Sparkfun OpenLog Artemis IMU Installation, Setup, and ROS Integration Guide:

ROS Package (by: Francis Le Bars): https://github.com/lebarsfa/razor_imu_9dof
(The package was built upon the previous Razor model, so do not worry about the "Razor" name, it will work for the Openlog Artemis :))

Flashing Razor_AHRS Firmware:

Requirements:

- External Machine to flash software to IMU (Jetson aarch64 does not support the Arduino IDE tools necessary for successfully completing this)
- Arduino IDE
- USB-C cable

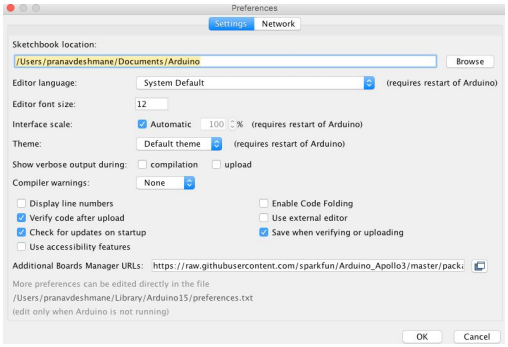
First, we need to flash the Razor_AHRS firmware to the IMU board. This will allow the ROS package to read in the data in the proper format and contains the logic for fusing the three sensor readings (accelerometer, gyroscope, and magnetometer) and calibrating them.

1. Download CH340 drivers necessary to accept OLA Artemis IMU if not already installed.
    a. To check for drivers, plug in IMU and run command: **ls /dev/tty* *could be USB0 or ACM0* (Linux), ls /dev/cu* (Mac) *could be similar to /dev/cu.wchusbserial1450,* or check Ports from Device Manager (Windows)**. Make sure whether Artemis IMU can be read, if not follow instructions at the link below to download drivers and verify connection:
    b. https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers#introduction
    c. If struggles continue, make sure to try different usb cable/ports, check patch installation procedure (linux), and try restarting the computer (this actually worked for me)

2. Install Arduino IDE: https://www.arduino.cc/en/software
    a. Open Arduino IDE and select Tools->Port that the IMU is found on (ex. /dev/cu.wchusbserial1450), and open the serial monitor. The configuration menu and the readings of the IMU should be printed if the ch340 driver was successfully installed.
    b. Follow the instructions here to set the timestamp (this may cause errors if not set in the future): https://learn.sparkfun.com/tutorials/openlog-artemis-hookup-guide/configuration

3. Download Sparkfun Apollo3 board definition using the Arduino IDE board manager.

a. Open Preferences and paste the following url into the “Additional Board Manager URLs” and hit “OK”: https://raw.githubusercontent.com/sparkfun/Arduino_Apollo3/master/package_sparkfun_apollo3_index.json

i. 

b. Navigate to the board manager through Tools->Boards->Board Manager. Search “Sparkfun Apollo3 Boards” (this will show up only if the previous step followed successfully) and install the board definition.

4. Download the following repository provided by Francis Le Bars (link also at top). This same repo will serve as the ROS package for the Artemis IMU and contains the code necessary for fusing the three sensor readings (accelerometer, gyroscope, and magnetometer) and calibrating them: https://github.com/lebarsfa/razor_imu_9dof.

5. Using the Arduino IDE, open the Razor_AHRS.ino file located within the src/Razor_AHRS folder from the repository above.
    a. Uncomment the line `#define HW__VERSION_CODE 16832 // SparkFun "OpenLog Artemis" version "DEV-16832"`, to select the Artemis as the appropriate board.
    b. Then, navigate to Tools->Boards and select the Sparkfun Apollo3 Boards -> Sparkfun RedBoard Artemis ATP board, compile and flash the software to the board.

Setting up ROS package on Jetson:

Now that the software is flashed successfully, we can set up the ROS package on the Jetson.

1. On the JNX clone the repository from Francis Le Bars above and follow the setup steps in the README for the catkin_ws setup. However, make sure to clone the repository in the link above, **it is miswritten in the README.** The appropriate commands are below.

```
cd ~/catkin_workspace/src
git clone https://github.com/lebarsfa/razor_imu_9dof.git
cd ..
catkin_make
```

2. Navigate to the config folder, copy the razor.yaml to a new “my_razor.yaml” file. Open and change the port in my_razor.yaml to the appropriate port the IMU is being read into (ex. /dev/ttyACM0). Here is also where the calibration values can be modified after the calibration process is completed.

```
roscd razor_imu_9dof/config
cp razor.yaml my_razor.yaml
```

3. Source the workspace and catkin make to generate the necessary Razor dynamic files

```
cd ~/catkin_workspace/
source devel/setup.bash
catkin_make
```

4. Now you are ready to run the IMU node publisher to output your readings into the /imu topic.

```
roslaunch razor_imu_9dof razor-pub.launch
```

5. Check topics to see if /imu topic is being published on

```
rostopic list
```

6. Echo the imu topic to check that the readings are properly published

```
rostopic list razor_imu_9dof razor-pub.launch
```

7. For visualization, the visual library needs to be installed using the bionic branch of the visual library also written by Francis Le Bars. This is because the python-visual library is now deprecated. These steps are in the visualization guide and as of now only work using a Linux machine running ROS Noetic and Python 3.

This guide is written to utilize the external machine approach for the initial flashing of the software. We are still looking to see if ideal JNX complete flashing is attainable without arm64 tool architecture errors within Arduino IDE

- Update: Dusty Franklin of Nvidia and Francis Le Bars himself have advised to simply flash with an external machine first.

Calibration:

Reference Calibration Guide linked above.