Documentation: publisher and subscriber model

The model that is used to exchange data between the autonomous system and the simulator is called the ‘publisher/subscriber model’.

It is a server and a client that both connect to bidirectional socket over TCP, they can both send and receive data.

To do so, we do the following:

The publisher side:

1- We create an object (let's call it car)
2- We serialize it using Json and we store it in a variable (call it carObject)
3- We convert the serialized object (carObject) to a byte array
4- We bind our socket to a predefined port (let it be 10101)
5- We send it to all clients that are connected to the same socket as the publisher (broadcast it)

The subscriber side:

1- We create an object ‘car’ (the same object we want to receive with the same attributes)
2- We connect to port 10101 (connect to publisher)
3- Receive data and store it in a byte array
4- We convert it back to a Json object (call it recievedData)
5- We de-serialize the Json object and we store it in our object ‘car’(the one on the subscriber’s side)

Things to watch out for:

- Size of the data that is being sent
- Size of the data that is being received
- Use the same encoding on both sides
- When you receive data you have to handle it yourself (create a call back method for that)

Where can you see this in our project:

- ./scripts/communication/publisher.cs

    - In method ‘Start’
        - Data is received
        - Size is calculated
        - Data is received
        - Data is read
        - Callback method is called with data as its parameter

- ./scripts/communication/interfaceController.cs
    - In method ‘Start’
        - A new subscriber is created
        - Method ‘onControlTarget’ is used as its call back method
    - Method ‘‘onControlTarget’ is the callback method that is called when the subscriber reads data
        - It converts data to json
        - It de-serializes the json object into ‘*ControlResultMessage*’ object
        - It then calls a method ‘*ApplyControlResult* ’ that uses the data to control the car

- ./scripts/car/CarController.cs
    - In method ‘*ApplyControlResult* ’
        - This method takes the data that was already handled (de-serialized and converted) and uses it to control the car

Examples of that Model:

In “./LidarCameraSimUnity/sandboxeProjects/” you can find a project that implements this model with a python script that acts as a publisher

P.S: The data size is hardcoded so when you send data choose a speed between 0-9 and a yaw_rate between 10-90 (or maybe try to make the dataSize dynamic)