:

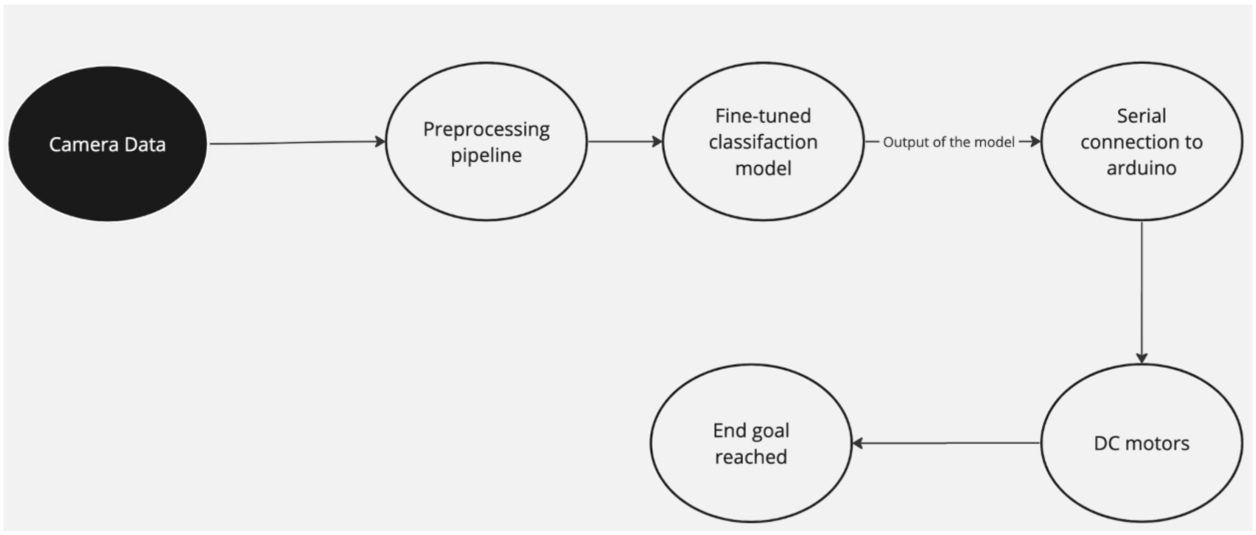
- Utilized multiple ROS packages to make sure the car avoided obstacles
- Wrote a ROS package from scratch to interface the car’s actions
- SLAM algorithm was utilized and visualized through ROS using the data cloud

Hand Gesture Based Car Controller

Purpose: Control a car using hand gestures
Links: https://github.com/Ucicek/Hand_gesture_car_controller

I built the entire car from scratch. There are 3 important parts to the car. Computer Camera to obtain information, **CNN** to make a prediction, and DC motors to execute necessary actions for the car.

The software and hardware that I contributed worked as follows:



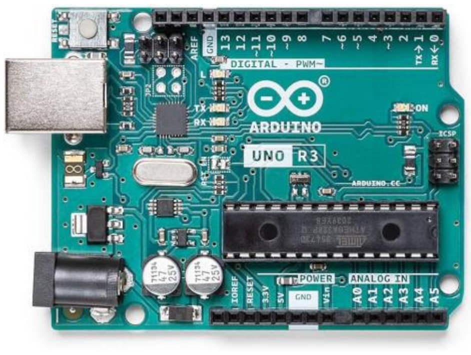
Software Highlights:

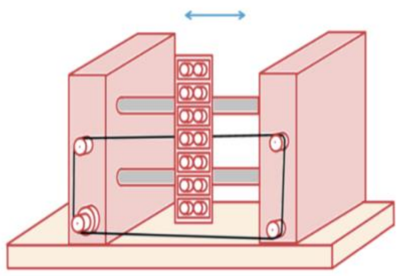
- Collected my own images and created a dataset to fine-tune a CNN model
- Utilized EC2 instances to overcome computation problems in my local PC
- Hyperparameter optimization and use of MediaPipe to increase accuracy by 25%

Hand Gesture Detection with Ultrasound Sensors

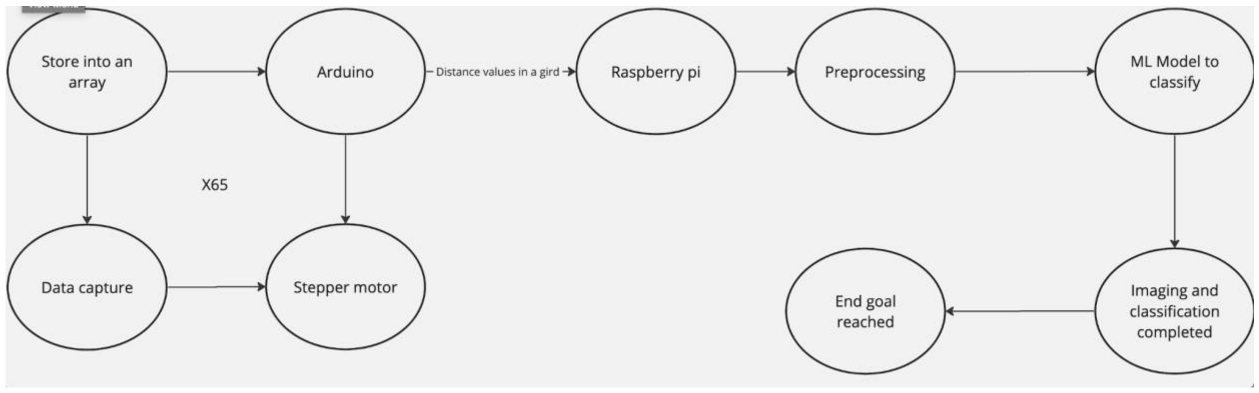
Purpose: The system is designed for recognizing hand gestures using ultrasound technology, specifically for playing Rock, Paper, Scissors.
Links: https://github.com/Ucicek/Hand_gesture_car_controller

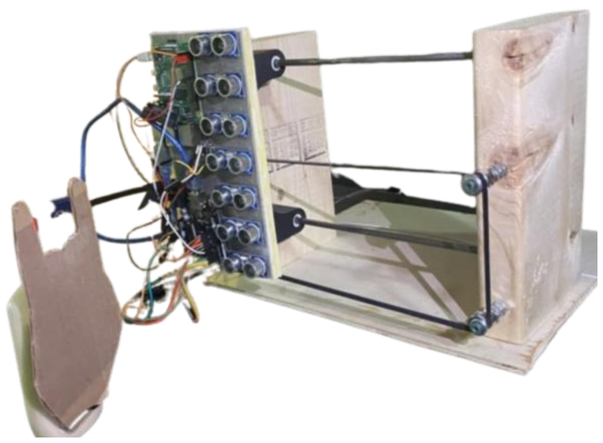
The system utilizes a vertical array of HC-SR04 ultrasonic sensors on a horizontal gantry.
The Arduino controls sensor data collection and gantry movement. The Raspberry Pi processes this data, creating a heat map of hand gestures.





The system works as follows:





Software highlights:
- Created a connection between Raspberry pi and the Arduino through hardware flags
- Used the grid created by the distance sensors to create an ultrasound image
- Used a basic ML model to classify the ultrasound image

Final look of the system