Software Defined Vehicle Safety System

Table Of Contents

Demo Video

COMP4900 Project Team 12 SDV Demo

This is the Demo of Team 12's SDV Project

https://www.youtube.com/watch?v=Rbw_1RU8jmo

Participants

- Kateryna Besida: keterynabesida@cmail.carleton.ca
- Tom Lam: tomlam@cmail.carleton.ca
- Minh Thang Cao: minhthangcao@cmail.carleton.ca

Contributions of participants:

- **Code**: Participated equally in algorithm development (link to the repo)
- **Report**: Equally contributed to description of algorithm, system design and results

- **Demo**: Have equal share in creation of demo presentation

Directory structure

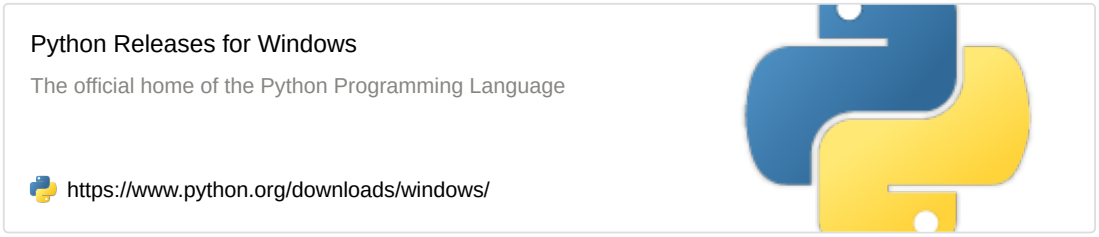
```
❯ tree --gitignore
.
├── Makefile
├── README.md <--- This file
├── frontend.sh <--- The frontend setup script
└── src/
    ├── SDV.c <--- Store the main() function
    ├── backend/ <--- The C backend that handle all logic
    │   ├── includes/ <--- All header files
    │   │   ├── CommDispatcher.h
    │   │   ├── CommListener.h
    │   │   ├── View.h
    │   │   ├── actuators.h
    │   │   ├── commons.h <--- Stores all common structs + configurations
    │   │   ├── simulator.h
    │   │   ├── tests.h
    │   │   └── utils.h
    │   ├── actuators/ <--- All actuators (ACC + ABS + Manual Driver + State machine)
    │   │   ├── abs.c
    │   │   ├── acc.c
    │   │   ├── control_systems.c
    │   │   ├── dispatcher.c
    │   │   ├── manual_driver.c
    │   │   └── pseudocode.py
    │   ├── comm/ <--- The module that handle backend <-> frontend communication
    │   │   ├── CommDispatcher.c
    │   │   └── CommListener.c
    │   ├── simulator/ <--- The module that handle environment simulation
    │   │   └── simulator.c
    │   ├── tests/
    │   │   ├── testActuators.c
    │   │   ├── testComm.c
    │   │   ├── testSimulator.c
    │   │   └── testView.c
    │   ├── utils/
    │   │   ├── data.c
    │   │   └── thread.c
    │   └── views/ <--- The module that handle printing on the backend
    │       └── View.c
    ├── frontend/ <--- The Python frontend that handle the GUI
    │   ├── Display.py
    │   ├── ViewDispatcher.py
    │   ├── ViewListener.py
    │   └── requirements.txt
    └── utils/ <--- All python utilities
        ├── CSVParser.py <--- Produce graphs from csv files
        └── csv_files/ <--- Store all csv files

12 directories, 33 files
```

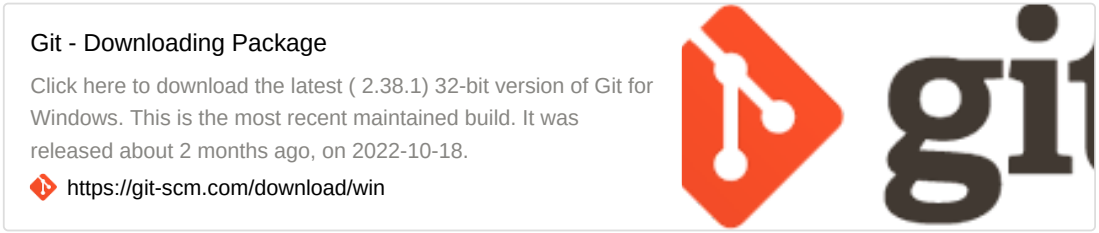
Setup instruction

Please contact us for any problem with your setup

1. Prerequisites:
    - Linux / MacOS:
        - `python3` ≥ 3.10
        - `pip3` ≥ 22.3.1
        - `python-tk`: Perferrably from your package manager
        - `virtualenv` (only if needed - usually included with Python but please install if not)
    - Windows
        - python ≥ 3.10 (pip should already be included with python windows installation)



        - git bash: This is required for running setup script



2. For Linux / MacOS hosts, feel free to use any terminal emulator of your choice that run bash or zsh. For Windows hosts, it's crucial to use **git bash** for the following steps
3. Setup the IP addresses to allow the Python front-end to establish a socket connection with the SDV backend
    a. Change the working directory to the project source

```
$ uname
Linux
$ cd 4900SDV/
```

b. Copy the `.env.template` file to `.env`

```
$ cp .env.template .env
```

The copied `.env` file initially looks like this

```
$ cat .env
LISTENER_IP=1xx.xx.xx.x
DISPATCHER_IP=1xx.xx.xx.xxx
```

c. Replace the `LISTENER_IP` with your **host IP address** on the perspective of your target.

- Can be done by running `ifconfig` in the QNX target terminal and take the `inet` address of your host under the `wm0` device. In this case it's `192.168.56.101`

```
# uname
QNX
# ifconfig
wm0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
  capabilities rx=7<IP4CSUM,TCP4CSUM,UDP4CSUM>
  capabilities tx=3f<IP4CSUM,TCP4CSUM,UDP4CSUM,TCP6CSUM,UDP6CSUM,TSO4>
  enabled=0
  address: 52:54:00:5d:43:bb
  media: Ethernet autoselect (1000baseT full-duplex)
  status: active
  inet 192.168.56.101 netmask 0xffffff00 broadcast 192.168.56.255
  inet6 fe80::5054:ff:fe5d:43bb%wm0 prefixlen 64 scopeid 0x11
```

d. Replace the `DISPATCHER_IP` with your **QNX VM target IP address** on the perspective of your host.

- For Linux/MacOS host, run `ifconfig` and take the address of your QNX vm. In this case it's `192.168.56.1`
  - For Virtualbox VM, the device name may start with `vboxnet`
  - For VMware VM, the device name may start with `vmnet`

```
$ uname
Linux
$ ifconfig
... Other network devices
vboxnet0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.56.1  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::800:27ff:fe00:0  prefixlen 64  scopeid 0x20<link>
        ether 0a:00:27:00:00:00  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
        TX packets 76  bytes 11955 (11.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
... Other network devices
```

- For Windows host, run `ipconfig` or `arp -a` (depends on your adapter type) in the gitbash terminal

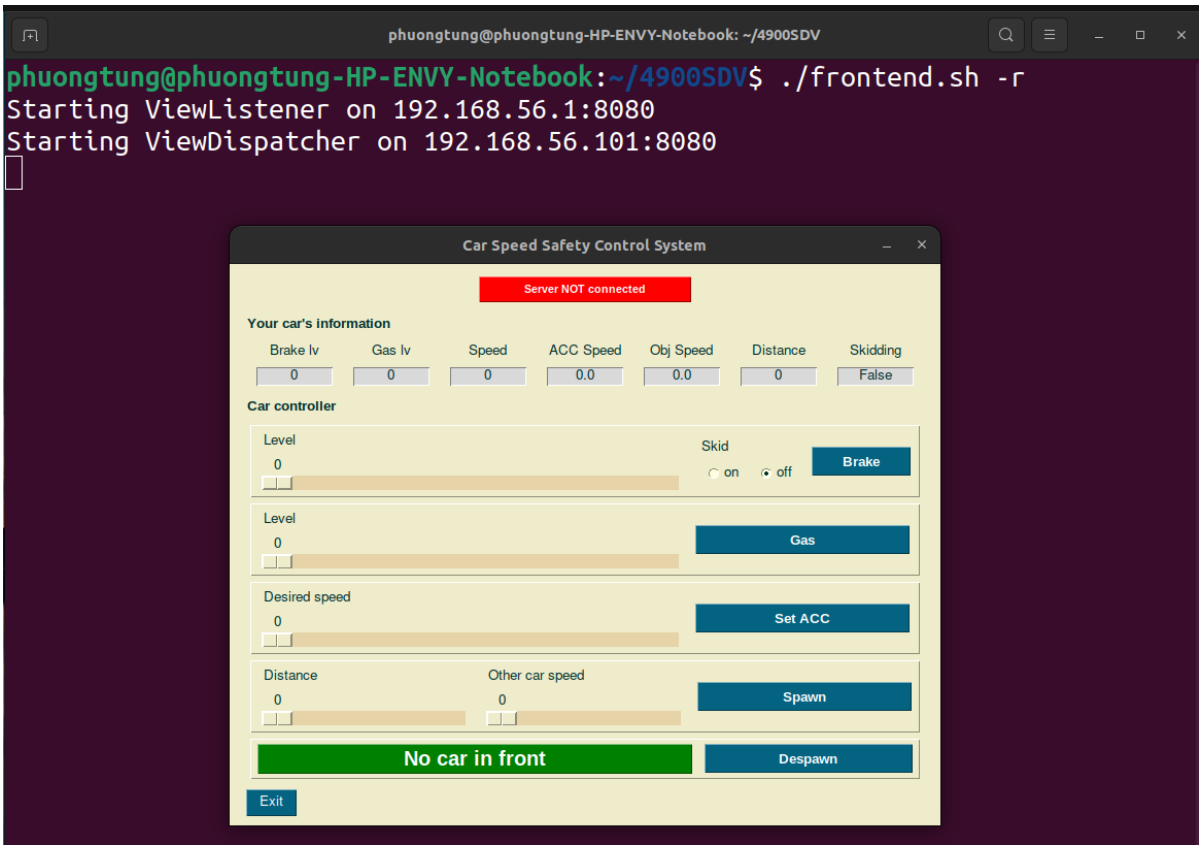
e. The resulting `.env` file should look something like this

```
$ cat .env
LISTENER_IP=192.168.56.1
DISPATCHER_IP=192.168.56.101
```

4. Build and run the frontend on your **host** terminal

```
$ ./frontend.sh -c # Clean
$ ./frontend.sh -i # Install
Installing for Linux/MacOS
... Installation logs
12147 INFO: Building EXE from EXE-00.toc completed successfully.
$ ./frontend.sh -r
Starting ViewListener on 192.168.56.1:8080
Starting ViewDispatcher on 192.168.56.101:8080
```

The GUI should pop up on your host computer like this. It will say "**Server NOT connected**" to reflect that our backend hasn't run



- Tips: If the command failed and printed this error, try swapping the `LISTENER_IP` with `DISPATCHER_IP` and try again.

```
$ ./frontend.sh -r
Starting ViewListener on 192.168.56.101:8080
Window exited
[Errno 99] Cannot assign requested address
```

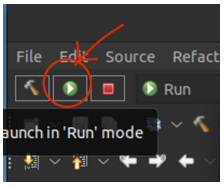
- If it still doesn't work, please redo step 3

5. Set up the IP addresses on the backend to allow it to establish a socket connection with the frontend
   a. Modify the IPs in `src/backend/includes/commons.h`
      - Replace `COMM_SERVER_ADDRESS` with the ip from `DISPATCHER_IP` from above
      - Replace `VIEW_SERVER_ADRESS` with the ip from `LISTENER_IP` from above
      - The resulting `src/backend/includes/commons.h` should look something like this

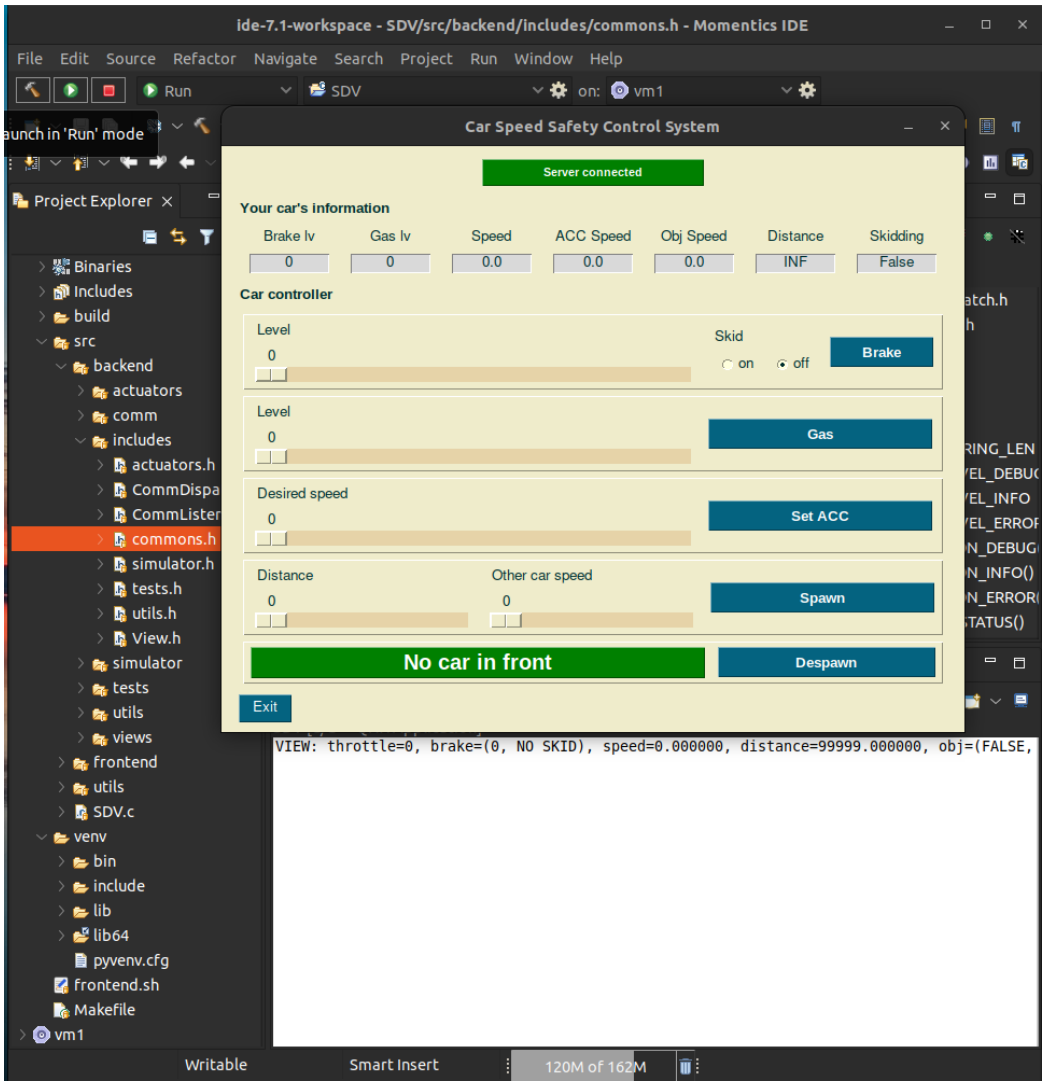
```
commons.h 3, M
SDV > src > backend > includes > commons.h > ...
42  /** Configurations */
43
44  // Socket configurations: SERVER_ADDRESS is the IP address of the CommListener server
45  //                        VIEW_SERVER_ADDRESS is the IP address of the ViewDispatcher server
46  #define COMM_SERVER_ADDRESS "192.168.56.101"      // Tom's IP
47  #define VIEW_SERVER_ADDRESS "192.168.56.1"        // Tom's IP
```

6. Build and run the SDV from the Momentics IDE

   Click on the “Launch in ‘Run’ mode” green play icon

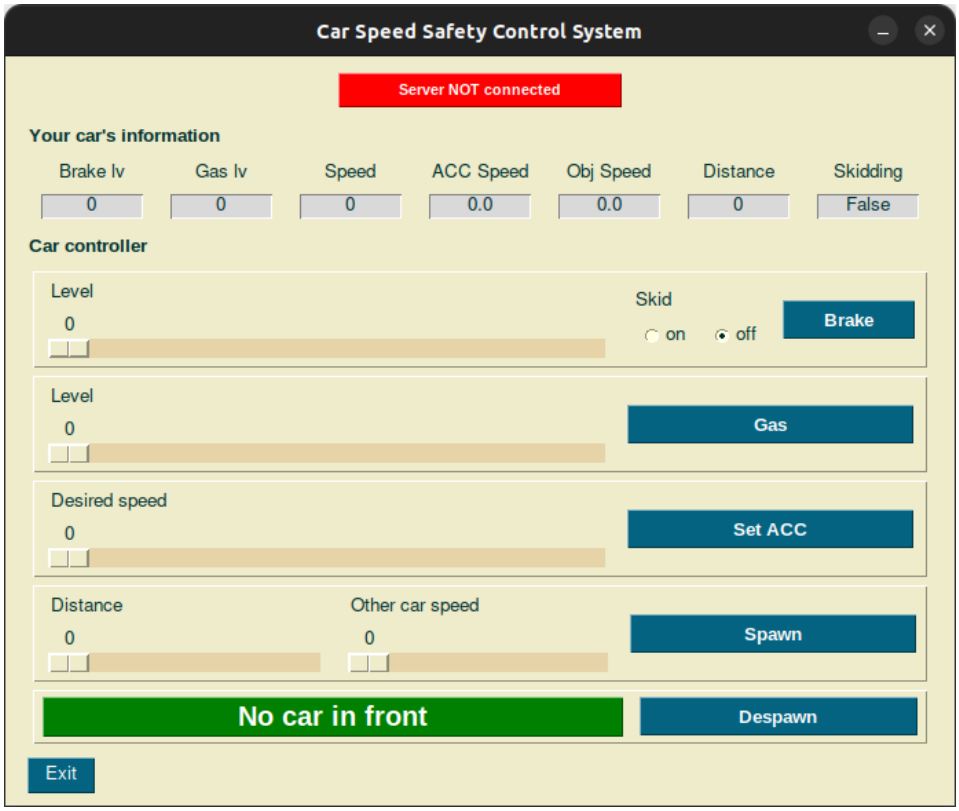


   Once the backend runs, the frontend should will show **Server connected**



If the frontend still show **Server NOT connected**, try swapping the `COMM_SERVER_ADDRESS` ip with `VIEW_SERVER_ADDRESS` ip and try running backend again

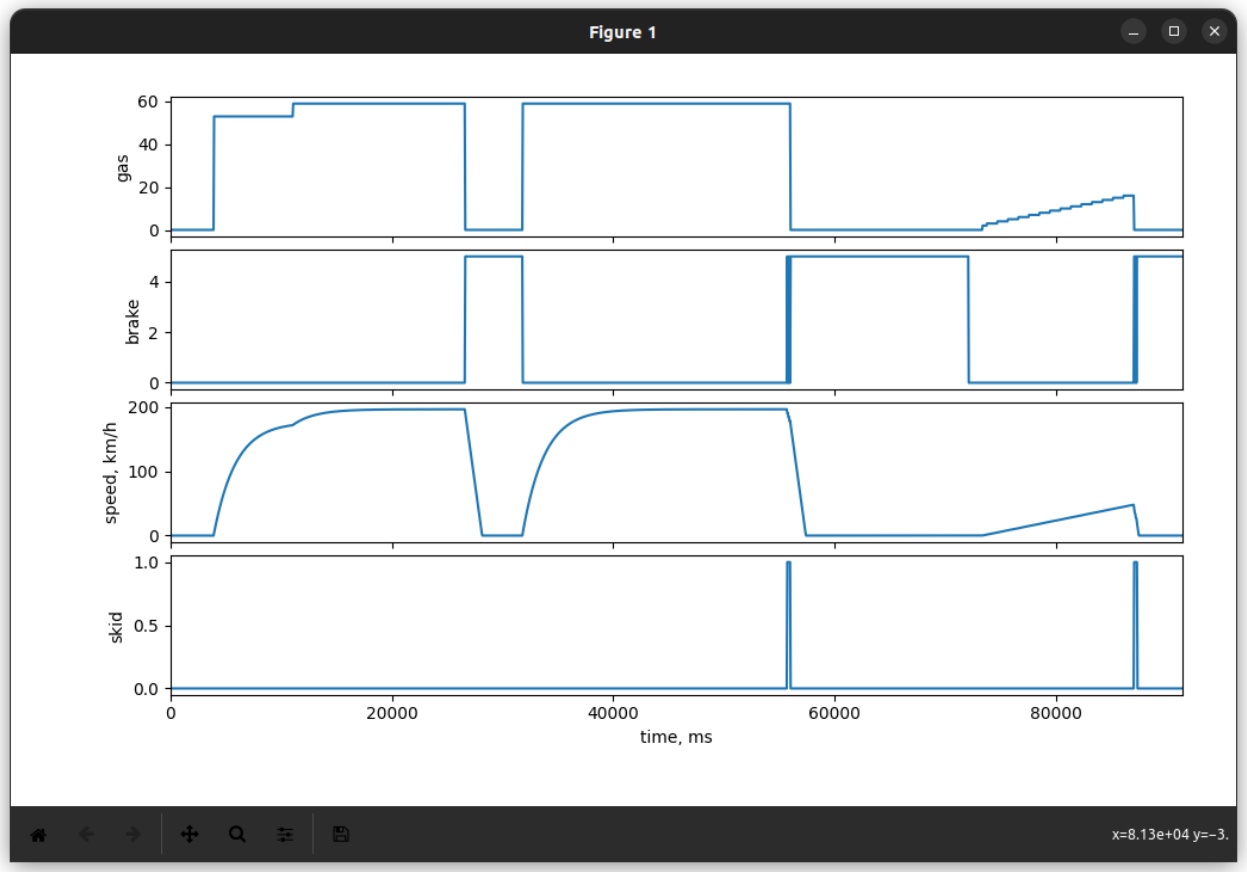
Graphical User Interface user manual:



- If **“Server NOT Connected”** - SVD is not running.
- If **“Server Connected”** - SDV is running.
- To engage the process, the appropriate buttons ****must be clicked**** (simply choosing the value available on a scroll bar will not engage the process).
- The levers (scroll bar) values are not reset automatically to 0, and have to be moved manually to the new value.
- To Exit the program properly - click the **“Exit”** button at the bottom.
- To start the car - the car should gain the speed, either by pressing the gas with the value greater than 0 (by pressing “**Gas**” button), or by applying the ACC with value greater than 0 (by pressing “**Set ACC**” button).
- To disengage the action of manual actuator (brake or gas/throttle) - the corresponding value should be set to 0.
- To disengage the ACC, the value of desired speed should be set to 0.
- To engage the Object in front the values of distance and speed should be set first and then the **“Spawn”** button clicked.
- To remove the object in front, **“Despawn”** button should be clicked.

- Pressing the **“Spawn”** button again, while the object in front is moving, will reset the object’s distance and speed to those set on scroll bars. We do not simulate multiple vehicles, as the distance sensor will only see in front.
- To simulate braking press the **“Brake”** button with the value set to other than 0.
- To simulate the slippery road conditions, select the “Skid ON” radio button with brake bar value other than 0.
- To disengage the slippery conditions, the radio button “Skid Off” must be selected and **“Brake”** button pressed.

Produce graphs from SDV output user manual



To see the graph of the saved simulation:

1. Make sure you exited the program properly by clicking **“Exit”**. This saves data to file to VM.

2. Copy the `data_log.csv` file from the `/tmp/data_log.csv` in VM to the host machine via `scp` to `~your_name_on_host_goes_here/4900SDV/src/utils/csv_files directory`.
    - Example

```
$ scp 4900vm:/tmp/data_log.csv ~/4900SDV/src/utils/csv_files
```

3. Navigate to the `4900SDV/src/utils` directory and run the following command from this directory `python CSVParser.py`

```
$ cd 4900SDV/src/utils
$ python CSVParser.py # python3 for Linux/MacOS
```

4. You will see the default graph.
5. If you want to see the different graphs, open the CSVParser.py file and change desired graphs output on line 151 to the choice available from line 136.

```
def plot_data(filename, data_names: list[str]):
    """
    Plot the data in the filename with a list of subplot data names
    All names: ["time, ms", "gas", "brake","speed, km/h", "object", "object_speed, km/h","distance, m", "skid"]
    """
```

Line 136

```
if __name__ == "__main__":
    # Put all graphs want to plot here
    # plot_data("csv_files/ACC_Manual_preemption.csv", ["gas", "brake", "speed"])
    plot_data("csv_files/data_log.csv", ["gas", "brake", "speed, km/h", "distance, m", "object_speed, km/h"])


    plt.show() # Don't modify this and please do only one show
```

Line 151