Örnek Takım Kodu Dokümantasyonu

Klasör Yapısı

Ana dizin altında `base` ve `test` adında iki ayrı dizin bulunmaktadır. `base` dizininde takımların kullanacakları iki basit python modülü bulunmaktadır; `base_uav.py` ve `rabbit_communication.py`. `base_uav.py` modülü `rabbit_communication.py` modülünü de kullanarak simülasyonla haberleşmeyi sağlamaktadır. `test` klasöründe İHA'ları ve simulasyon ortamını kolayca test etmeyi sağlayan iki python modülü bulunmaktadır.

RabbitMQ Haberleşmesi

Yeni versiyonla birlikte artık tüm haberleşme sağlanan `base/rabbit_communication.py` tarafından sağlanmaktadır. Bu haberleşme modülünün üç ayrı işlevi bulunmaktadır; senaryo parametrelerinin alınması, iha komutlarının simülasyona gönderilmesi ve iha'ya ait pozisyon, sensor ve link bilgilerinin simülasyondan alınması.

Takım Kodlarının Organizasyonu

Her takım burada sağlanan `main.py` modülünü değiştirip kendi modüllerini oluşturacaklardır. Dosyanın isminin aynı kalması ve parametre olarak da 0 dan maksimum İHA sayısına kadar bir sayıyı İHA id'si olarak komut satırı parametresi olarak alması gerekmektedir. Yarışma esnasında sizin İHA kodlarınızı sağlayacağınız bu `main.py` modülüyle çalıştıracağımız için burada belirtilen yönergelere uyulması oldukça önemlidir.

Takımlar bir tane yeni sınıf oluşturup bu sınıfın `base/uav_base.py` modülünden türemesini sağlayacaklardır. Mevcut kodda bunun basit bir örneği bulunmaktadır:

```
from base.base_uav import BaseUAV
class SampleUAV(BaseUAV):
    def initialize(self):
        pass

    def act(self):
        pass
```

Kendi kodunuzda `SampleUAV` ismi yerine kendi sınıfınızın ismini vermeniz gerekiyor. **Önemli nokta yeni oluşturduğunuz bu sınıfın `BaseUAV` sınıfının `initialize` ve `act` sınıflarını ezmesi gerekmektedir**

`initialize` metodu simülasyon başlamadan yapmak istediğiniz işlemleri yapabilmenizi sağlayacaktır. Simulasyon parametreleri ayrıca `self.params` değişkeninde hazır olacaktır.

`act` metodu ise simülasyondan veri aldığınızda çağrılacaktır. Bu metodun çağrılma sıklığı ve hızı tamamen simülasyonun hızına bağlıdır. Bunları kontrol etmek zorunda değilsiniz. Bu metodun en sonunda tüm karar verme işlerinizi bitirdiğinizde bir hareket komutu üretip bunu `UAVBase` sınıfına ait `send_move_cmd` metoduyla göndereceksiniz.

`initialize` ve `act` metodunda simülasyon alınan verileri kullanmanız gerekmektedir. Simülasyonun senaryo parametreleri `self.params` değişkeninde erişilebilir. Simülasyondan alınan İHA'nın pozu, sensor ve link verileri `self.uav_msg` değişkeninde erişilebilir. `self.uav_msg` değişkeni her adımda yenilenip en son mesajın bu değişkende oluduğunu unutmamak gerekiyor. Eğer bir önceki adımdan bir veri kullanılacaksa bu veri başka bir değişkende saklanması gerekmektedir.

SampleUAV

`SampleUAV` sınıfını inceleyelim. Bu sınıf rastgele gidilecek hedefler oluşturup İHA'nın o noktaya ulaşmasını sağlamaktadır. İHA belirlenen noktaya ulaşınca yeni bir rastgele hedef oluşturulmaktadır.

Sınıf içinde kullanılacak değişkenlerin ilk değerlerini `initialize` metodunda vermek işimizi kolaylaştırabilir. Burada gidilecek hedef noktalarını `self.target_position` değişkeninde saklıyoruz. `initialize` metodunu sınıfınızın yapıcı (constructor) metodu gibi düşünebilirsiniz. Ayrıca kendi geliştirdiğiniz metodda yapıcı (constructor) metod tanımlamamanız gerekmektedir. Yoksa `BaseUAV` sınıfının yapıcı metodunu ezerseniz ve bu da iletişim modüllerini **çalıştırmamanıza** sebep olur.

`act` metodunda öncelikle kolaylık olması açısından `self.uav_msg` değişkenindeki verilerden İHA'nın pozunu `self.pose` değişkenine liste olarak ekliyoruz. Daha sonra İHA'nın belirlenen hedefe ulaşıp ulaşmadığını kontrol ediyoruz. Eğer hedefe ulaşılmışsa yeni bir hedef seçiyoruz. En son adımda İHA'nın belirlenen hedef noktasına uçmasını sağlayacak hareket komutları oluşturulup gönderiliyor. Bu sınıfta kodlanan belirli bir noktaya gitme (`move_to_target`) veya belirlenen noktaya ulaşılıp ulaşılmadığını kontrol etme (`reach_to_target`) gibi kodları ihtiyacınıza göre yeniden kodlayabilir takım kodlarınızda kullanabilirsiniz.

Test

Testlerimizi kolaylaştırmak için klavye kontrol modülü (`keyboard_teleop.py`) ve mesaj dinleyici modül (`listen_uav_msg.py`) geliştirdik. Bu modüller kontrol edilecek İHA'nın idsini komut satırı argümanı olarak almaktadırlar. Bu programlar "0" id'li ayrı komut ekranlarında şu şekilde çalıştırılmaktadır;

```
python keyboard_teleop.py 0
python listen_uav_msg.py 0
```

Yarışma Kodlarının Teslimi ve Çalıştırılması

Her takım içinde kendi kodları olan bir sıkıştırılmış docker imajı teslim edecek. Alınan bu docker şu şekilde çalıştırılacak: - docker imajı yüklenmesi

`docker load -i sample_team.tar`

- docker imajı çalıştırılır

  `docker run -it --name sample_team_v0 --net=host sample_team`

- Her bir İHA kontrolcüsünü çalıştırmak için yeni interactive bash çalıştırılacak.

- Id'si 0 olan İHA şu şekilde çalıştırılacak.

  ```
  docker exec -it sample_team_v0 bash
  python main.py 0
  ```

- Diğer İHA'lar da aynı şekilde çalıştırılacaktır. Mesela Id'si 1 olan İHA da şu şekilde

  ```
  docker exec -it sample_team_v0 bash
  python main.py 1
  ```

  Mevcut kodlarımızdan docker imajı oluşturma hakkında fikriniz yoksa `Dockerfile` ve `create_docker_image.sh` dosyalarını inceleyebilirsiniz.