Описание задачи

Есть три сущности: агент, сервер и клиент.

- Агент - поставщик данных, внешнее устройство, которое собирает различные показания с датчиков и сенсоров, и в момент получения новых данных отправляет их на сервер в виде JSON пакета через WebSocket. Формат сообщений для агента за пределами этой задачи. К одному агенту может быть подключено несколько датчиков, каждый со своим уникальным идентификатором.
- Сервер - агрегатор данных, внешнее устройство. Собирает данные со всех подключенных агентов и посылает эти данные, а также информацию об агентах клиентам по запросу или же по зарегистрированному событию. API взаимодействия клиента и сервера описан ниже.
- Клиент - потребитель данных, инструмент для получения информации с сервера по агентам и данных от них. Подключается как модуль / библиотека в пользовательской программе.

API Сервер <-> Клиент

Общение между клиентом и сервером происходит через WebSocket. Пакет сообщения в формате JSON следующего вида:

```
{
    «event»: «new_event»,
    «data»: {
        «field_1»: «field_data»,
        - - - - - - - - - - - - - - -
        «field_n»: [data_1, data_2]
    }
}, где
```

- «event» - название события / запроса в строковом формате
- «data» - данные запроса, может быть пустой строкой («») в случае их отсутствия

Сообщения Клиент -> Сервер

- Подключение нового клиента

```
{
    «event»: «client_connect»,
    «data»: «»
}
```

  Клиент должен отправить его серверу после подключения по WebSocket

- Отключение клиента

```
{
    «event»: «client_disconnect»,
    «data»: «»
}
```

  Клиент должен отправить его серверу перед отключением от WebSocket

- Запрос информации о подключении сенсора

```
{
    «event»: «sensor_connection_status»,
    «data»: {
        «sensor_id»: «CE7238J»
    }
}
```

Может быть отправлен клиентом, когда понадобится информация о подключении конкретного датчика

- Подписка на сенсор
```
{
    «event»: «subscribe_sensor»,
    «data»: {
        «sensor_id»: «CE7238J»
    }
}
```

Клиент сообщает серверу, что хочет подписаться на данные с конкретного датчика. Если клиент подписан на сенсор, то сервер будет сразу же присылать ему новую информацию от этого датчика.

- Отписка от сенсора
```
{
    «event»: «unsubscribe_sensor»,
    «data»: {
        «sensor_id»: «CE7238J»
    }
}
```

Клиент сообщает серверу, что хочет отписаться от датчика. После этого сервер перестает посылать показания датчика клиенту.

Сообщения Сервер -> Клиент

- Информация о подключении сенсора
```
{
    «event»: «sensor_connection_status»,
    «data»: {
        «sensor_id»: «CE7238J»
        «connected»: false
    }
}
```

Присылает в ответ на « Запрос информации о подключении сенсора» от клиента

- Новые показания датчика, на который была выполнена подписка
```
{
    «event»: «new_sensor_data»,
    «data»: {
        «sensor_id»: «CE7238J»
        «sensor_readings»: «SOME_DATA» // в строковом формате, для клиента содержание по сути не важно
    }
}
```

Присылает клиенту, если агент прислал новые показания датчика, и клиент подписан на этот датчик

Постановка задачи

Вам необходимо написать инструмент клиента, который может быть подключен (import / include) к программе разработчика и реализует следующий необходимый функционал:

- Инициализация с аргументами для подключения к WebSocket серверу (host (str) и port (str || int))

  // client = Client(«localhost», 5192)

- Подключение к WebSocket серверу по заданным аргументам host и port, и отправка сообщения серверу о новом соединении (сценарий «Подключение нового клиента» из API)

  // client.connect()

- Отключение от WebSocket сервера и отправка сообщения серверу об отключении (сценарий « Отключение клиента» из API)

  // client.disconnect()

- Запрос статуса подключения по конкретному сенсору, возвращает булевое значение

  // client.sensorConnected(«CE7238J»), размер идентификатора сканера от 1 до 40 символов

- Подписка на сенсор

  // client.subscribe(«CE7238J»)

- Отписка от сенсора

  // client.unsubscribe(«CE7238J»)

- Регистрация коллбека для данных от конкретного сенсора. Когда от сервера придут новые показания по этому сенсору, клиент должен вызвать зарегистрированный коллбек и передать ему показания в аргументы в виде (sensor_id, sensor_readings). Также возможно, что у коллбека будут дополнительные аргументы, которые задает непосредственно разработчик.

  В случае, если до этого не была произведена подписка на сенсор, она должна быть выполнена тут.

  ```
  //  def myCallback(sensor_id, sensor_readings, my_arg1, my_arg2, my_arg3):
  //      …
  //  some_arg = AwesomeClass()
  //  client.registerCallback(«CE7238J», myCallback, callback_args = («arg1», 24, some_arg))
  ```

- Удаление коллбеков для конкретного сенсора

  // client.removeCallback(«CE7238J»)

- В случае нештатного отключения (проблемы со связью или с сервером), клиент должен автоматически подключиться при восстановлении связи

Вместе с исходниками инструмента должны предоставляться:
- Документ, в котором указаны необходимые зависимости
- Скрипт для сборки библиотеки / модуля (возможно в формате инструкция + setup.py / CmakeLists.txt / Makefile)