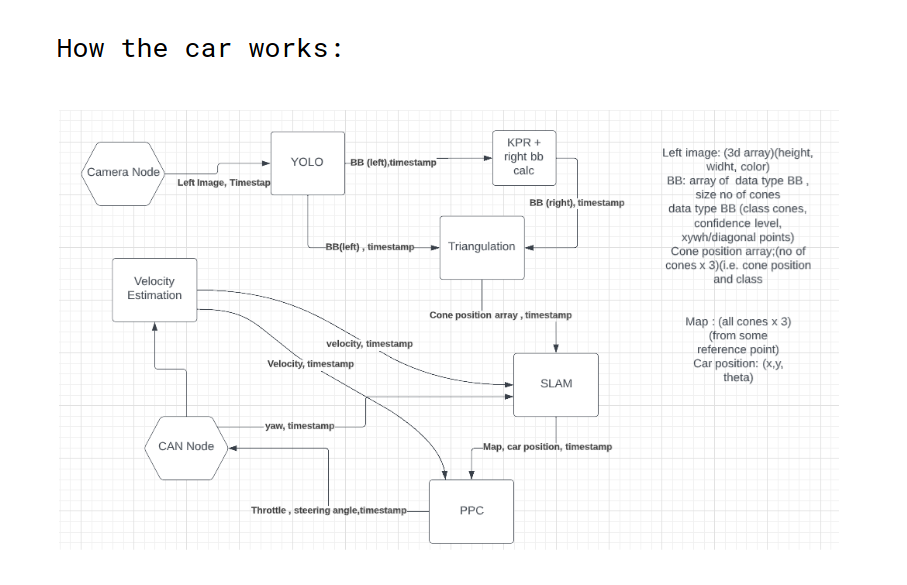
**SYSTEM INTEGRATION**

**Instructions**

* Parts of the module highlighted in green are “checkpoints” and trainees are required to update the following task sheet once every checkpoint is completed.
* Documentation is a must for every checkpoint and trainees are required to create a google doc sheet where they document their learnings, errors encountered and doubts. This google doc must be made accessible to “anyone with link” and the link for the same must be updated on the task sheet.
* Feel free you are encouraged to ask doubts to JDEs if you feel stuck or want to understand a topic better
* Keep in mind, performance in modules is how the team will judge your abilities and effort to assign subsystems once the time comes
* Have a fun learning experience!

As this is your last trainee module, you shall have a good idea of what each and every subsystem does. What your job is as a system integration engineer is that you have to ensure efficient and fast communication between all of the subsystems as well as sensors and actuators. You are the first line of control between sensors and actuators and our Autonomous system.

By now, you shall know what most of the parts of the above chart mean. One thing you might be unsure about is CAN. We will cover that in the later parts of this module. 

Checkpoint 1:

You need to:

1. List out the datatypes of each of the signals that are being transferred.

2. In reality, these signal transfers suffer from the problem of time lags. How would you deal with time lags? And how will you deal with the difference in the frequency of the sensor data?

**ROBOT OPERATING SOFTWARE (ROS)**

But how exactly do you handle such an intricate network of nodes and ensure that everything is working at the same time? How do you take inputs from the sensors and feed them into your nodes (After all they are just electrical signals)? This is where ROS becomes your best friend.

It creates a network wherein there is no end-to-end communication between any 2 nodes. A node publishes a message to a topic. Now this message can be read by any node that subscribes to the topic. You can think of it as a new broadcaster. News reaches the news station from a reporter and this news can be seen by anyone who wants to, using their television. As far as the sensor problem is concerned, ROS is a very widely used mechanism and all sensors have ROS drivers publishing to a topic.

Use this to brush up your ROS concepts:

[Galactic](https://docs.ros.org/en/galactic/Tutorials/Beginner-Client-Libraries.html)

[Humble](https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries.html)

Checkpoint 2:

You are given a rosbag zip file [here](https://drive.google.com/file/d/1hz6MFXTMdEczK29tWAmj2V52xBhWloB_/view?usp=sharing). Download it and unzip it.

You will see that there are 2 stereo cameras (zed) placed left and right. You are receiving data corresponding to the left camera of each stereo cam.

1. Figure out which topics are publishing image files from zed.
2. Your task is to synchronize the data so that they are published at the same time. (Receiving time is approximately the same. Look into how you can improve it).
3. There are 2 ways you can d o it. One is using a built-in function and the other is manually. Do both.

**CAN Introduction:**

Our end-goal is to make all our code run onto this beautiful vehicle, the ADS-DV:



[Video](https://www.youtube.com/watch?v=KyW8mgNWLnk)

To interface with this car, you have to use something known as CAN. By interface, we mean communicating with the actuators. Which in turn means giving signals to actuators and also taking feedback from them (think about why feedback is important and include it in your **documentation**) along with the general status of the car which you will learn more about in later parts of your tenure.

[Here](https://www.youtube.com/playlist?list=PLERTijJOmYrApVZqiI6gtA8hr1_6QS-cs) are a few videos that will give you a general idea of what CAN is. Do not watch all of them, just get a rough overview of what it is.

Refer to its online documentation for more details and feel free to contact us if you get stuck somewhere.

Yet, there is more to Sys Int than just Sys Int.

**SIM DEV**

As we have been emphasizing again and again in your recruitment paper, interview and the introductory meet, Simulation development is a very important part of the work we do here in the Driverless subdivision. Since we do not have the ADS DV, there are 2 ways to test if our code actually works—

Simulations



And the bot:

We use Gazebo as our simulator. The only rule about sim dev is that “The simulator closest to the real world is the best simulator.” (Cringe af). But true af too.

Go through these [tutorials](https://classic.gazebosim.org/tutorials) to get an idea of what gazebo actually does.

Checkpoint:

You will see that the tutorial also tells you how to model a ‘Velodyne VLP32’ lidar. However, what we use is ‘Velodyne Puck VLP16’. Your task is to model it. You might find the technical details over [here](https://velodynelidar.com/wp-content/uploads/2019/12/63-9229_Rev-K_Puck-_Datasheet_Web.pdf). Remember to make it as precise and true to the datasheet as possible.

**– - - HAPPY LEARNING ☺ - - -**