User’s Manual: Autonomous Car System

Introduction

Welcome to a guide on how to set up the Autonomous Car System project! The project simulates how autonomous cars must move and respond to high priority vehicles, like ambulance or police cars. The autonomous cars will yield to the high priority vehicles while remaining on the road. The following instructions will discuss the equipment, software, and steps necessary to run a model of an autonomous car system with multiple cars controlled by Raspberry Pi 4B single board computers.

Equipment

The project uses the materials and equipment listed in Table 1. An Amazon link is provided as a purchase option.

Table 1: Equipment List

	Description	Purchase Link
RPi Set	Raspberry Pi 4 Model B 4GB board, Power supply, USB to USB-C cable with in-line switch, 2 x Micro HDMI to HDMI cables, SanDisk 3D Gen3 X3 256Gb 2-Plane NAND Flash Die for 32GB MicroSD Card, SD card reader, 3 x Heat sinks, Raspberry Pi case with an integrated fan.	https://www.alliedelec.com/product/rs-pro-by-allied/1925286/71868752/
Freenove 4WD Smart Car Kit	Freenove 4WD Smart Car Kit for Raspberry Pi 4 B 3 B+ B A+, with Face Tracking, Line Tracking, Light Tracing, Obstacle Avoiding modules.	https://www.amazon.com/Freenove-Raspberry-Tracking-Avoidance-Ultrasonic/dp/B07YD2LT9D
Table Tennis Ball	A yellow color ball, helping the smart car to recognize each other.	https://www.amazon.com/KEVENZ-50-Pack-Assorted-Plastic-Tennis/dp/B01NA9UMI0/

Software Packages

The following software must be installed on the Raspberry Pi. The appropriate command line installation commands are listed in the table below for each package.

To install the Raspberry Pi OS, please follow the instructions in tutorial.pdf Chapter 0. To assemble the Smart Car Set, please reference tutorial.pdf Chapter 1. The tutorial file is in the .zip file downloaded from the Freenove official website.

Here is a detailed description about how to set up the necessary environment of the RPi. Command line commands are used so please be familiar with controlling the RPi with terminal or SSH. Additionally, take note to complete the following steps in the order listed.

- Set python3 as default python
  - Enter directory /usr/bin: `cd /usr/bin`
  - Delete the original python link: `sudo rm python`
  - Create new python links to python: `sudo ln s python 3 python`
  - Check python, Press Ctrl Z to exit: `python`

- Configure I2C:
  - `sudo raspi config`
  - Choose “5 Interfacing Options” , “P5 I2C” , “Finish” in order.
  - Install the following modules:
    ```
    sudo apt-get install i2c-tools
    sudo apt-get install python3-smbus
    ```
  - Test if the installation is successful: `i2cdetect -y 1`

- Install PyQt5:
  ```
  sudo apt-get install python3-pyqt5
  sudo apt-get install python3-dev
  ```

- Install WS281X library: `sudo pip3 install rpi_ws281x`

- Install Opencv library
  - Install opencv development environment:
    ```
    sudo apt get install y libopencv dev python3 opencv.
    ```
  - Install some tools:
    ```
    sudo apt get install y python 3 pil python 3 tk
    ```

- Install imutils module:
```
pip install imutils
```

- Install Git:
```
sudo apt-get install git
```

- Install the example code:
```
cd ~ git clone https://github.com/Freenove/Freenove_4WD_Smart_Car_Kit_for_
Raspberry_Pi.git
```

- Ensure all packages are up to date:
```
sudo apt-get update
sudo apt-get upgrade
```

- Restart the computer:
```
sudo reboot
```

Instructions

Before the experiment:

1. Acquire all equipment and software following the steps in the previous parts

During the experiment:

2. Construct test course for cars (Figure 1)

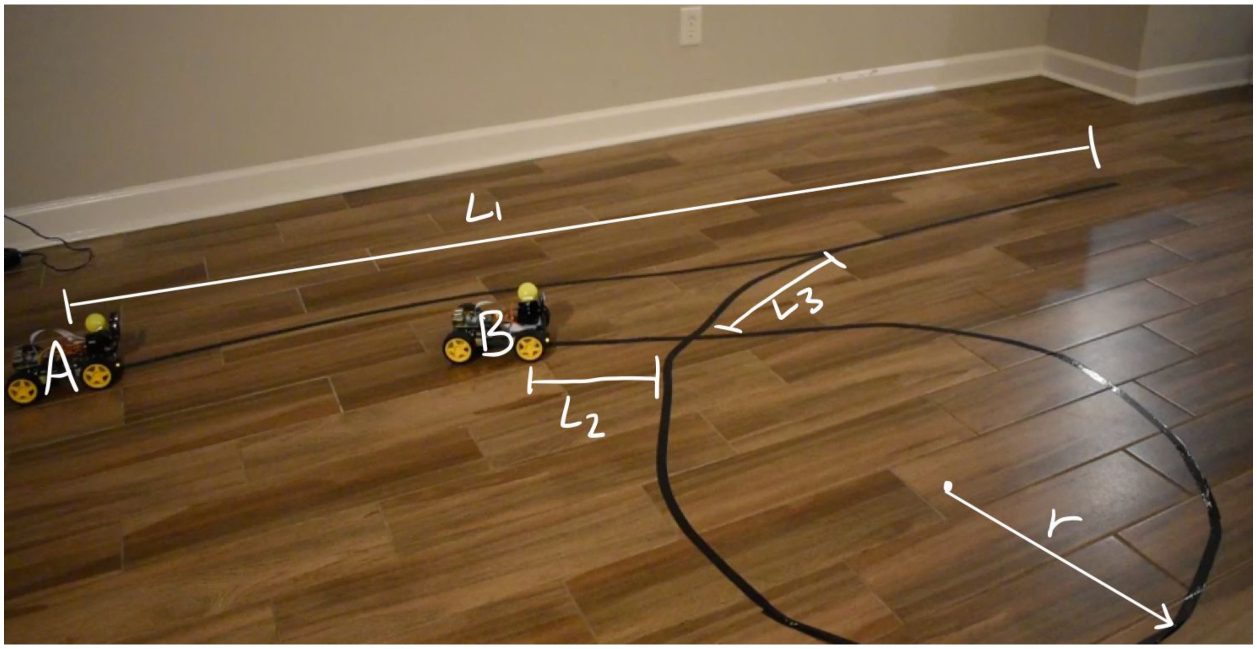


Figure 1: Test Course Setup, where L1 = 107 in, L2 = 18in, L3 = 21 in, and r =19 in

3. Place “high priority” car at point A
4. Place “civilian” car at point B
5. Remote SSH to both cars and be ready to start the autonomous programs
6. Start “civilian” car first `python civilian_main.py`
7. Start “high priority” car after 20 seconds `python high_priority_main.py`
8. Observe how the “civilian” car yielded to the “high priority” car.
9. Experiment with different interactions between the two cars to verify the yielding by the “civilian” car.

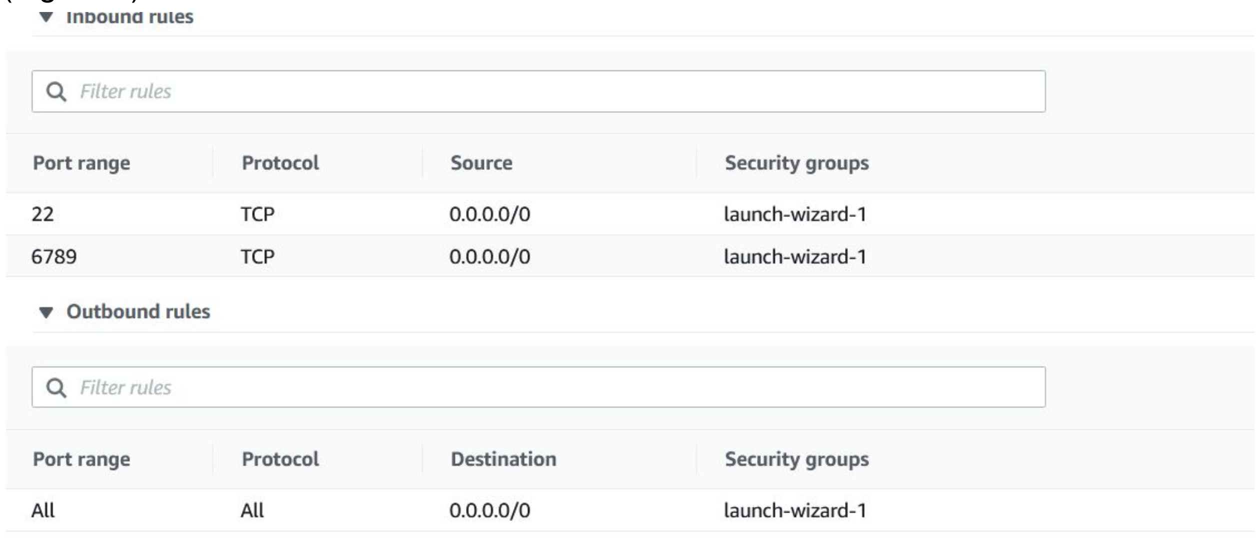
After the experiment:

10. To save the recorded video of each car type `python Image2Video.py` after each running of `python decision.py`. When executing decision.py the camera will save each frame it captures and save them in the folder ‘frame’. Image2Video.py will convert these images in ‘frame’ into a video with the format of mp4.

Network Module Installation and Setup

The central server can be set up to monitor the states of each car. The update() function can be invoked on a Decision object (an object representing the movement of the car) to send updates to the server. The user can add the update function calls as desired to the */libraries/decision.py* functions highPriority(), mediumPriority(), lowPriority() for observational purposes.

Once those parameters are added, an Amazon Web Services (AWS) server can be set up by following this tutorial. On step 6, add port 6789 to the security group as well (Figure 2).


▼ Inbound rules

Filter rules

Port range	Protocol	Source	Security groups
22	TCP	0.0.0.0/0	launch-wizard-1
6789	TCP	0.0.0.0/0	launch-wizard-1

▼ Outbound rules

Filter rules

Port range	Protocol	Destination	Security groups
All	All	0.0.0.0/0	launch-wizard-1

Figure 2: Security group settings for AWS instance

Once the Linux instance is functional, run the following commands:

1. Install Java
   ```
   sudo apt install default-jre
   sudo apt install default-jdk
   ```

2. Install GIt
   ```
   sudo apt-get install git
   ```

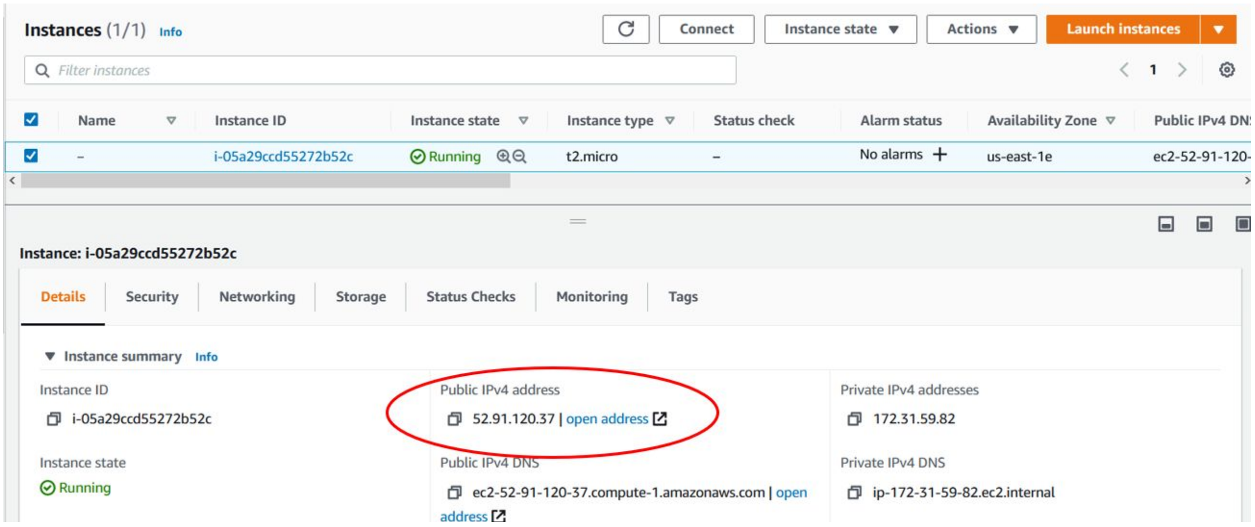
3. Ensure all packages are up to date:
   ```
   sudo apt-get update
   sudo apt-get upgrade
   ```

4. Clone the repository
   ```
   git clone https://github.com/degrace1/Autonomous-Line-Tracking-Car.git
   ```

5. Run the server program
   ```
   cd libraries/network/server/src
   javac *.java
   java Server
   ```

Once these commands have been executed, the AWS instance will be running the server program and will output logs to the console describing the updated states of the cars.

Finally change the value of the constant “address” on line 16 in */libraries/decision.py* to the public IPv4 address of the server, shown in Figure 3. For example, address = ‘52.91.120.37’. Once done, the previous section’s Instruction steps can be followed to attain logs for the state of each car using the central server.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
✓	–	i-05a29ccd55272b52c	Running	t2.micro	–	No alarms	us-east-1e	ec2-52-91-120-

Instance: i-05a29ccd55272b52c

Instance ID: i-05a29ccd55272b52c
Public IPv4 address: 52.91.120.37 | open address
Private IPv4 addresses: 172.31.59.82
Instance state: Running
Public IPv4 DNS: ec2-52-91-120-37.compute-1.amazonaws.com | open address
Private IPv4 DNS: ip-172-31-59-82.ec2.internal

Figure 3: IPv4 address of AWS Instance