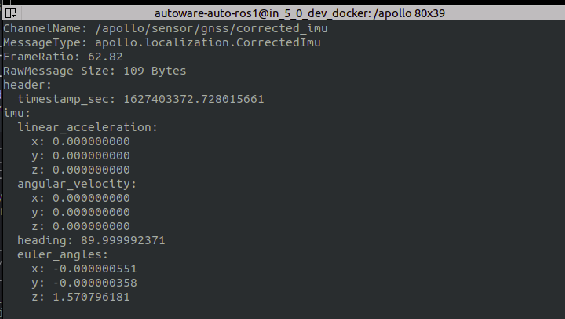
Tutorial To Extract Cyber Channels data (“From Cyber Monitor”).

Channel Extraction Example: For **corrected\_imu channel**

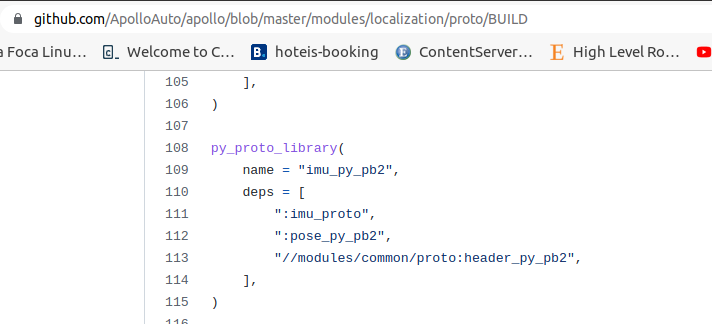
1 – Go to the folder **apollo/cyber/python/examples** and create a copy of the file **listener.py** inside this folder. Now rename this file to “**sensor.py”,** in this example to **corrected\_imu.py**

2- Open the file **corrected\_imu.py** and modify the the line of code where it is “test\_node.create\_reader”, change the following channel: from **channel/talker** to ***apollo*/sensor/gnss/CorrectedImu.** This channel name can be easily found launching the command $**cyber\_monitor**  or $**cyber\_channel list** in terminal

****

3- Still workig on the line “create\_reader..” of this code, change the MessageType name from **ChatterBenchmark,** to the MessageType displayed by the desired channel, which wishes to extract the data. In this case it is “**CorrectedImu**

4- Identify the library where the name of the message above is imported. This library can be found in the **BUILD** file contained into the path **modules/localization/proto,** then must to look for the line of code written **py\_proto\_library** and the desired sensor, in this case it is called **imu\_py\_pb2**

****

5- After identify the libary that contains the specified message type, it must import this libary in the code which will extract this message Type. In this case must be included in the module **corrected**\_**imu.py,** the line below:

**from modules.localization.proto.imu\_pb2 import CorrectedImu**

It is important to notice here, that the word **py** was removed between the words **imu\_pb2.** This is due some build binary feature or other Apollo team compilation nomenclature…And the word “py” must be removed for all sensors: sensor\_py\_pb2

6 – To debug and visualize if the code is working, and the data is being catched by the callback function, just use a print statement**:**

print(data.imu.euler\_angles.x) the attributes “imu.euler\_angles.x” can be taken acessing the cyber\_monitor channel to visualize the messages attributes the sensor provides.

7- Now it is required to modify the **BUILD** file located in the same folder where the **corrected\_imu.py** was put. In this case I put inside a created folder ***cyber/*python/cyber\_py3/channels\_data\_extraction.** It is mandatory to include the block of code below, which will build the message type locating its library:

py\_binary(

name = "**corrected\_imu**",

srcs = ["corrected\_imu.py"],

deps = [

"//modules/localization/proto:imu\_py\_pb2",

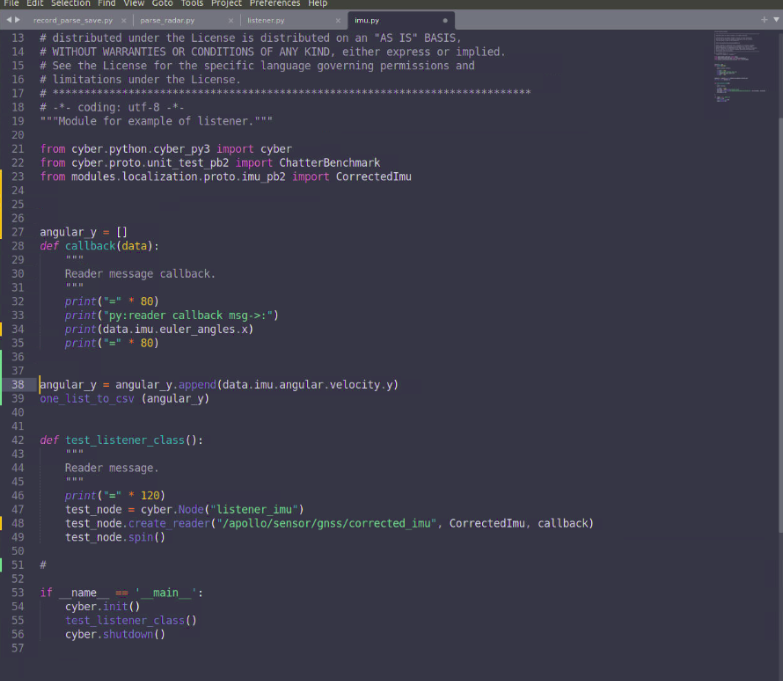
"//cyber/python/cyber\_py3:cyber",

],

)

8- Afterwards the user can just store the data in a List (Append) , DataFrame or like open and write a csv file. This code conversion for csv is available on **channels\_extraction.py** file and extracts data for Odometry and Imu attributes.

9- The corrected\_imu.py file will have at the end this sketch:



10- Finally the last step is Build these modules. In case for just build the corrected\_imu.py

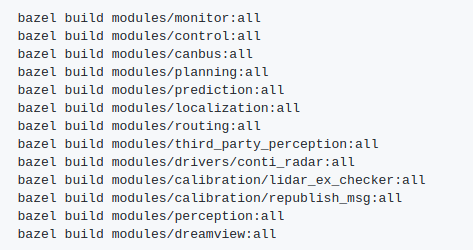
the command is below:

$**bazel build cyber/python/cyber\_py3/examples:imu**

However this usually builds this module and deconstruct the general build (Dreamview and othe modules). So it is required afterwards build again these modules with:

**$./apollo.sh build\_opt\_gpu**

Or then build all the cyber module separately:

dfd

In this case

**$ bazel build cyber/python:all**

11- Finally the LAST STEP, run the module!!!

$ **./bazel-bin/cyber/python/cyber\_py3/channels\_data\_extraction:corrected\_imu**

**Corrected\_imu i**s the name you choose in the **BUILD** file.

And you should see the images printed on the screen!

