# Live plotting of Sensor data (Documentation).

## Overview

This project is a real-time data acquisition, processing, and visualization system. It consists of three Python scripts that interact with each other to generate, transmit, and plot sensor data. The components are as follows:

* **‘generator.py’** : Simulates sensor data and publishes it to a ROS topic.
* **‘server.py’** : Acts as a bridge, receiving the ROS topic data and forwarding it to a socket client.
* **‘plotter.py’**: Receives the data via a socket connection and visualizes it using Matplotlib.

## Script Details

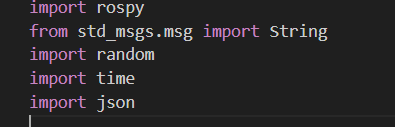
### Dependencies.

* Ubuntu OS ( 20.04)
* ROS 1 (noetic) { **FULL VERSION** }
  + - * + While running the server socket code on computer, if any error occurs remove the ROS and manually install the full installation.
        + For any help visit < https://medium.com/@createwithabd/ros-noetic-installation-guide-on-ubuntu-20-04-388568d24bcf >
* Wifi or ethernet connection for both client and server.

### 1. generator.py

The `generator.py` script simulates sensor data and publishes it to a ROS topic (`sensor\_data`). It creates random values for acceleration, velocity, and distance, and associates them with a timestamp. The data is published at a configurable rate using the ‘rospy’ library.

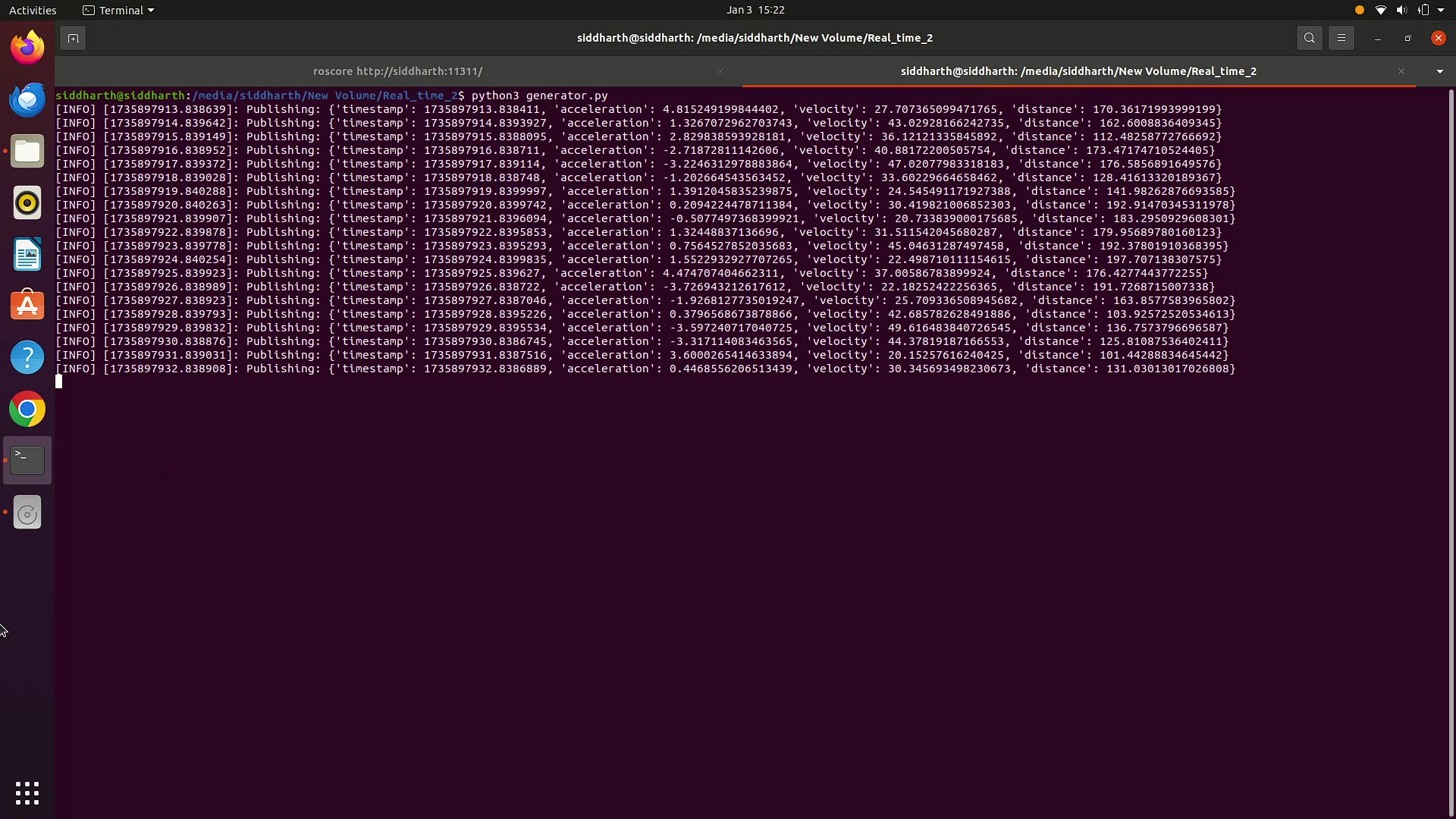
Required libraries to install for generator.py



- Data simulation with ‘random.uniform’ for values. This can be changed and it can be coded to get directly from the Car sensors.  
- JSON serialization for publishing structured data.  
- ROS integration with a publisher node.

Do not forget to open a master ROS before doing anything by using ‘roscore’.

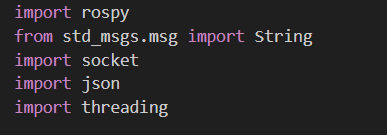
The data should publish in the ROS topic like the below image.



### 2. server.py

The `server.py` script establishes a socket server that listens for client connections. It subscribes to the ROS topic (`sensor\_data`), receives data, and forwards it to connected clients in JSON format.  
Key Components:  
- Socket server setup with Python's ‘socket’ library.  
- ROS topic subscription and data forwarding.  
- At once only one client can be connected to the server.

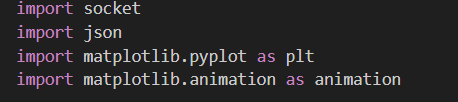
Required libraries to install are below.



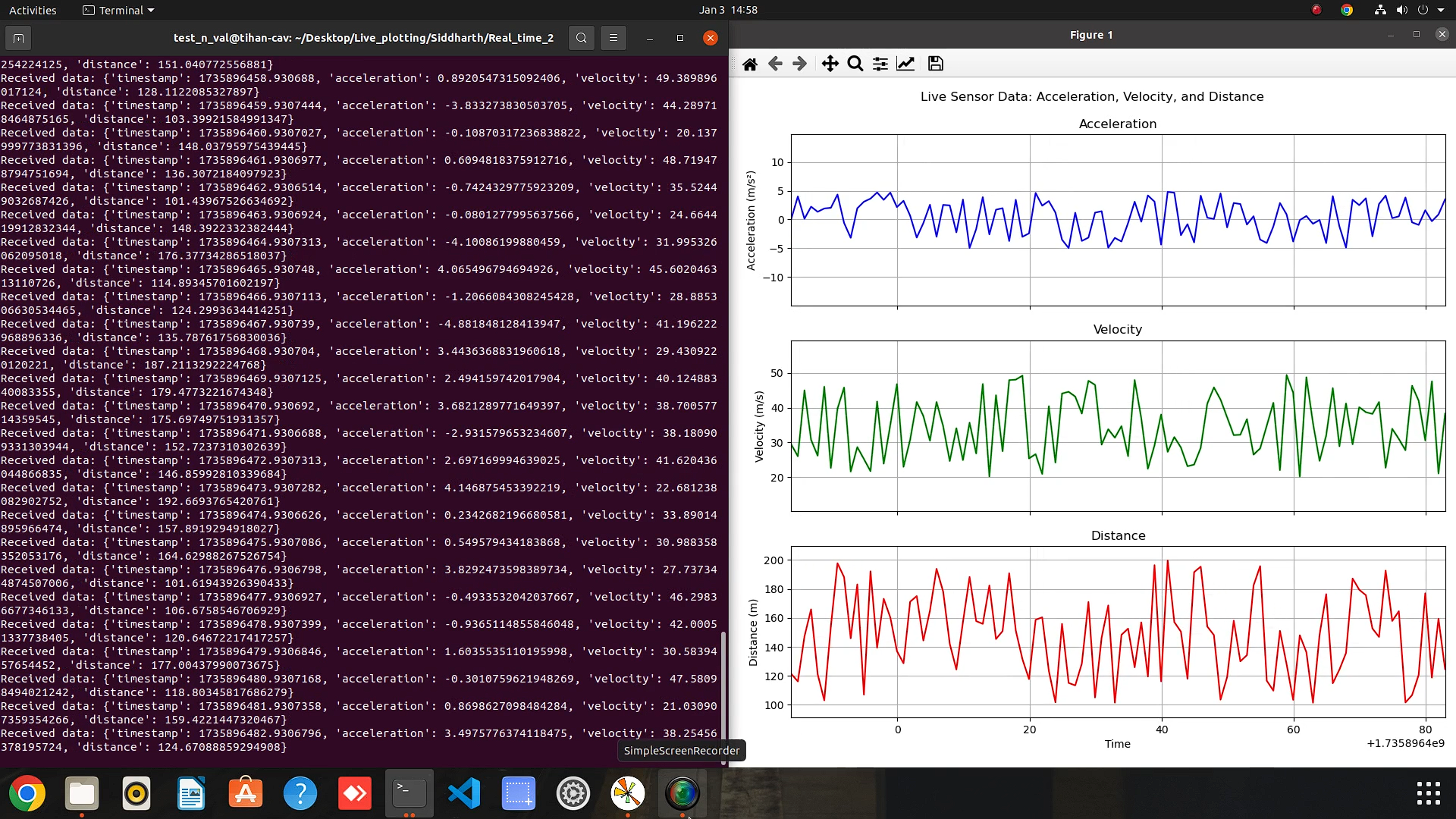
### 3. plotter.py

The `plotter.py` script acts as a client to the socket server, receiving data and visualizing it in real-time. It uses Matplotlib to plot data against time. The script dynamically updates the plots as new data arrives. The client side has no requirement of ROS or Ubuntu. It just should be able to run the required libraries like matplotlib and animations.  
Key Components:  
- Socket client setup for receiving JSON data.  
- Data parsing and dynamic plot updates.  
- Matplotlib animation for real-time visualization .

Required libraries to install before running it on client side.



The client side should look like this :



## Interactions and Workflow

1. The `generator.py` script generates and publishes sensor data to the ROS topic.  
2. The `server.py` script subscribes to the ROS topic, receives data, and forwards it to any connected clients via a socket.  
3. The `plotter.py` script connects to the socket server, receives the forwarded data, and visualizes it dynamically.

## Execution Instructions

1. Start the ROS Master using `roscore`.  
2. Run `generator.py` to start publishing sensor data.  
3. Start `server.py` to enable data forwarding via a socket server.  
4. Execute `plotter.py` to visualize the data in real-time.