Getting Started

System Requirement

The current AI Racing Tech software (Jan 2024) is based on ROS2 Iron. As of Jan 2024, it will build on ROS 2 Humble as well

Required:

- Ubuntu 22.04 in native OS, virtual machine, or Windows Subsystem for Linux (WSL). Ubuntu 22.04 is what the race car runs on and it is strongly recommended you develop in native Ubuntu 22.04 Linux to avoid issues.
- Install ROS 2 Iron Desktop (current stack is on ROS Iron) or ROS 2 Humble Desktop.
- Alternatively, use the pre-built docker image of the entire race stack.

Optional:

- If you want to use any deep learning or a perception enabled simulator you will need a discrete NVIDIA GPU (likley an RTX 20 series or newer).
- Perception Deep Learning (Linux only):
  - CUDA 11.8 Toolkit (should be installed in `/usr/local/cuda-11.8` if you have mutliple versions of the CUDA toolkit installed or `/usr/local/cuda` if you only have CUDA `11.8` globally installed). The CUDA toolkit is an API that sits ontop of the low level CUDA graphics drivers. **Make sure you only install the CUDA toolkit and NOT the CUDA drivers that are packaged with the toolkit**. If you accidentally install the CUDA drivers as well you may black screen your computer and need to repair it from the recovery boot CLI. See Eric's linux utils repo for a convienent CUDA toolkit installer script and a bash alias to switch between different CUDA versions (works by changing the environment variables to point to the correct CUDA version).
  - TensorRT 8.6.1.6
- For Windows

- To use rviz2 in wsl2, follow the wsl2 instruction to download the latest cuda driver here, then in ubuntu export DISPLAY=:0.0, after this, rviz2 should be able to pop in windows 10

Note:

- ARM64 architecture is untested.

Install

We will assume you have ROS 2 installed and sourced in all these terminals

1. Clone `race_common`. Install `python3-vcstool`.

```
git clone git@github.com:airacingtech/race_common.git
sudo apt update
sudo apt install python3-vcstool
```

2. Import `race_common` dependencies.

```
cd race_common
make vcs-import VCS_FILE=race.common.${ROS_DISTRO}.repos
```

3. Import any other dependencies if working on a particular vehicle or simulator setup.

```
make vcs-import VCS_FILE=iac.${ROS_DISTRO}.repos # if working on IAC car
make vcs-import VCS_FILE=svl.${ROS_DISTRO}.repos # if working on SVL simulator
```

4. Install dependencies. In general if you run a VCS step post this then you need to rerun the `make rosdep-install` command

```
make rosdep-install
pip3 install environs
```

5. Install casadi.

```
source /opt/ros/<ROS_DISTRO>/setup.bash # replace ROS_DISTRO with your ROS distro
name
sudo apt install -y gcc g++ gfortran git cmake liblapack-dev pkg-config --
install-recommends
sudo apt install -y --no-install-recommends coinor-libipopt-dev libslicot-dev
cd ~/ && git clone https://github.com/casadi/casadi.git -b 3.6.4 casadi
cd casadi && mkdir build && cd build
cmake -DWITH_IPOPT=ON -DWITH_SLICOT=ON -DWITH_LAPACK=ON -DWITH_QPOASES=ON -
DWITH_OSQP=ON .. && make
sudo make install
cd ~/ && rm -rf casadi
echo export
LD_LIBRARY_PATH="${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}/usr/local/lib" >>
~/.bashrc
source ~/.bashrc
```

6. Build up to your specific use cases. In race_common, run the following.

```
make iac # if working on IAC car
make svl # if working on SVL simulator
make # if not building for any particular platform
```

NOTE: You may have to run `make rosdep-install` if CMake cannot find the `ament_cmake_vendor_package`. Whenever you VCS import a new target, `make rosdep-install` should be run.

NOTE: If you have not installed the Vimba 6.0 SDK `make iac` will fail. If you want to build without the Vimba SDK, you can run `make iac-nocam` or `make iac-debug-nocam` to avoid building packages that require the Vimba SDK.

Warning: Several packages in `race_common` will run `FINDCUDA.cmake` which will fail if your ENV variables are not set to point to CUDA 11.8. If you receive the following error while building:

```
--- stderr: image_proc
CMake Error at /usr/share/cmake-3.22/Modules/FindPackageHandleStandardArgs.> >
cmake:230 (message):
  Could NOT find CUDA: Found unsuitable version "XX.Y", but required is exact
  version "11.8" (found /usr/local/cuda-XX.Y)
...
```

then change your environment variables to point to CUDA=`11.8`. If you have multiple CUDA toolkits installed you can use the `cuda` bash alias given in Eric's linux utils repo to change the environment variables to CUDA `11.8`. Then `make clean` (`rm -rf build install log`) so the old files built with the wrong version of CUDA are removed and `make` again.

7. Copy `example.env` and rename as `race.env`. The environment variables requires further configurations, as discussed below in corresponding setups for each vehicle or simulator.

```
cp example.env race.env
```

8. The `.env` file specifies configurations for `race_common`. Modify the `.env` as needed. The comments describe for what reasons you would want to change it

```
# The TTLs you want to load. Look at src/common/race_metadata/ttls for available
```

```
options
TTL_FOLDER="LVMS_ENU_TTL_CSV"

# GPS Origin of the TTLs. It's available as the last 3 values in the first row of
the TTL header. You can find TTLs by going to the folder above and clicking on
any of the CSV files under the relevant folder
GPS_ORIGIN_LAT=36.27207268554108
GPS_ORIGIN_LON=-115.0108130553903
GPS_ORIGIN_ALT=594.9593907749116

# Simulation settings
MAP_NAME="Putnam"
NUM_CARS=1
SENSORS="GPS_LIDAR" # GPS, GPS_LIDAR, GPS_LIDAR_CAMERAS

# The Vehicle you are running. Can take the values of SVL_IAC_CAR, IAC_CAR,
HAWAII_GOKART right now
VEHICLE_NAME="IAC_CAR"

# Parameter configuration you want to load. Can take the values of IAC_LVMS,
SVL_LVMS, HAWAII_GOKART_AAIS, AW_SIM_HAWAII_GOKART_AAIS and AW_SIM_IAC_LVMS now
RACE_TYPE="IAC_LVMS"

# Whether to launch the manual control converter or RVC. Basically always set to
False
LAUNCH_MCC=False

# Whether to launch the VKS or not. For now always set to True
LAUNCH_VKS=True

# Whether to use sim time or not. Set to True if playing a ROS 2 bag
clock option or if using OSSDC sim / SVL sim
USE_SIM_TIME=False

# Whether to launch UDP nodes on either side of the connection. This is only
needed if you want to send joystick messages to another computer in your network
and receive telemetry back from that computer (Like communicating with the IAC
car during it's run)
LAUNCH_UDP=False

# Whether to launch Z1 or not. If you want to run it set to True
LAUNCH_Z1=False

# Whether to launch ghost car or not. If you want to test against a ghost car set
to True. Keep LAUNCH_PERCEPTION to False in this case
LAUNCH_GHOST_CAR=False

# Whether to launch race control or not. Only set to False if you are receiving
race control from some other source like My Laps
LAUNCH_RACE_CONTROL=True

# Whether to launch perception or not
LAUNCH_PERCEPTION=False

# Whether to launch localization or not. For now always set to False it does not
do anything
LAUNCH_LOCALIZATION=False
```

```
# Launches a utility node that launches boundaries and current trajectory
LAUNCH_BPP=True
```

Next Step

Time to test on some simulations:

- Developing on the Autonomous Stack
- Running on OSSDC simulator
- Running on Autoware simple planning simulator