Remote Pilotless Vehicle

Setup Guide

Version 1.0

1/5/2024

Table of Contents

Table of Contents

Introduction 3

Getting Started 3

- Welcome 3
- Powering on the Vehicle 4
- Network Connection 5
  - The Router 5
  - Wi-Fi Network 5
  - USB Tethering 5
  - Ethernet Ports 6
- Development Environment Setup 6
  - Remote Development Setup 6
- OpenVPN Setup 10
  - Network Model 10
  - OpenVPN Server Setup 11
  - OpenVPN Client Setup 11
- Mini-PC Setup (OpenCV) 12
- Post-setup 15

Introduction

This setup guide is intended to help you get set up to be able to develop on the vehicle. It includes information about how to set up your computer to talk to the vehicle. You'll only have to do most of these steps once for a new computer. For day-to-day usage, refer to the user guide. This is a living document, so if your team adds or removes things on the vehicle, please add it to this document for future reference!

Getting Started

Welcome

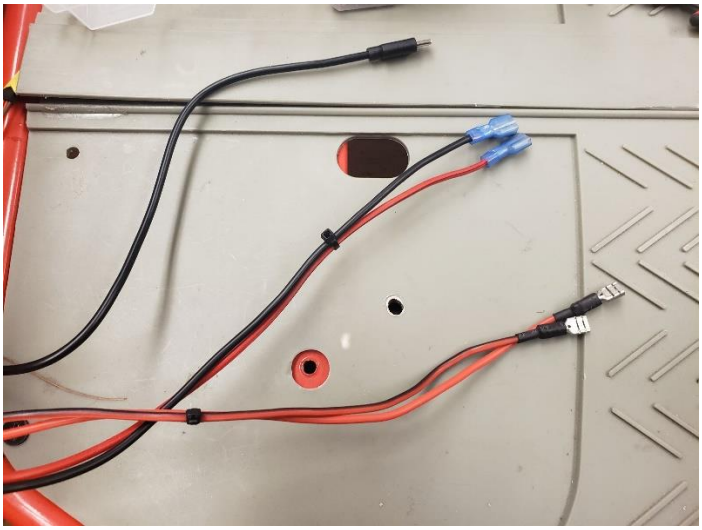
This is the Remote Pilotless Vehicle project under the guidance of Dr. Tamer Omar. It has undergone several revisions, and this document pertains to the vehicle's progress as of Fall 2023. These were the goals that the vehicle accomplished by Fall 2023:

- Remote control via keyboard, Xbox controller, and the Thrustmaster wheel on the VRX simulator.
- Cellular connectivity between the vehicle and the pilot.
- Stream live video back to the pilot with reasonable quality and low latency.
- Have two modes: full manual control, or full autonomous mode.
- Lane detection under full autonomous mode.
- Onboard battery monitoring.
- Collision avoidance using sensors.

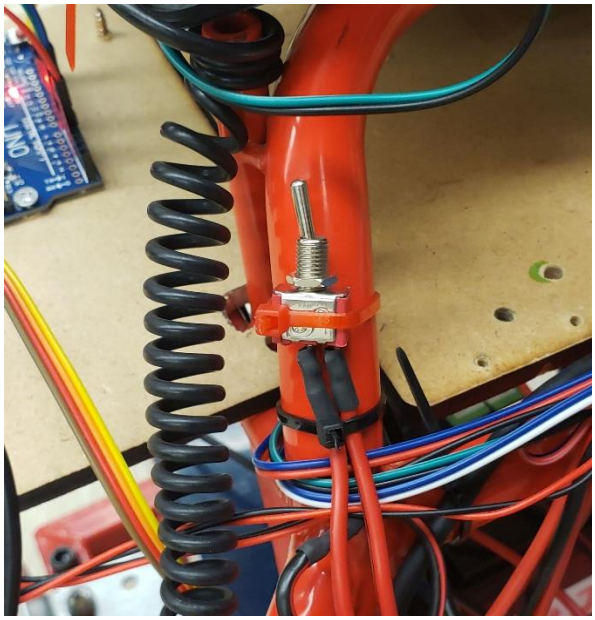
You can find more information in the final report from Fall 2023.

Powering on the Vehicle

The first thing to do is to power on the vehicle. Locate the two pairs of female spade connectors as shown below:



The wires with the connectors that have blue insulation are for everything EXCEPT the high-current motors used for propelling and steering the vehicle. These wires power the Raspberry Pi, the Wi-Fi router, and everything else. These wires expect 12 volts, with red being positive and black negative. There is no reverse voltage protection! Please ensure the correct polarity before connecting. You can use one of the two sealed lead acid batteries onboard the vehicle or you can use any 12-volt power adapter that can provide at least one amp of current (useful for when you don't need to move the vehicle but need it on). If things don't turn on, flip the toggle switch shown below:



Once you do, everything should light up like a Christmas tree.

The other pair of wires is for the high-current motors mentioned above. The reason for two 12-volt lines is so the Raspberry Pi doesn't reset if the voltage dips too low when driving hard on a shared battery. Don't be alarmed by any sparks when you connect these wires, that's from the current rushing into the capacitors of the motor drivers.

Network Connection

The Router

The GL.iNet GL-AR300M16 router is the small grey square box on the rear of the vehicle. The purpose of this router is to bridge your smartphone's internet connection with the computers onboard the vehicle. It also broadcasts a Wi-Fi network for accessing the computers.

Wi-Fi Network

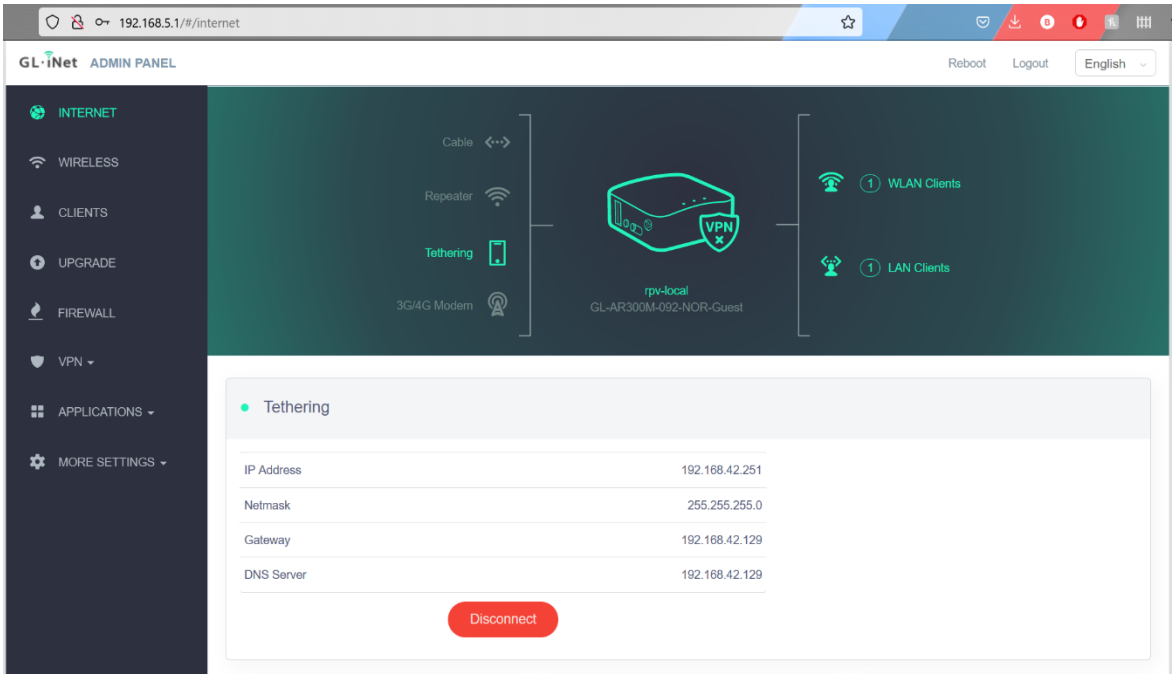
After powering on, wait a few minutes until you see all three lights lit on the Wi-Fi router. You should now see a Wi-Fi network called "rpv-local". Connect to this network with the password "02222023**"** with the computer you intend to write code and test the vehicle with.

FYI: Windows 10 does this strange thing where it disconnects from the Wi-Fi network if it doesn't have internet access. If you suddenly lose connection while trying to do the next few steps, this is why.

USB Tethering

Once connected, you'll notice that your computer doesn't have an internet connection. That's because the router doesn't have any way to reach the internet. To give it (and the whole vehicle) an internet connection, you need to connect a smartphone to the USB port and turn on USB tethering on your phone. If an Android device last connected to the router, it will automatically begin piping the internet connection from the phone without any further action. The same goes for iOS devices. However, if the last device connected is different from the one you just connected, you need to initialize the connection from the router side.

To do this, open a web browser and type in "192.168.5.1". You should be greeted with a login page. Login with "admin" and "022223" or "02222023" (I forgot). You should now see the home page below:



If you see a red “Disconnect” button, then there is no further action needed. If you see a “Connect” button, click on it, and wait for it to show an IP address. The router will remember the last type of device connected so you only need to do this once if you continue using that type of device.

Once connected, you should be getting internet access. One thing to note is that cell reception is very weak in the lab. Your internet connection is going to be very slow. To get around this, turn on Wi-Fi on your phone and let it connect to the school’s network. If your phone supports it, it can share this Wi-Fi connection with the router which will give the Raspberry Pi a high-speed connection to the internet. Just remember to turn Wi-Fi off if you want to run the vehicle over a cellular connection.

Ethernet Ports

There are two Ethernet ports on the router: WAN and LAN. Typically, WAN is used for the internet connection “source” while LAN is for local devices. Since the WAN connection is provided by the phone over USB, this frees up the WAN port. Thus, the WAN port has been configured to act like another LAN port. This gives both the Raspberry Pi and the Linux mini-PC a network connection. You can connect an Ethernet switch to either the WAN or LAN port if you need more ports. If you want to change the function of the WAN port back to the original, the shortcut to do this is on the homepage.

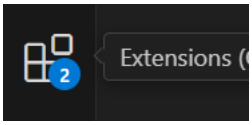
Development Environment Setup

Remote Development Setup

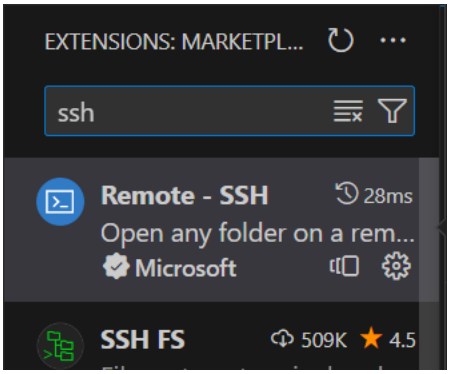
Before we start writing code, we first need to set up our development environment on our computer. To make it easier to do such tasks without having to re-upload code to the Raspberry

Pi every time we make a change, we will use Visual Studio Code with the SSH extension. Your VS Code IDE will connect to the Pi and stay in sync with it while you are connected. This means that when you write code in your IDE, your code is simultaneously being written to the Pi. There is no need to bother with remote desktop or manually copying files over with this method. Here are the steps to get set up:

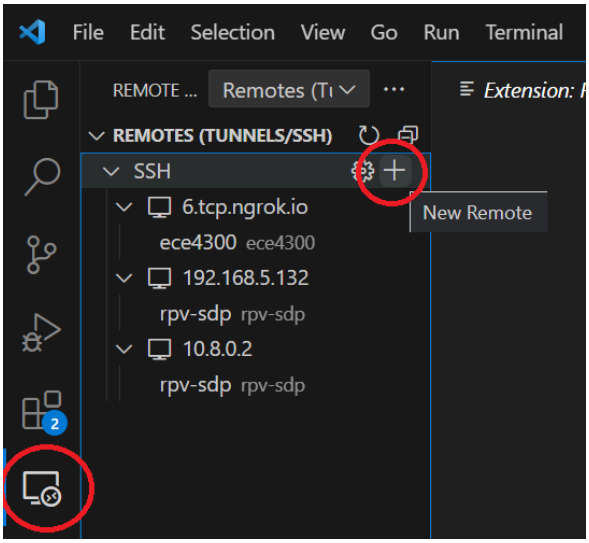
1. Download the latest version of VS Code here: https://code.visualstudio.com/download
2. Run the installer and open the IDE.
3. When you finish the initial setup, look for the Extensions icon shown below:



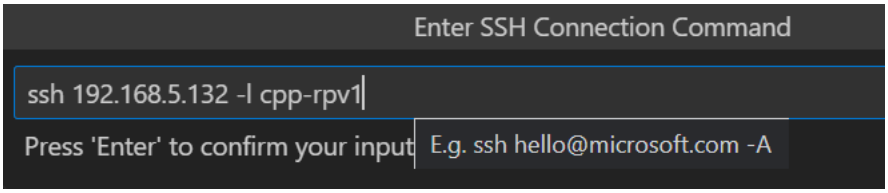
4. Click on it and type “SSH”.
5. Locate the search result shown below:



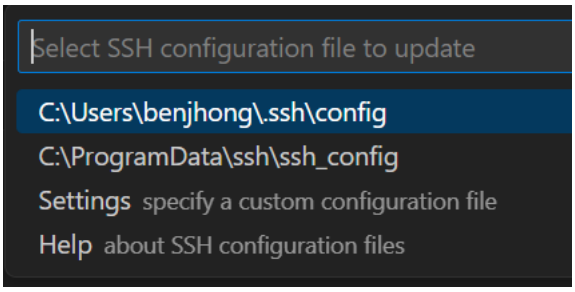
6. Click on the green “Install” button after clicking on that result.
7. Once installed, you should see a new icon on the left icon bar.
8. Click on the new icon and click on the “+” to add a new remote.



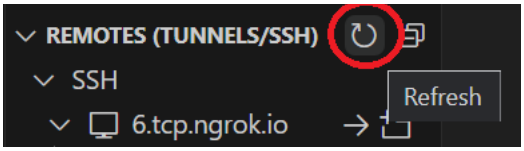
9. In the text box at the top of the screen, type the following:



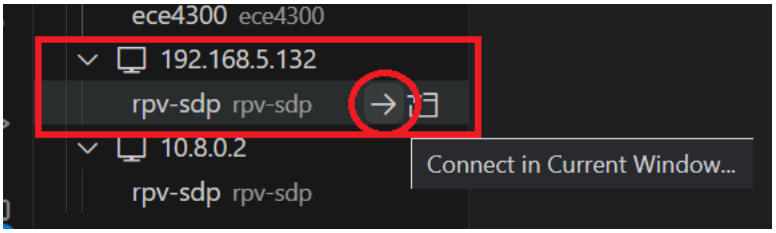
10. Next, hit Enter on your keyboard unless you want to specify a different config file.



11. If the newly added target doesn't show up on the list, click the Refresh button:

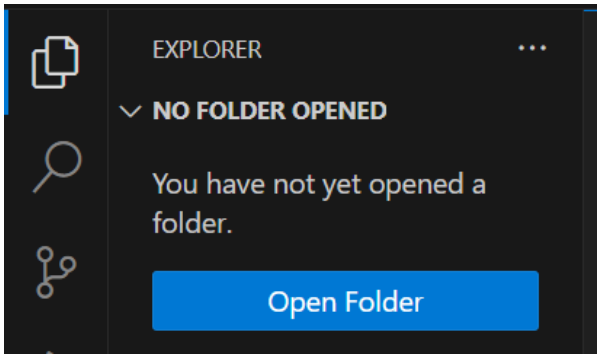


12. You should now see the new target. Hover over "rpv-sdp" and click on the arrow:



13. When it prompts for a password, type "022223**"** and hit Enter.
    a. If it never prompts for a password, ensure the Pi is powered on and you are connected to the Wi-Fi network.
14. Select "Linux" when it asks for the remote operating system.
15. If there is a new VS Code update, it will begin downloading it on the Pi. This may take a few minutes, depending on the speed of the internet connection.

16. Once finished, click on the Folder icon, and click "Open Folder":



17. Navigate to the folder “rpv-sdp” and click “OK”. It should be in “/home/cpp-rpv1/”
18. You should now see all the source code.

PuTTY Setup

In some cases, it may be useful to connect to the Pi’s terminal without having to open VS Code. You can use any SSH terminal program for this, but PuTTY was the simplest.

1. To begin, download the appropriate installer here: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
2. When finished, open PuTTY and type “192.168.5.132” into the Host Name text box. Leave the port as 22. Leave connection type as SSH.
3. Click “Open” at the bottom and click “Accept” if prompted.
4. Type “cpp-rpv1” as the username.
5. Type “022223” as the password (hidden for privacy).
6. Done!

If you would like PuTTY to save the IP address, you can jump to step 2, fill in accordingly, give that session a name under “Saved Sessions”, and click “Save” to save your settings.

VNC Viewer Setup

You can also connect to the Raspberry Pi’s desktop using VNC instead of connecting a monitor to the Pi.

1. To begin, download the appropriate installer here: https://www.realvnc.com/en/connect/download/viewer/
2. When finished, open VNC Viewer and type “192.168.5.132” into the text box. Press Enter to connect.
3. Type “cpp-rpv1” as the username.
4. Type “022223” as the password.
5. Done!

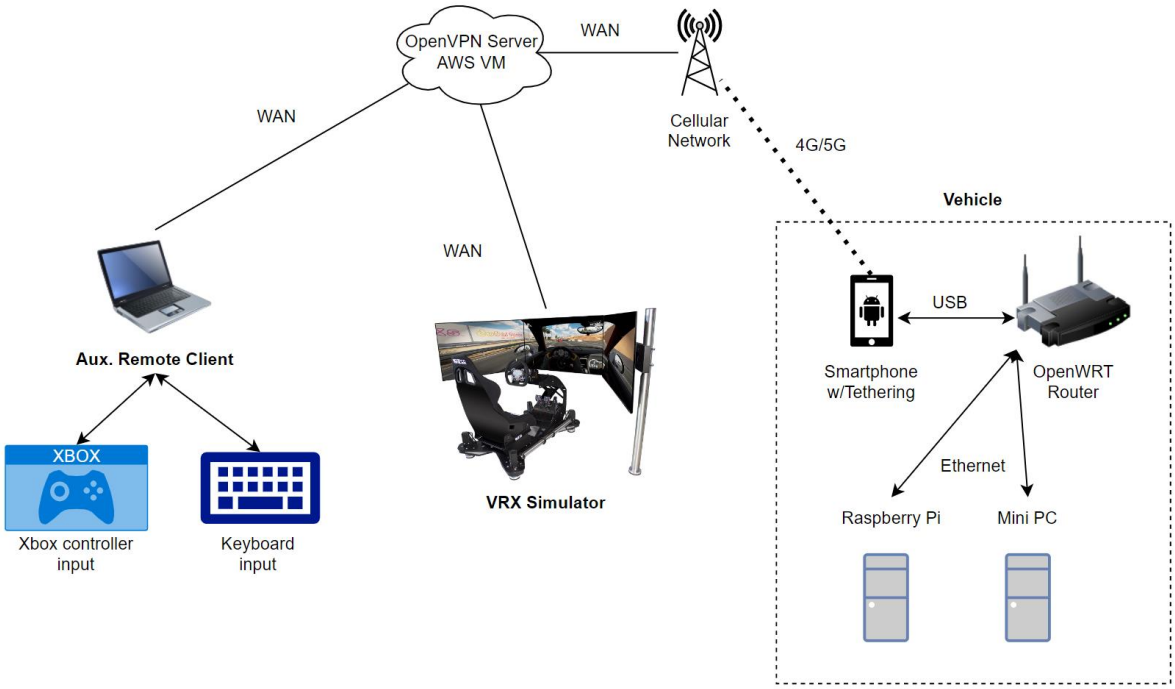
OpenVPN Setup

At this point, the vehicle is powered on, you are connected to the router via Wi-Fi, your smartphone is tethered, and your VS Code is set up and connected to the Raspberry Pi. From here, you can launch the client and server programs as outlined in the user guide.

However, if you also need to be able to operate the vehicle remotely over cellular and not Wi-Fi, you'll need to set up an OpenVPN server.

Network Model

Shown below is the layout of the network. Since typical cellphone carriers don't allow port forwarding, the client has no way of reaching the server running on the Raspberry Pi. To get around this, OpenVPN is used to establish a tunnel between the two using a central server hosted in the cloud. In our testing, we used Amazon Web Services as they offer a 1-year trial for their T2 micro instance.



In this model, there are four VPN clients:

1. Your laptop (Aux. Remote Client)
2. The VRX simulator
3. The Raspberry Pi on the vehicle
4. The Mini PC on the vehicle

All clients connect to the central VPN server. When all three clients are connected, they can see and talk to each other.

OpenVPN Server Setup

You can use any VPS service for the OpenVPN server. You can install the server manually, but we found that the following script made installation a breeze. Run this inside the VPS:

```
https://github.com/Nyr/openvpn-install
```

When you run the script, set the server to use TCP and listen on port 443. You can use the default DNS server. Ensure that you have opened this port on the VPS.

Next, the script will ask you for a name for the first client. You can name it whatever you want, just make it meaningful. After the installation finishes, re-run the script three more times to add three extra clients. In total, you should have four as stated above. When you create the new clients, the script should tell you where the client config file is located. For us, it was in **/root/<client name>.ovpn.** Use any SFTP client to retrieve these files as they will need to be copied to the three client computers.

OpenVPN Client Setup

Linux Setup

The OpenVPN client is already installed on the Raspberry Pi, but in case you need to reinstall it, use the following command to do so:

```
sudo apt-get install openvpn -y
```

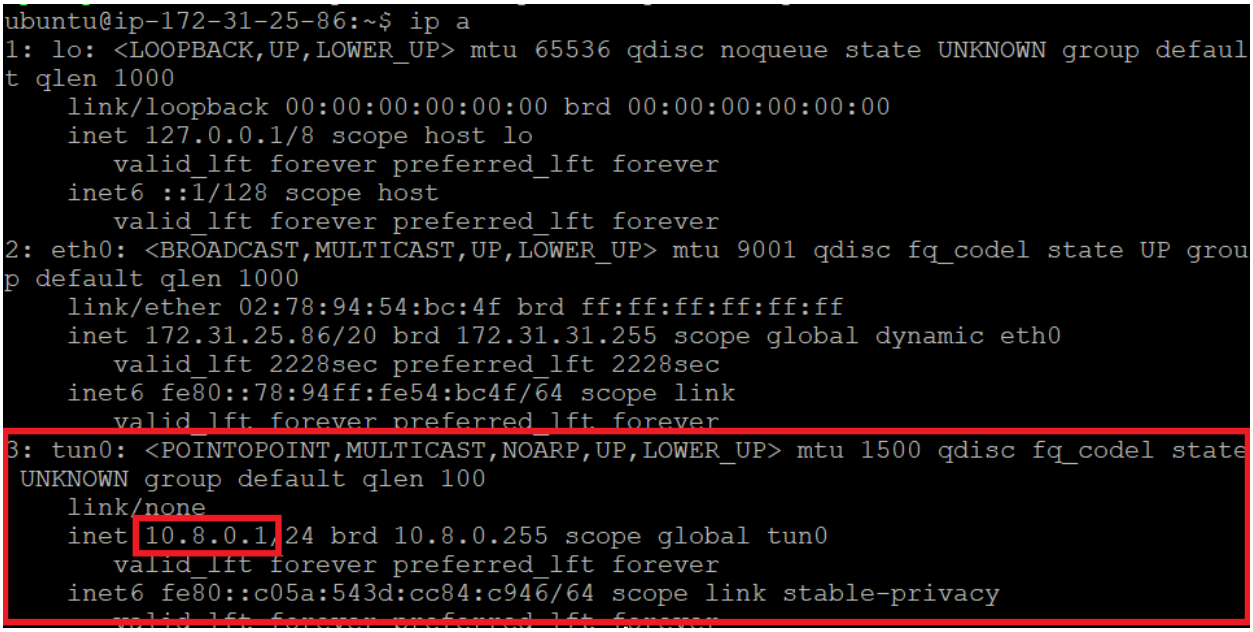
Once installed, connect to the server by running:

```
sudo openvpn --config /path/to/config.ovpn
```

Replace “/path/to/config.ovpn” with the location of the client file you copied in from earlier.

Note: After connecting, the terminal will be occupied by the client. Closing the terminal may terminate the connection. If you want to disconnect, press CTRL+C, otherwise leave the terminal as-is to stay connected.

After running the command, you should see “Initialization Sequence Completed” at the bottom, indicating that the connection was successful. To see your VPN IP address, open a new terminal and run “ip a” or “ifconfig” and look for the following:



Here, the VPN IP address of this machine is 10.8.0.1. Any other clients connected to the VPN can reach this machine using this IP address.

Run the same command on each client device to determine its VPN IP so that you can connect to them later.

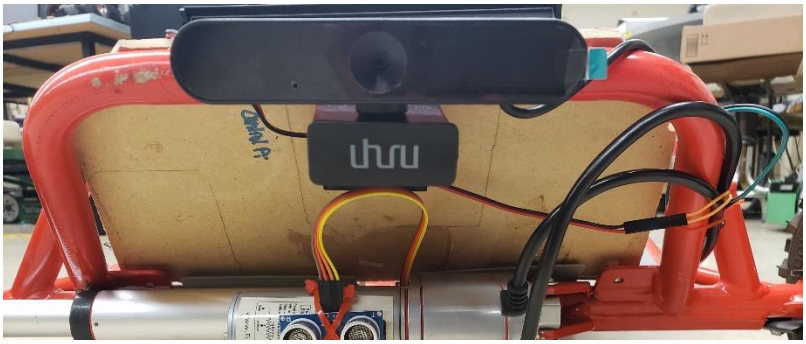
Windows Setup

For Windows clients, download the client program from https://openvpn.net/client/ and install it. Launch the program and import the profile. It may already start at the import page, but if it doesn't, click on the "+" icon at the bottom right corner. Click on "FILE" and browse for the .ovpn file. Once imported, click "CONNECT" to connect. While connected, you can scroll down to see your IP under "YOUR PRIVATE IP". To connect/disconnect, use the slider switch button.

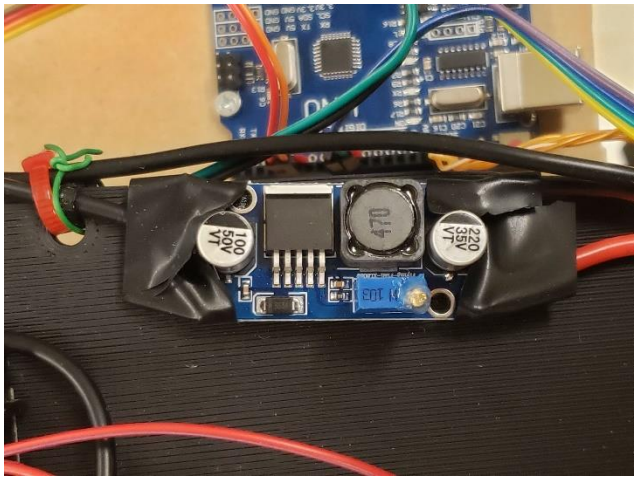
Mini-PC Setup (OpenCV)

To give the vehicle lane detection functionality, you'll need to use a small computer that can run Ubuntu to run an OpenCV Python script. The following is already implemented on the vehicle:

1. A USB webcam (front-mounted).



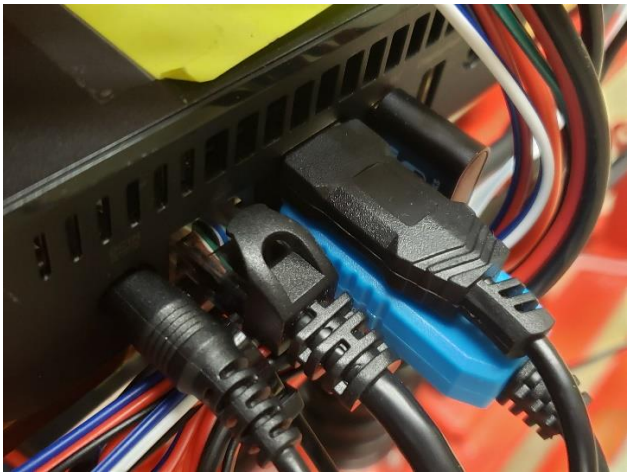
2. A boost converter to take 12V and boost it to 19V.



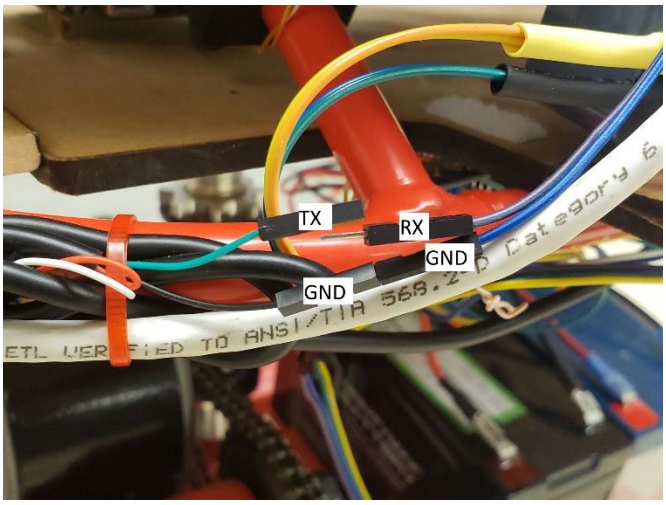
   You can adjust the output voltage to suit your needs using the blue potentiometer (covered in hot glue to lock it).

   It can take 3.0V-30V and boost it to 5V-35V at 4A. Uses the XL6009 DC-DC converter.

3. An Ethernet cable to connect it to the router.
4. A USB to TTL adapter to allow the computer to send serial commands to the Pi (blue thing).



Shown below is what the other end of the blue adapter looks like.



USB to TTL adapter pinout:

Since the PC is only sending to the Pi, the white receive wire is tucked away, along with the red 5V wire. If you need commands sent both ways, connect the white wire to the TX pin on the Pi.

The computer pictured earlier is an ASUS CN62 Chromebox that has an i7-5500 CPU, 8GB DDR3, and a 32GB SSD. It doesn't need to be as powerful as the i7 as the Python script wasn't very taxing on the CPU. The Chromebox does not natively support any OS except ChromeOS. The custom firmware at https://mrchromebox.tech/ was used to enable it to run Ubuntu Linux.

The software environment consisted of an OS version of Ubuntu 22.04.3 LTS, Python version 3.10.12, and the Python implementation of OpenCV version 4.8.1.

Since the Chromebox was a personal computer, it wasn't left behind. Any mini computer with similar specs should be able to run the lane detection script.

Note: If you intend to remotely access the computer using remote desktop software, you'll need a dummy plug to fake a monitor. Otherwise, you'll get a black screen. You can use the one plugged into the Raspberry Pi (it doesn't need it) pictured below:

You can use any software to allow remote connections. We used the built-in Ubuntu RDP server and the built-in Windows RDP client to remotely control the computer.

Post-setup

At this point, the initial setup is complete. You can now refer to the user guide for day-to-day usage.