Code Structure Overview Report

The purpose of this report is to provide an in-depth explanation of the code developed for controlling RVR robots in a swarm robotics scenario.

Important Note: the code folder "**RVR_Vicon_Swarm_Controller**" must be in the "**rvr_scripts**" folder.

The code consists of several components, each with specific roles and functionalities. The goal is to facilitate an understanding of how these components work together to coordinate the actions of RVR robots in a multi-robot system. Here's a structured breakdown of these components:

[1] Communication_Handler.py:

- **Function**:
  - This class is responsible for managing communication between devices over a network.
  - Serves as a foundational component for creating networked applications involving message exchange and robot monitoring.
- **Features**:
  - Facilitates the exchange of messages between devices.
  - Monitors the availability of robots.
  - Gracefully handles termination signals.

[2] rvr_vicon_swarm_controller.py:

- **Function**:
  - This script controls the movements of RVR robots in a swarm robotics scenario.
- **Core Component:**
  - It incorporates the **Agent** Class, which plays a pivotal role in controlling the RVR robots.
  - The Agent Class serves as the core component for coordinating the actions of RVR robots in a multi-robot system.
  - **Agent Class Features:**
    - Utilizes the Boid algorithm to calculate linear and angular velocities.
    - Establishes communication with other robots using the Communication_Handler class.
    - Receives position updates from VICON.

[3] Folder 'assets':

- **Purpose**:
  - This folder houses scripts and resources used by the rvr_vicon_swarm_controller script to enhance its functionalities. Here's a structured of these scripts:

[3.1] boid.py

- **Role:**
  - The Boid class is a fundamental component for simulating the behavior of individual agents in a swarm.
- **Key Attributes:**
  - Represents an individual agent in a swarm robotics scenario.
  - Follows specific rules, including alignment, cohesion, separation, and wall avoidance, to calculate linear and angular velocities.
  - Manages the computation of forces and velocities for each boid, including interactions with neighbors.
  - Maintains essential properties, such as position and heading angle.

Code Structure Overview Report

[3.1] Boids_Rules.py

- **Function:**
  - This script contains essential functions that drive the movement of individual agents (boids) within the swarm.
- **Key Behaviors:**
  - Implements rules of alignment, cohesion, separation, and wall avoidance to determine agents' velocities and positions within the swarm.

[3.2] Boids_Rules.py

- **Purpose:**
  - Handles the initialization and configuration of various parameters and settings for the simulation environment.
- **Features:**
  - Loads configuration data from a JSON file.
  - Sets up constants and variables.
  - Prepares the simulation environment.

[3.3] Helper_Functions.py

- **Purpose:**
  - Defines various utility functions with diverse purposes.
- **Tasks Covered:**
  - These functions perform tasks such as normalizing angles,
  - and normalizing velocities,
  - and calculating distances between two points.

[3.4] inputs_data.json

- **Contents:**
  - This JSON file contains essential data, including starting values for weight and ranges, and other important variables used in the contact.py file.