Automated Parking System

TABLE OF CONTENTS

1. Install Jetson Software with SDK Manager

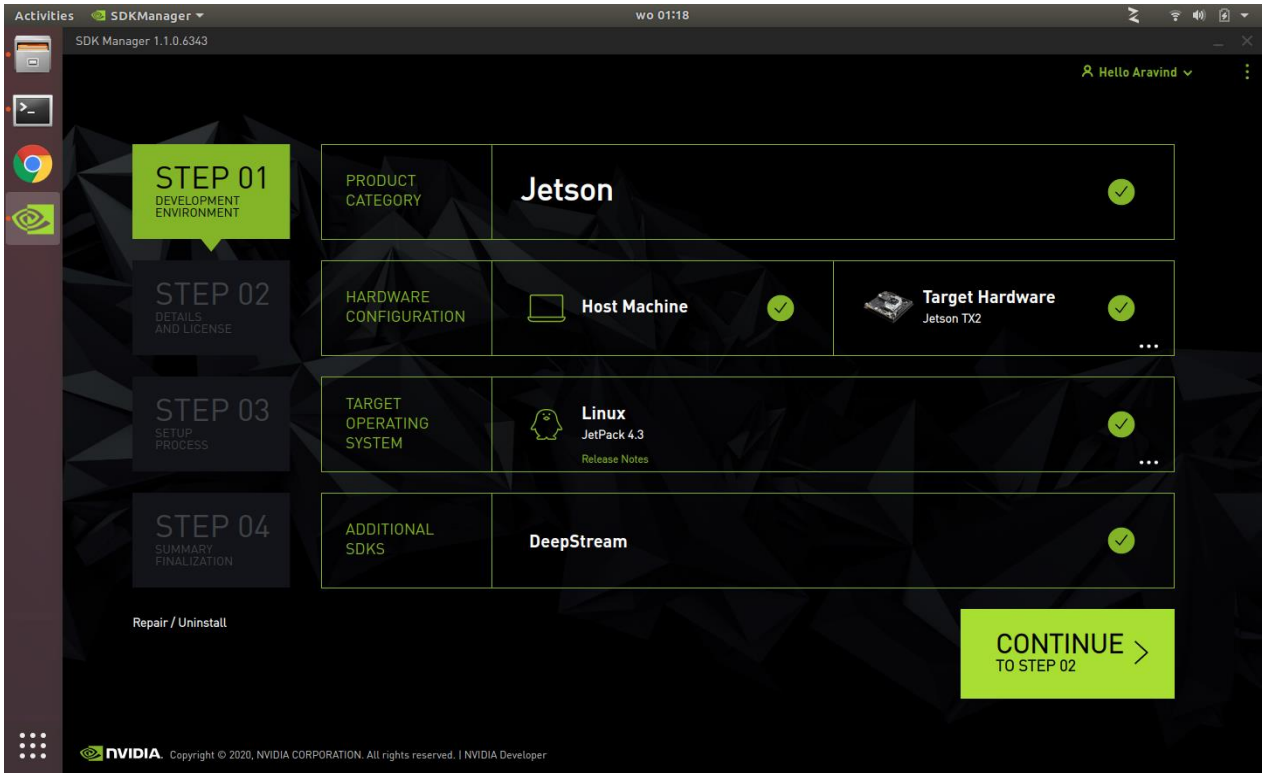
This section is intended to help you use the NVIDIA SDK Manager GUI to successfully configure your development environment.

Step 1: Setup the Development Environment

1. From the **Step 01 Development Environment** window, select the following:

- From the **Product Category** panel, select Jetson.
- From the **Hardware Configuration** panel, select the host machine and target hardware.
- From the **Target Operating System** panel, select the operating system and Jetpack version. (for Nvidia Jetson TX2, jetpack 4.3)
- Finally, select any **Additional SDKs** that you wish to install.

  An ellipsis (...) in the bottom right corner of a category box indicates that more than one option is available. Click on the ellipsis to show a drop-down menu of available options.



Your display may differ from the one shown here. The information in this screen is populated from your NVIDIA user account access and permissions. If you don't see your product category in the available selections, you may need to verify that your NVIDIA account is registered to the required programs.

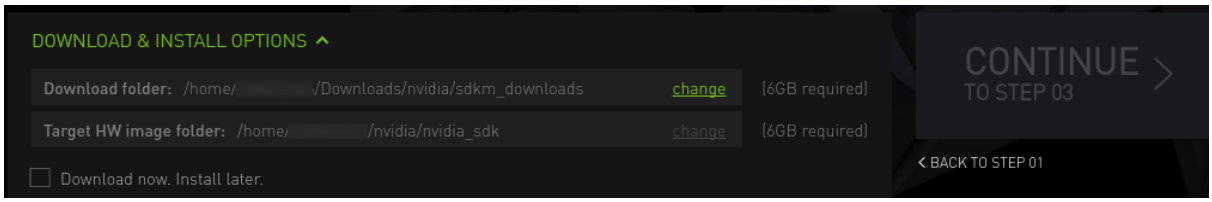
2. Click **Continue** to proceed to the next step.

Step 2: Review Components and Accept Licenses

1. From **Step 02 Details and License**, you can expand the host components and target components panels to review the components that will be installed on your system.
2. To review the licenses, click on the **license agreements** hyperlink at the bottom of the page.

3. Enable the checkbox to accept the terms and conditions of the license agreements.

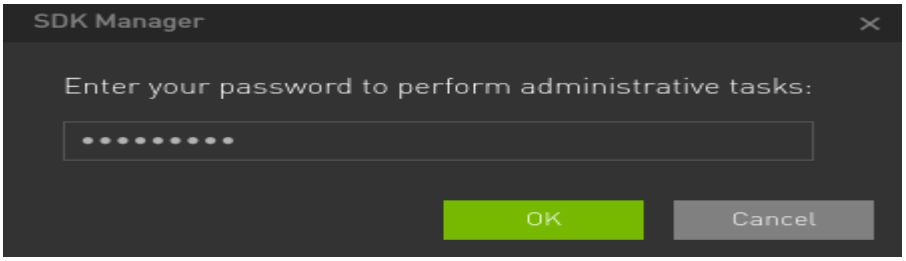
4. If you want SDK Manager to download all setup files to a location other than the default path, expand the **Download & Install Options** drop-down menu, then select the path you wish to use.



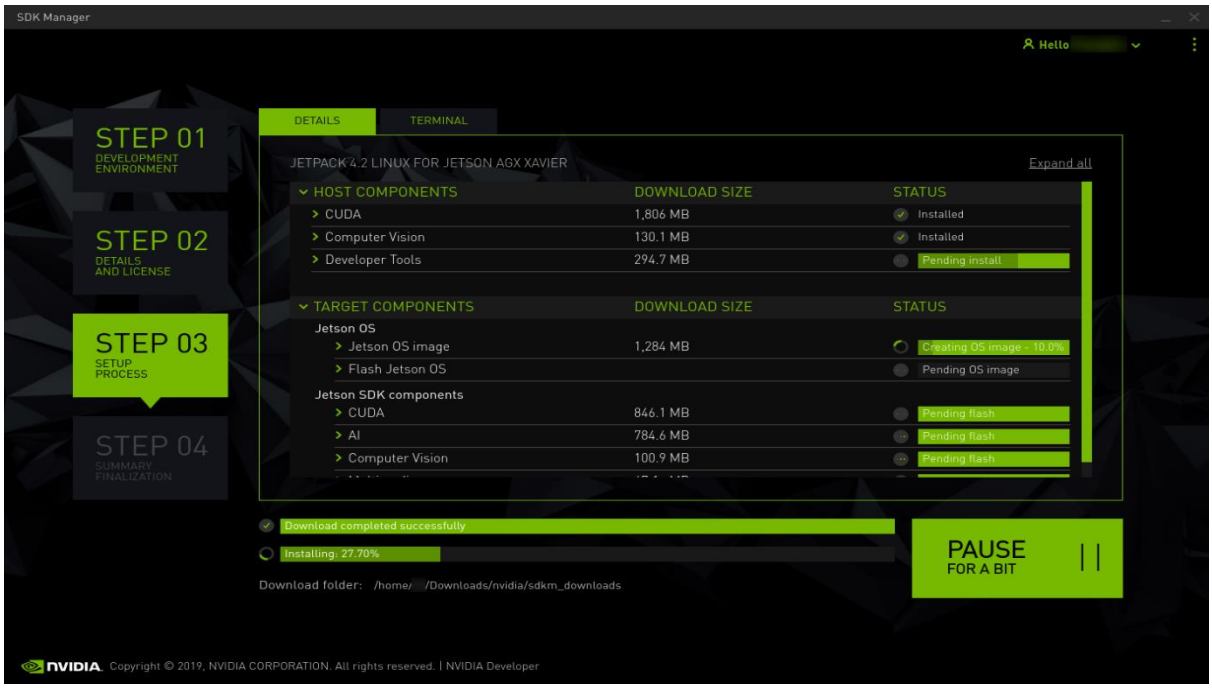
5. Select **Continue** to proceed to the next step.

Step 3: Installation

1. Before the installation begins, SDK Manager prompts you to enter your sudo password.



2. The display shows the progress of the download and installation of the software.

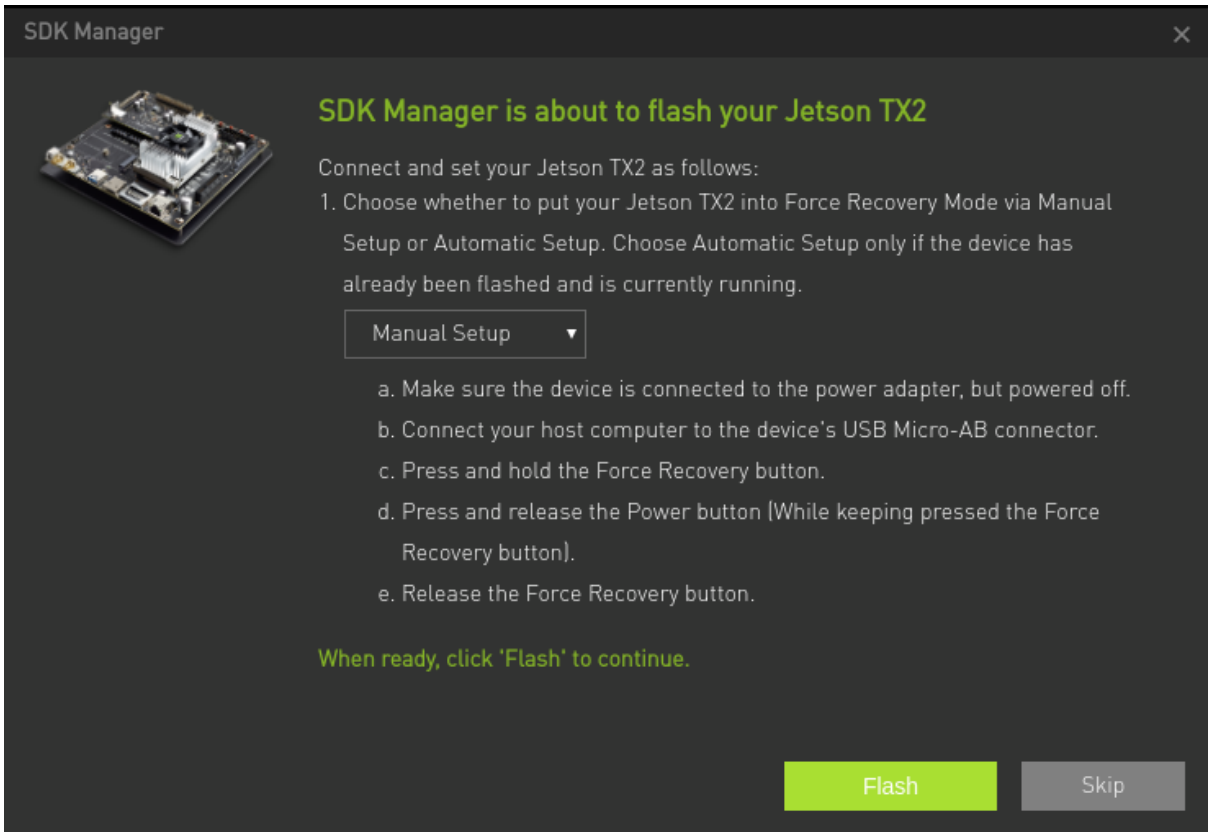


Select **Pause** / **Resume** to toggle the download and installation process.

3. At the top, you can toggle between the **Details** and **Terminal** tabs. The Terminal tab shows detailed information about the download and installation, with any errors highlighted.

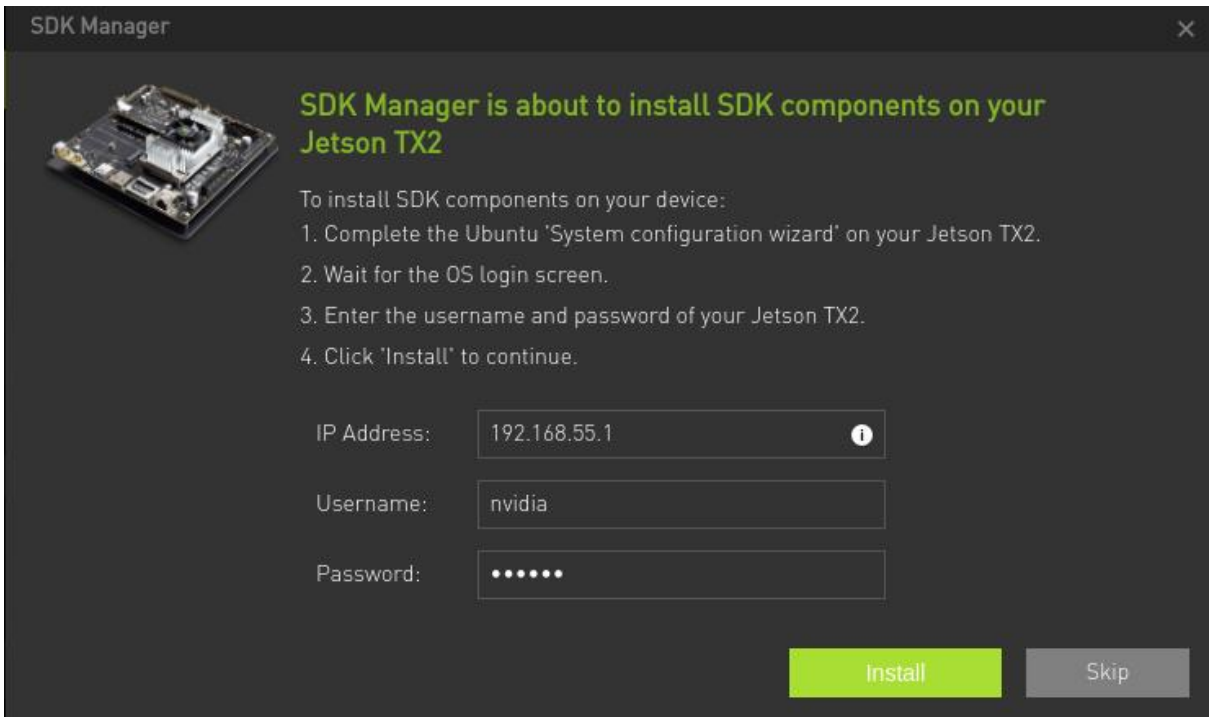
4. On the Terminal tab, you can use the **Filter text** field to filter and search for specific information.

5. SDK Manager opens a dialog when it is ready to flash your target device. A prompt provides instructions for preparing your device to get it ready for flashing.



6. After SDK Manager completes the flashing process, the monitor connected to your Jetson system will show a prompt for initial setup.

- As part of the initial setup process, select a username and password for the Jetson system.
- After the initial setup process is complete, the Jetson system boots to the Linux desktop.
- Enter the same username and password you created during the Jetson configuration into SDK Manager's post-flash installation dialog.



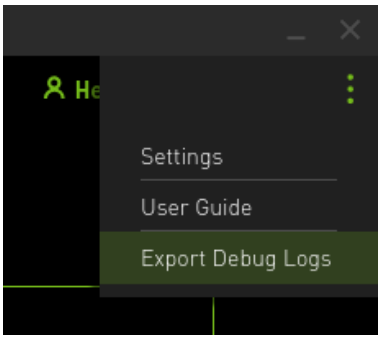
7. SDK Manager will now complete the installation of the software libraries. Skipping this step will not install any SDK components on your target hardware, and will keep a clean operating system on your device.

Step 4: Finalize Setup

1. From **Step 04 Summary Finalization**, there is a summary of the components that were installed, along with any warnings or errors that were encountered.

2. The **Export Debug Logs** link creates a ZIP file of all log files created during installation. This ZIP file is located in the same folder path where the SDK Manager installer downloaded all components.

   Alternatively, click the menu icon in the top right corner of the window ("⋮"), and choose Export Debug Logs from the drop-down menu in the top-right corner.



3. Consult the **Error Messages** for information about any errors you may encounter.

4. Click **Finish and Exit** to complete the installation.

This will install most of required dependencies, additional information can be found on desktop

2. Configuring VNC for Remote access

The setup needs to be accessed remotely without any wired connections for better mobility.

A VNC server allows access to the graphical display of a Linux for Tegra system over the network. This allows you to work physically remote from the Linux for Tegra system, and avoids the need to connect an HDMI display, USB keyboard, or mouse.

All commands specified below should be executed from a terminal on the Linux for Tegra system. This could be a serial port, an SSH session, or a graphical terminal application running on the HDMI display.

Installing the VNC Server

It is expected that the VNC server software is pre-installed. Execute the following commands to ensure that it is:

```
sudo apt update
sudo apt install vino
```

Enabling the VNC Server

Execute the following commands to enable the VNC server:

```
# Inform the system which graphic display to interact with
export DISPLAY=:0
# Enable the VNC server
gsettings set org.gnome.Vino enabled true
gsettings set org.gnome.Vino prompt-enabled false
gsettings set org.gnome.Vino require-encryption false
# Reboot the system so that the settings take effect
sudo reboot
```

Connecting to the VNC server

Use any standard VNC client application to connect to the VNC server that is running on Linux for Tegra. Popular examples for Linux are gvncviewer and remmina. Use your own favourite client for Windows or MacOS. VNC viewer is preferred for Windows.

To connect, you will need to know the IP address of the Linux for Tegra system.

Execute the following command to determine the IP address:

```
Ifconfig
```

Enter the IP address on remote viewer and connect to the system.

If any issues are found in remote access visit this page https://github.com/Aravindseenu/Nvidia-jetson-VNC-remote-access for Installing the VNC and setting up Nvidia as headless device.

3. Install Arduino IDE on Ubuntu – Nvidia Jetson.

The Arduino software or IDE (Integrated Development Environment) contains a text editor that is generally used for writing, compiling and uploading code in Arduino hardware. It helps to connect and communicate with the Arduino hardware. Arduino IDE can be run on all major operating system platforms like Linux, Windows, and Mac OS. It is available for both 32-bit and 64-bit OS platforms.

Download Arduino Software

First, you will need to download Arduino IDE package from the Download page of Arduino official website. Make sure to download the right version 32-bit or 64-bit depending upon your operating system. This is the official link of Arduino IDE downloads page:https://www.arduino.cc/en/Main/Software

Alternatively, you can use the following wget command to download the Arduino Software (IDE) package directly on the terminal. The latest version is 1.8.13

```
cd /tmp

wget https://downloads.arduino.cc/arduino-1.8.13-linux64.tar.xz

tar -xvf arduino-1.8.12-linux64.tar.xz
```

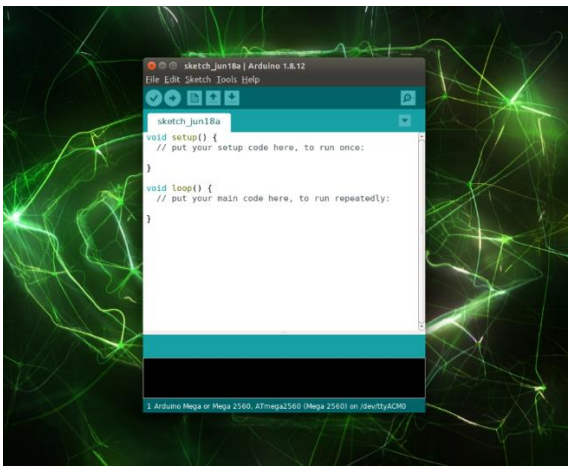
Install Arduino IDE

Now you will need to install the Arduino IDE. To prepare for the installation, navigate to the **Downloads** folder. You will need to uncompress the downloaded Arduino archived folder(if done manually). Now move into the extracted arduino-1.8.12 directory and run the installation script as root.

```
cd arduino-1.8.13/

sudo ./install.sh
```

Once the installation is done, a desktop icon will be created on your desktop. Open the software by clicking it



4. Uploading the Arduino program

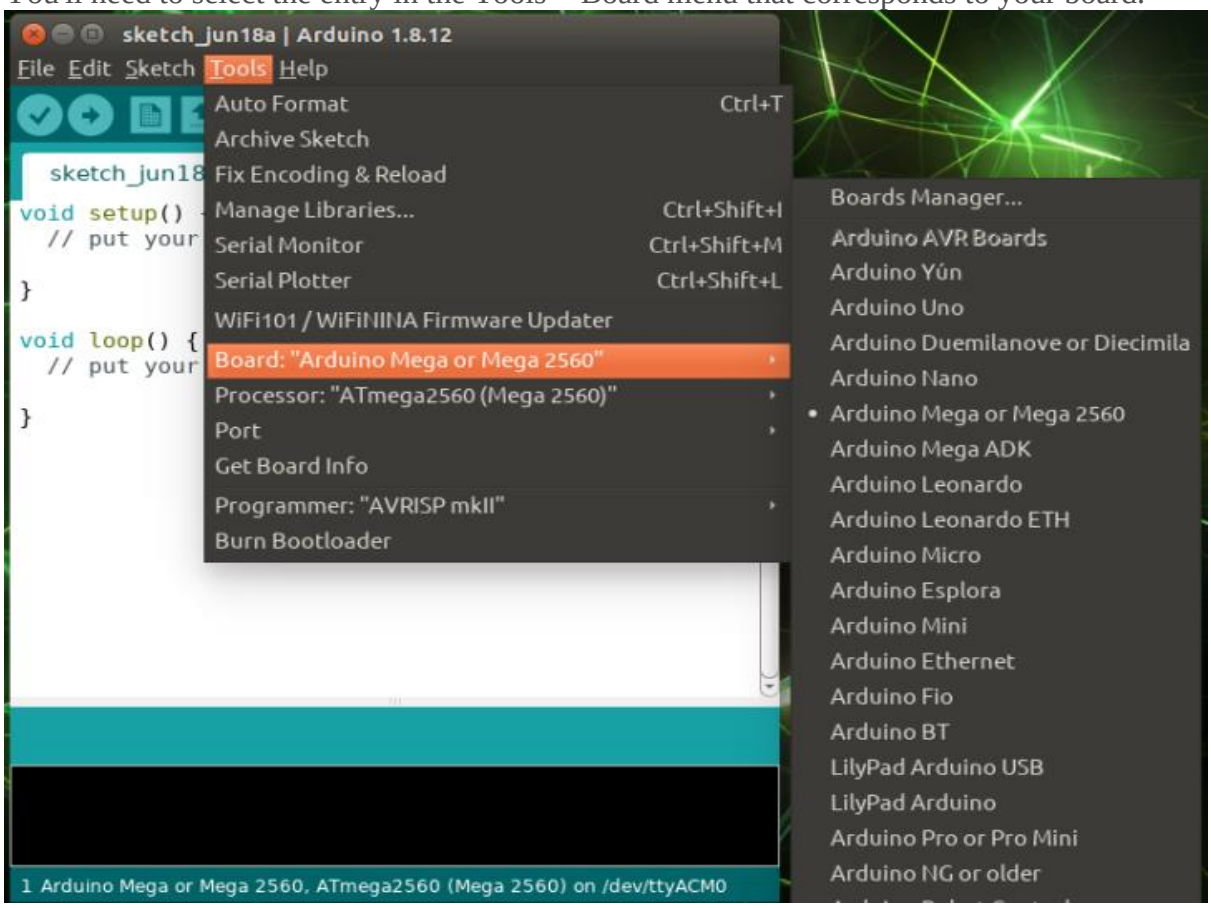
Connect the Arduino board to the NVidia Jetson with usb cable

Open the sketch

Open the Parking Arduino file. Folder Source code > Arduino files > parking_arduino.ino

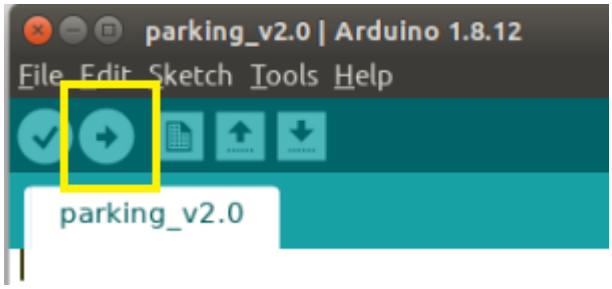
Select your board type and port

You'll need to select the entry in the Tools > Board menu that corresponds to your board.



Upload and Run the Sketch

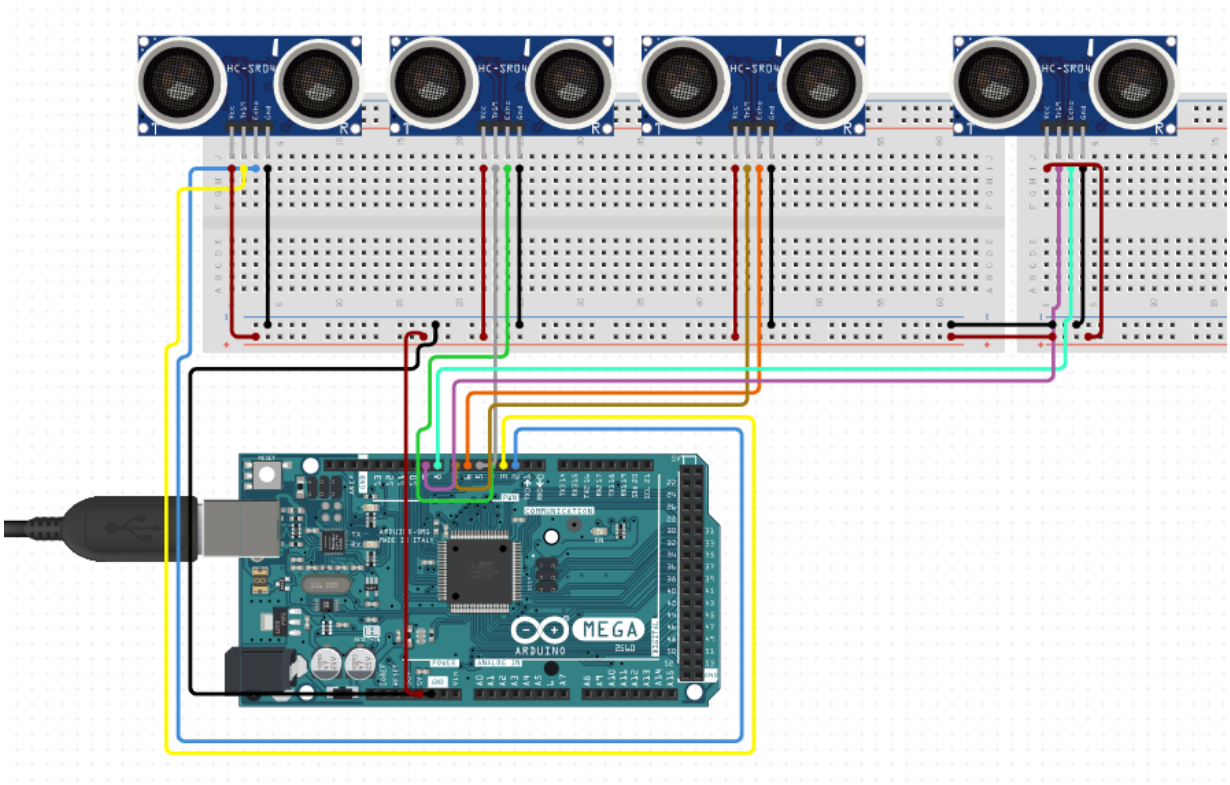
To upload the sketch to the Arduino, click the Upload button in the upper left to load and run the sketch on your board:



Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.

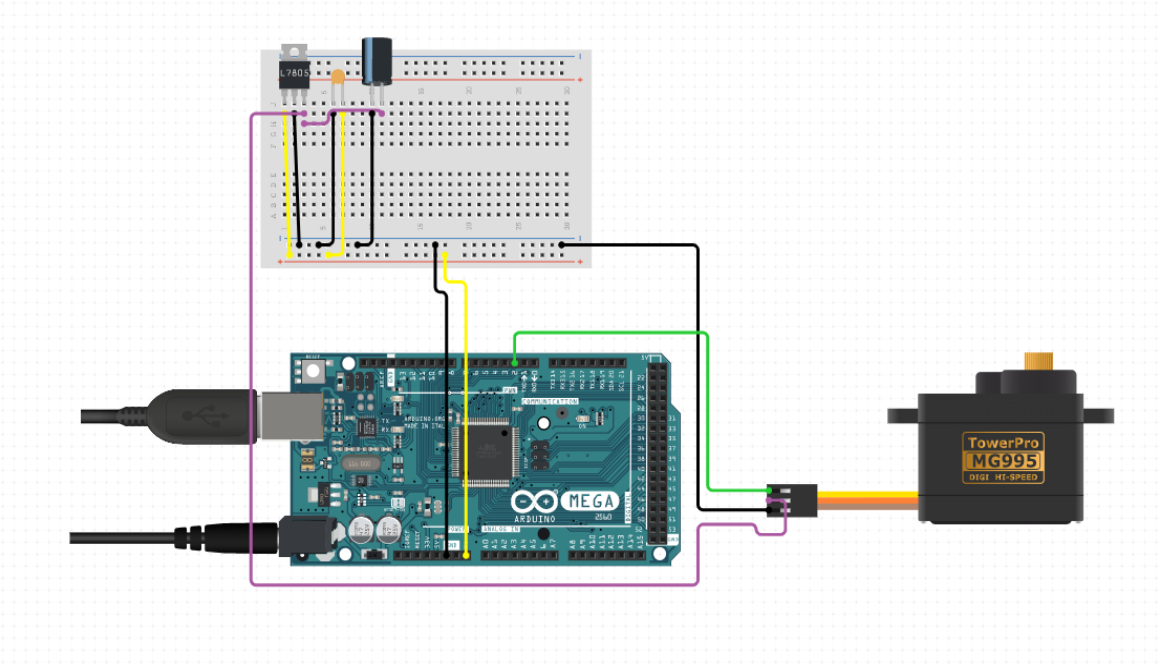
5. Connecting the sensors and actuators

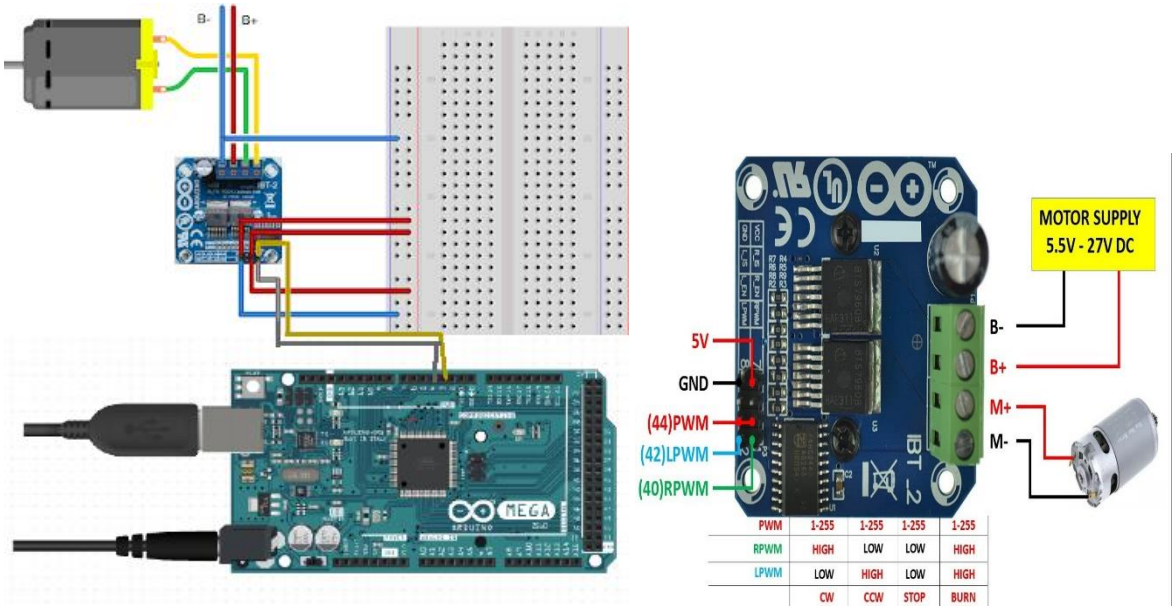
Connecting the sensors



Connect the sensors using breadboard or shield as per the diagram, or follow the instructions from the code. Place the sensors in their corresponding places.

Connecting the actuators





Connect the servo motor and traction motor as per the given circuit diagram (if the pins are changed then edit the codes accordingly)

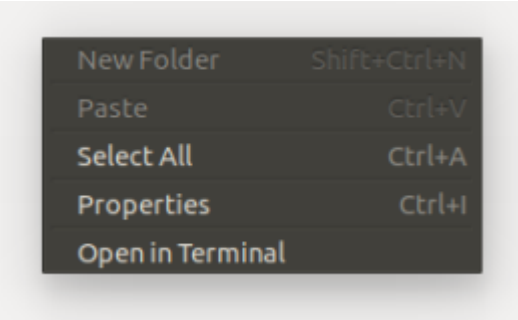
6.Connecting power supply

Connect the IBT2 driver to **12v** LIPO battery(3s) with proper XT-60 connectors and switches then place it in safe manner and connect the Nvidia with onboard power supply of **16 v** LIPO battery (4s) with XT-60 and DC 19v convertor jack. Try providing the batteries with buzzer for safety. Arduino will be power through jestson via the cable.

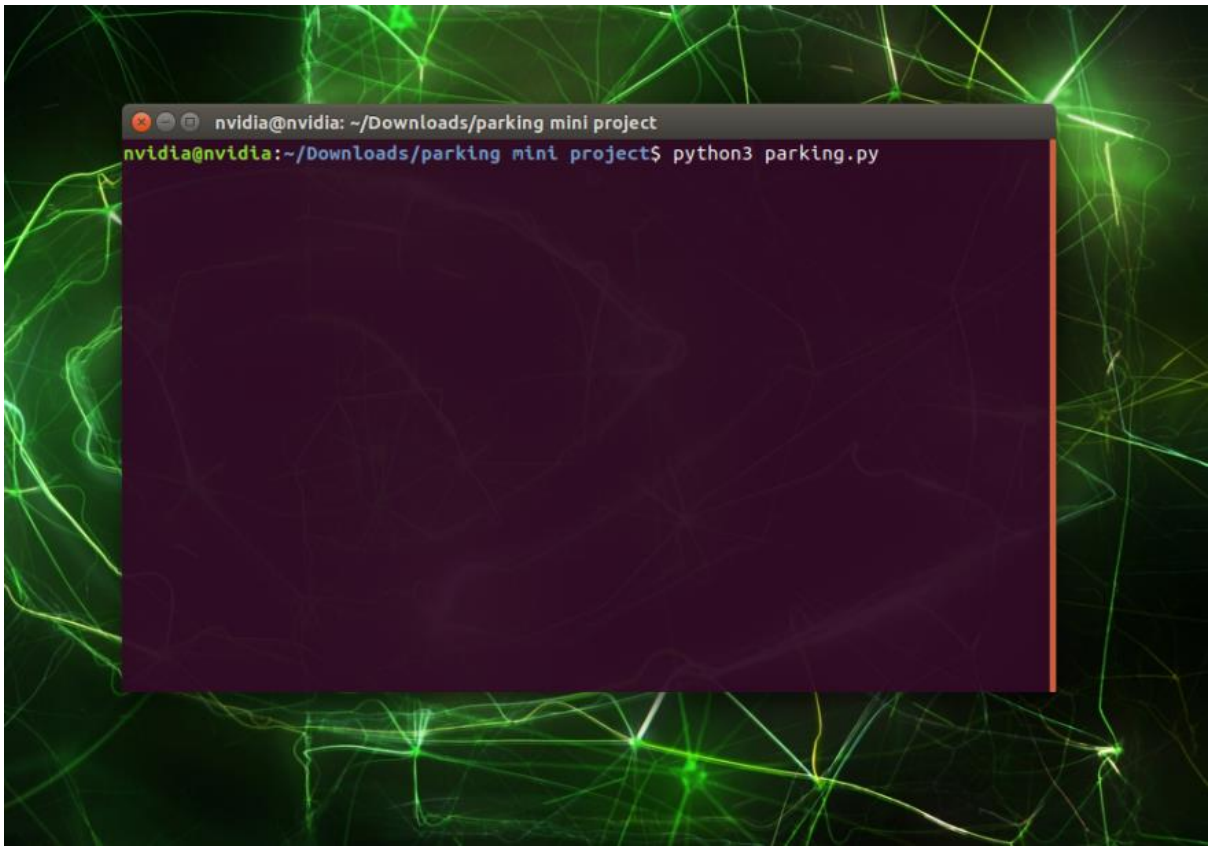
7. Executing the code

Once the kit is assembled as per the instructions, switch off the supply to motor.

Place the provided files in folder in the directory of own choice and open it in Terminal



Type "python3 parking.py" and click enter to execute the python file.



Draw a black/white line on the floor, place two boxes adjacent two it with the parking space between two boxes.

Place the car in beginning of line and switch on the motor power supply.

The car will start searching for space and once it detect the space it will park automatically. If there is no space the car will continue to follow the line in search of parking space.

8. To enable CAN bus on Nvidia Jetson TX2 Developer Board

I. Requirements

- Transceivers (up to 2 for 2 CAN bus ports). Waveshare transceiver is used here.
- Jumper wires
- 120 ohm resistor (optional)

II. How to wire

In Jetson TX2 dev board, it comes with 2 CAN controllers. So, one can have 2 different CAN ports by using 2 transceivers connecting to these CAN controllers. The following instructions wills show how to enable both CAN ports and let them communicate with each other.

Locate J26 / GPIO Expansion Header and its pin. You can find the spec of these pins on the Jetson TX2 Dev Kit Carrier Board Specification document at https://developer.nvidia.com/embedded/downloads , all the documents are free to download but requires a login.

=== Connecting Jetson & Transceiver ===

Can0	Can1
Pin 5 ------CAN RX Pin 7 ----- CAN TX GND ----- GND 3.3v -------3.3v	Pin 15 ----- CAN RX Pin 17------CAN TX GND ------ GND 3.3v ------- 3.3v

III. Commands

If using Jetpack 3.2 or later, you won't have to worry about enabling header for mttcan, this seems like a missing thing from earlier Jetpack version.

In a terminal, type following commands to setup CAN channels and their bitrate. The bitrate here is 500K but one can change to other numbers per your need.

```
sudo modprobe can
sudo modprobe can_raw
sudo modprobe mttcan
sudo ip link set can0 type can bitrate 500000
sudo ip link set up can0
sudo ip link set can1 type can bitrate 500000
sudo ip link set up can1
```

In another terminal, check if the CAN bus are set up

```
ifconfig
```

III. Example send & receive can message

For example, you connect both can0 to can1 together. You want to send a sample message from can0 and receive from can1.make sure connections are given as per the instruction on two Transceivers.

In one terminal, run:

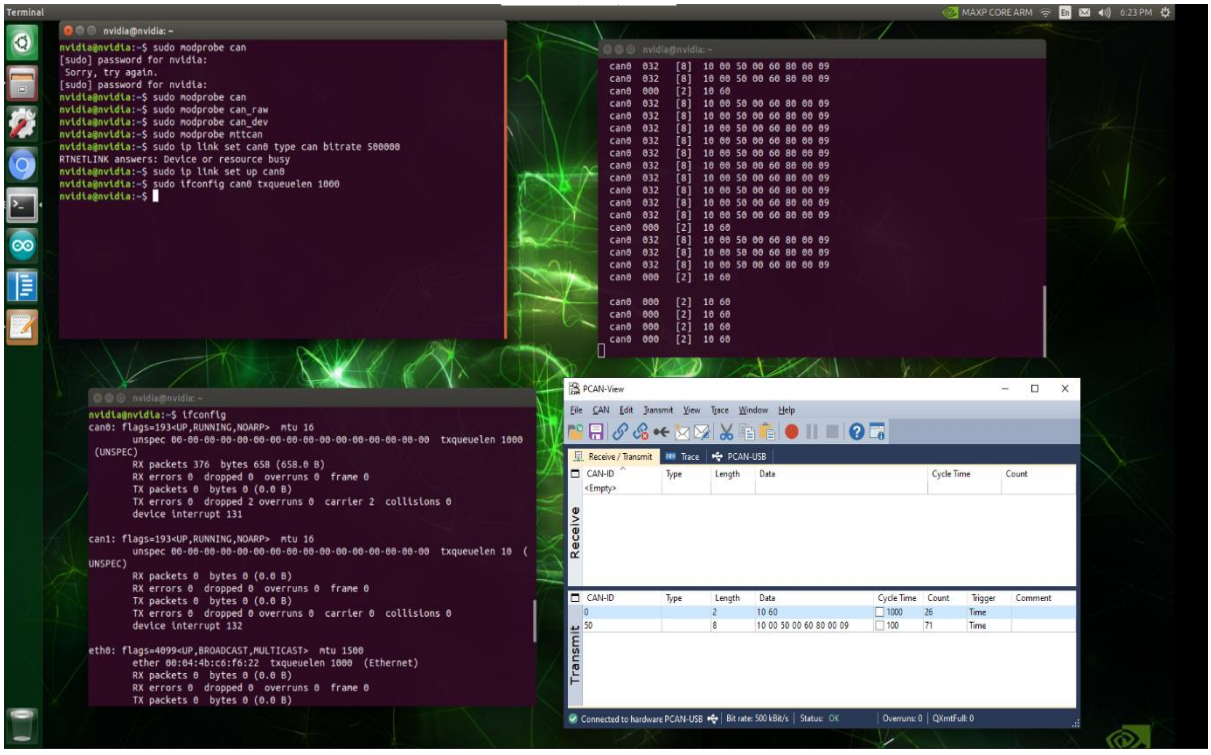
```
candump can1
```

In another terminal, run, for example:

```
cansend can0 666#1122
```

If the message was sent successfully, you will see the message on can1's terminal

Peak CAN dongle can also be used to test the same.



IV. Possible Error

- If any error for CAN try Re-installing can-ultils to send and receive CAN message over terminal

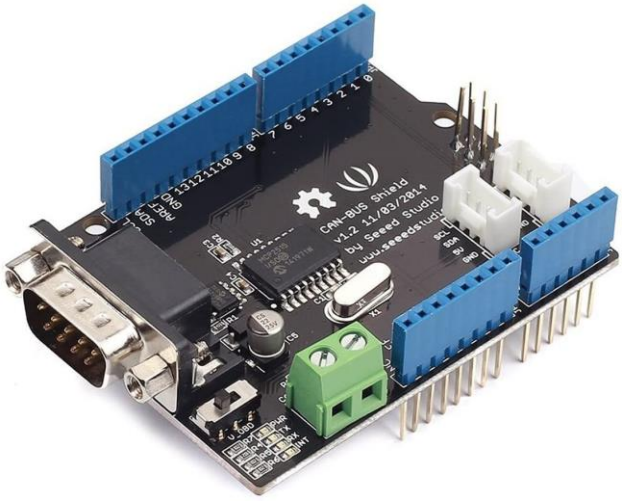
```
sudo apt-get install can-utils
```

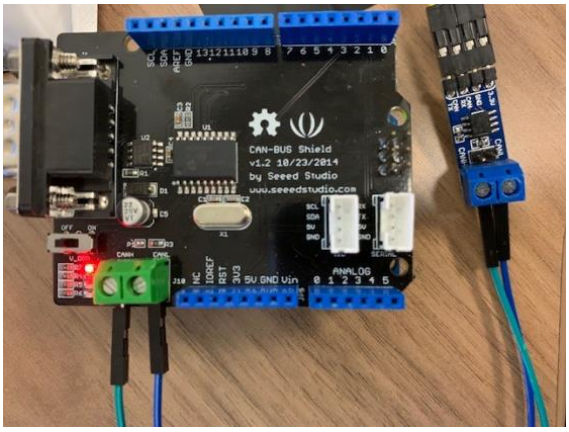
- Sometimes buffer space error might occur, After you enable the can0 interface with sudo ifconfig can0 up, run

```
sudo ifconfig can0 txqueuelen 1000
```

- The transceiver does not come with a terminal resistor (120 ohm) built-in. If the other CAN device you connect to also does not have the terminal resistor, you need to insert one into the system to get the message flow through. Otherwise, all messages would go into error state.

To use can with Arduino additional CAN Shield is required Connect the CAN_L & CAN_H of the CAN shield to the CAN_L & CAN_H of the second device and attach the shield to Arduino.





Now upload parking_can.ino into Arduino which can be found in additional files folder.

Then in the terminal run python3 parking_can.py and follow the same procedure.