Инструкция по использованию Slam Sim

**Содержание**

[**Установка 3**](#_Toc106133384)

[**Запуск Slam Sim на Windows 6**](#_Toc106133385)

[**Запуск Slam Sim на Linux 8**](#_Toc106133386)

[**API запросы 9**](#_Toc106133387)

[**Работа со Slam Sim на Python 12**](#_Toc106133388)

[**Подключение к CoppeliaSim 12**](#_Toc106133389)

[**Загрузка сцены CoppeliaSim 14**](#_Toc106133390)

[**Загрузка модели робота в CoppeliaSim 14**](#_Toc106133391)

[**Запуск, приостановка, и остановка симуляции в CoppeliaSim 15**](#_Toc106133392)

[**Движение робота вперед, назад и повороты в CoppeliaSim 15**](#_Toc106133393)

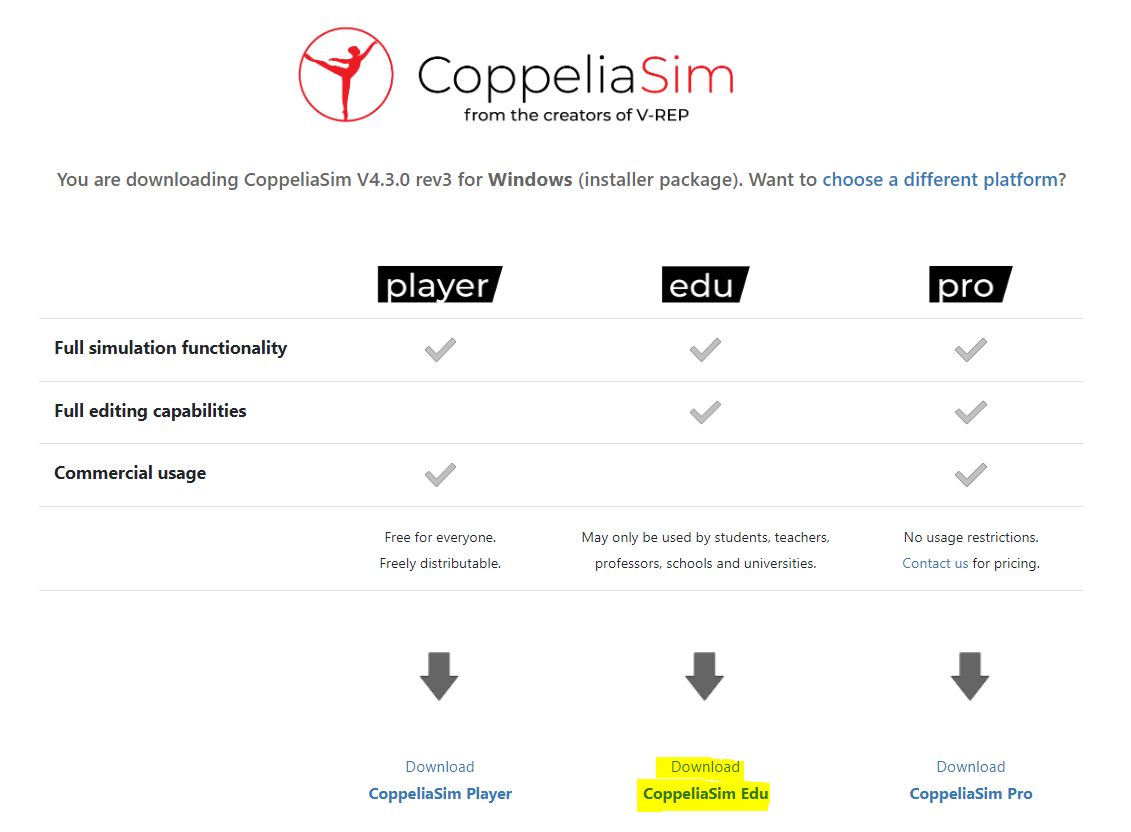
[**Получение координат робота на сцене в CoppeliaSim 17**](#_Toc106133394)

[**Опрос данных с лидара в CoppeliaSim 17**](#_Toc106133395)

# **Установка**

Slam Sim - пользовательский API, разработанный для упрощения использования V-Rep для разработки и оценки алгоритмов SLAM.

Прежде всего, установите последнюю версию V-Rep, доступную на официальном сайте Coppelia Robotics (http://www.coppeliarobotics.com /). Необходимо устанавливать версию EDU.



Важно! Если вдруг последняя версии CoppeliaSim не будет работать со Slam Sim попробуйте понизить версию CoppeliaSim. Slam Sim стабильно работает на версиях 4.2.0 и 4.1.0.

Примечание: Версии CoppeliaSim 4.2.0 и 4.1.0 сразу после установки будут выкидывать из приложения. В консоли будет писать Leaving… Чтобы пофиксить данный баг необходимо зайти в директорию, где установлен CoppeliaSim, далее открыть system/usrset.txt. В данном файле необходимо добавить 1 строчку в начале файла, в зависимости от версии.

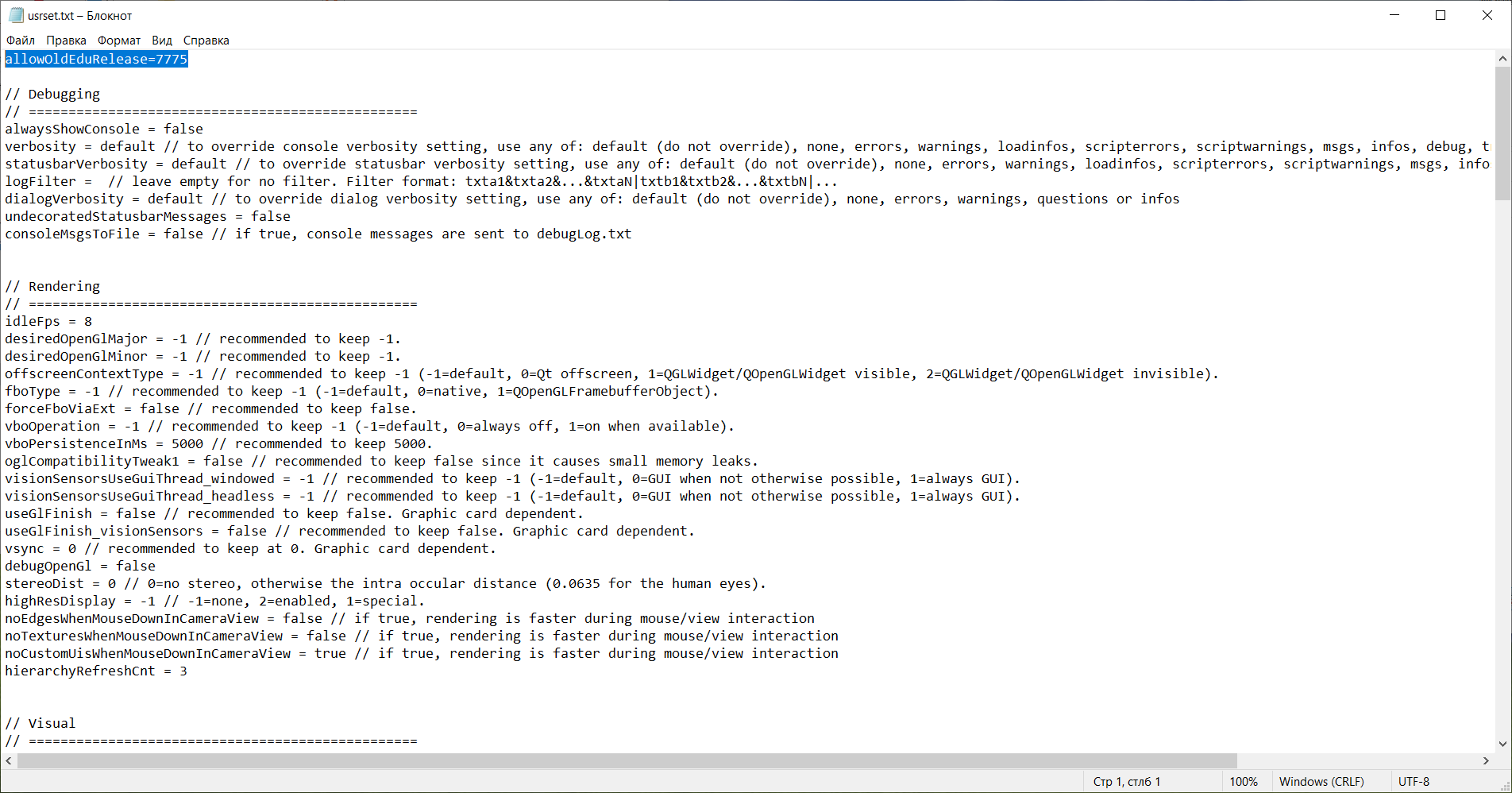
Для 4.2.0:

allowOldEduRelease=7775

Для 4.1.0:

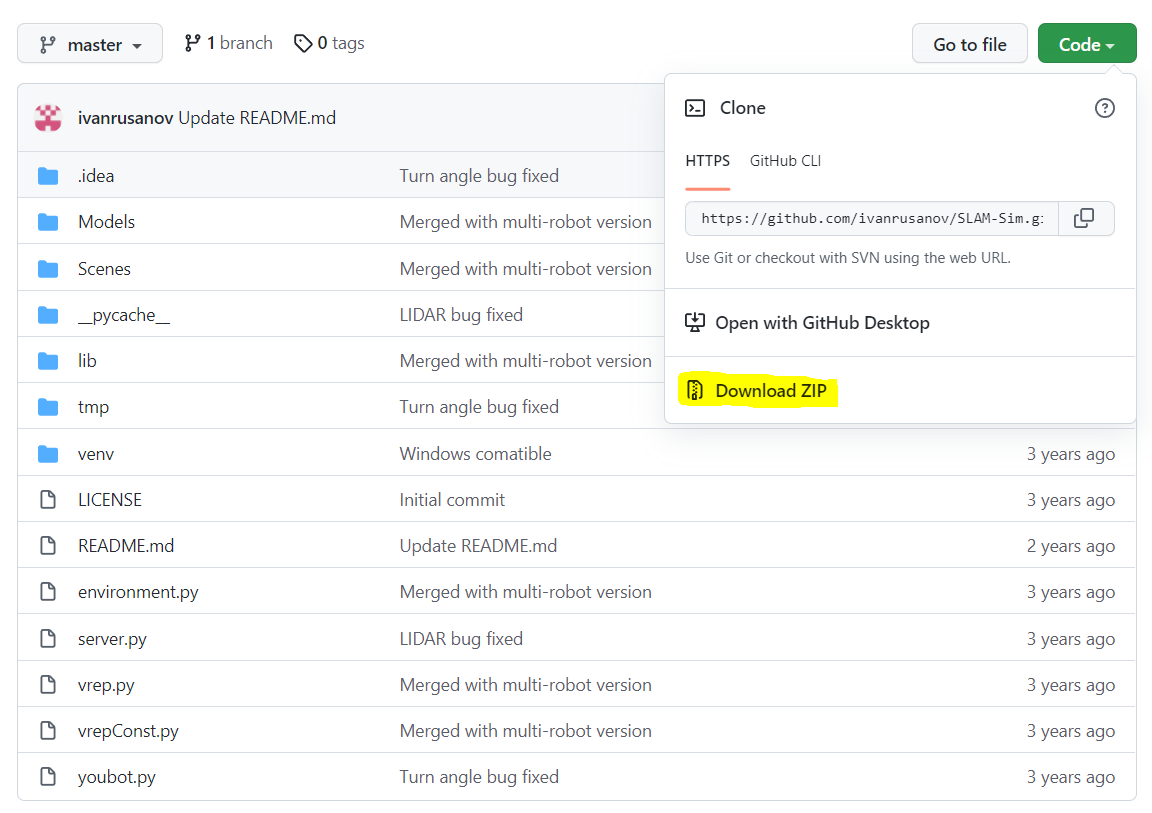
allowOldEduRelease=7501

Пример:

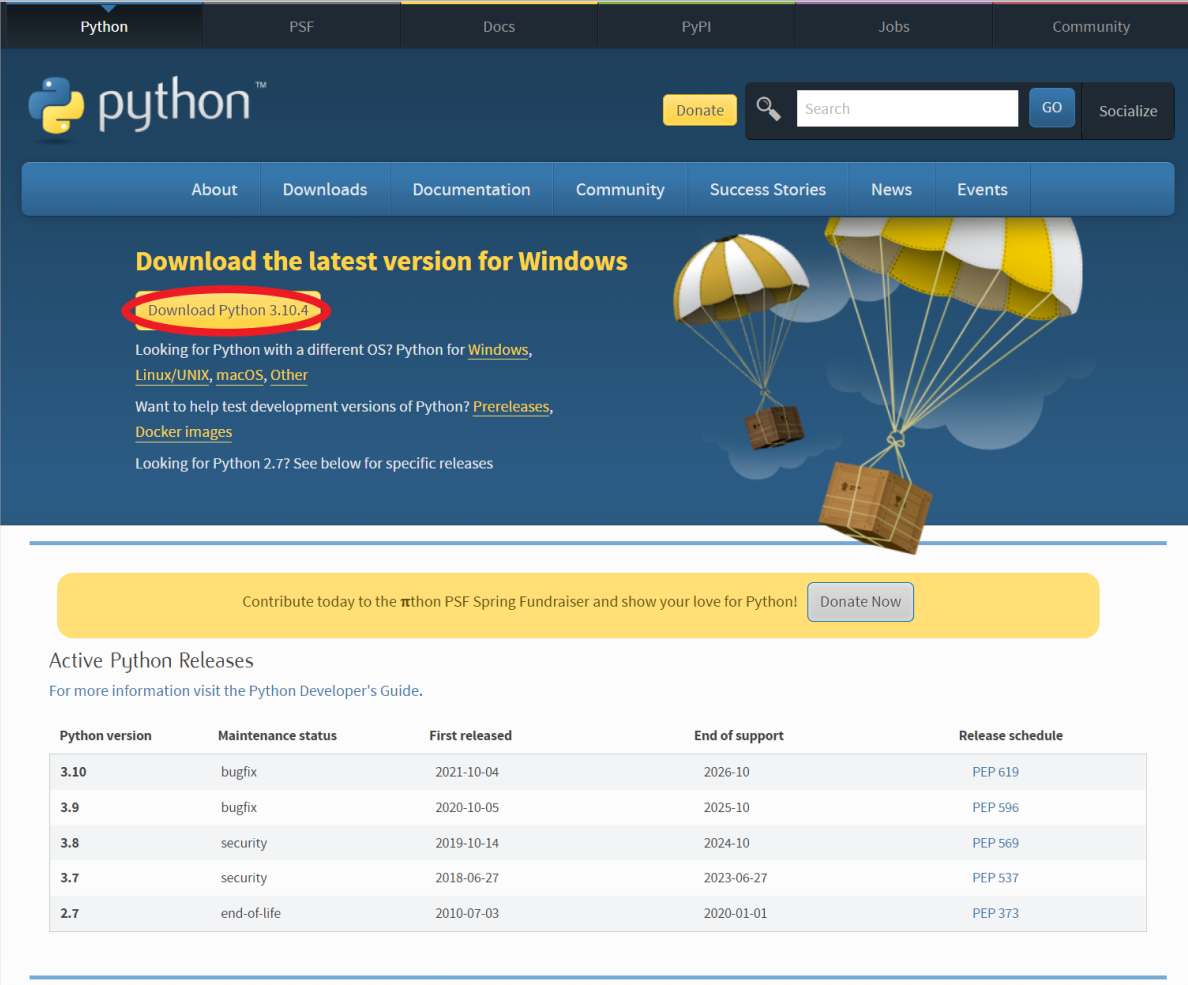


После изменения файл необходимо сохранить!

Затем загрузите файлы этого репозитория, нажав на кнопку "Download ZIP".



Убедитесь, что на вашем компьютере установлен Python 3 и pip. Если нет, установите его из https://www.python.org/downloads /.

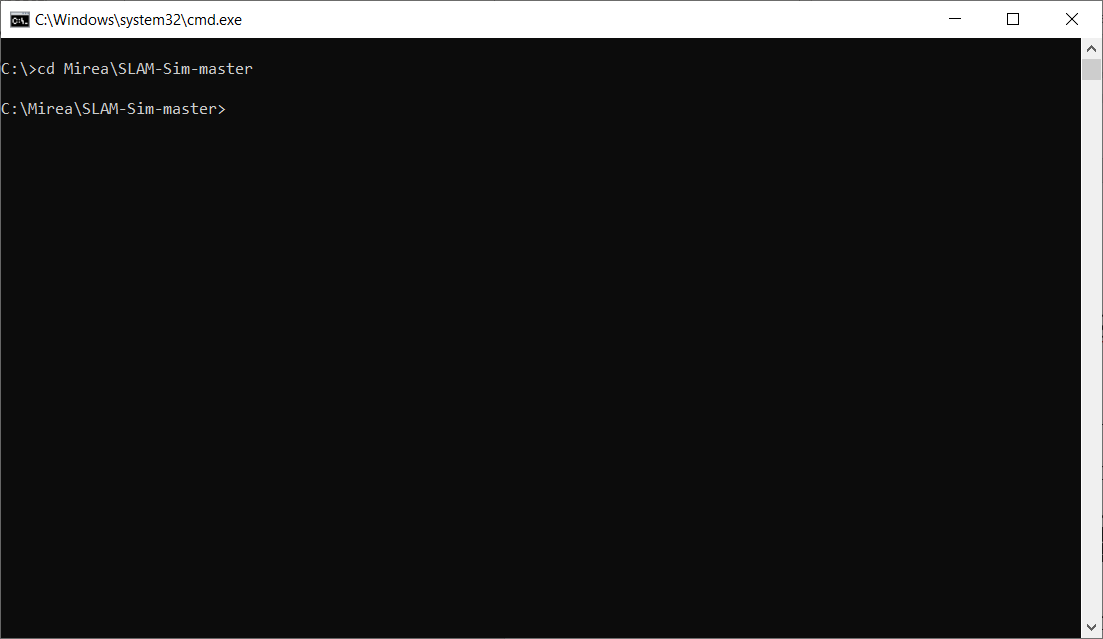


Если вы используете Ubuntu или Debian Linux, вы можете установить его из терминала с помощью sudo apt install python3 и sudo apt install python3-dev.

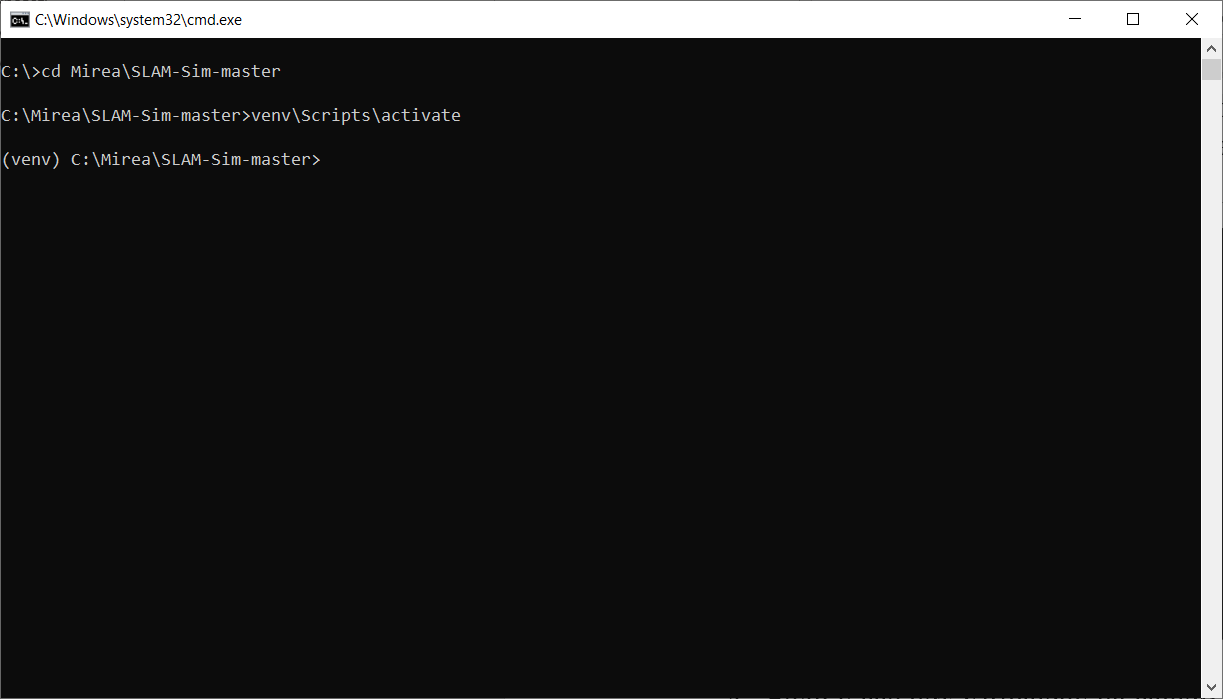
# **Запуск Slam Sim на Windows**

Для запуска Slam Sim выполните следующие действия:

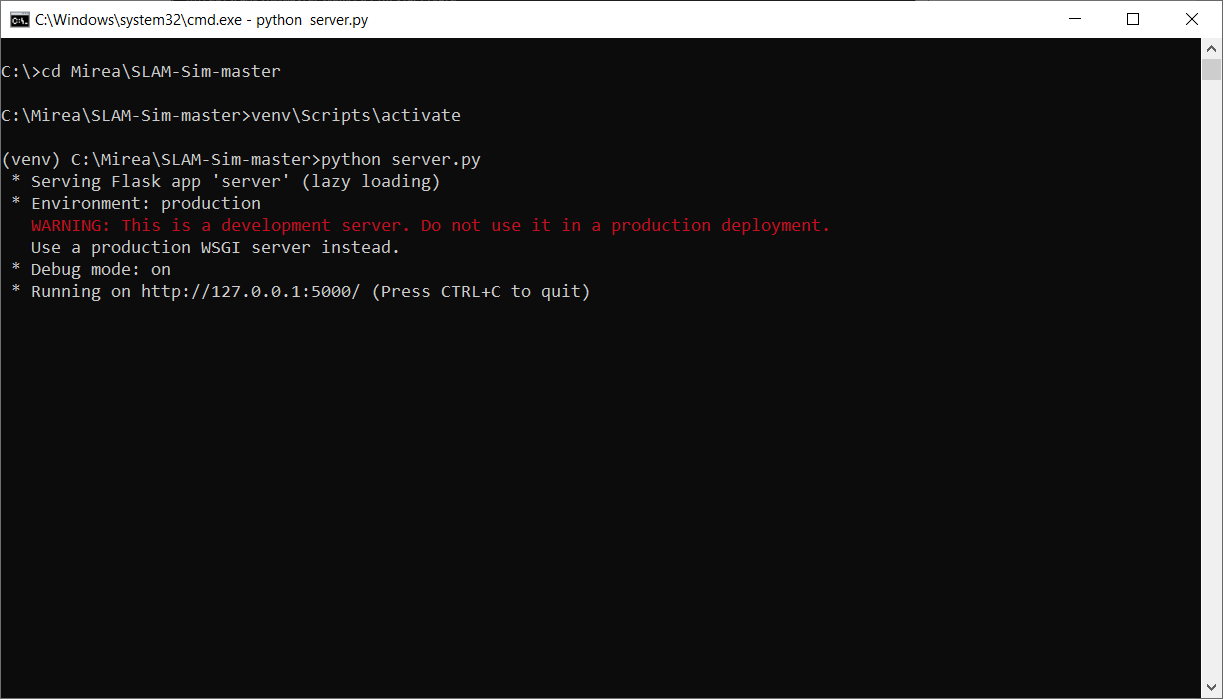
1. Откройте командную строку (нажмите Win+R и введите "cmd");
2. Чтобы установить необходимые пакеты python введите: pip install flask numpy requests
3. Перейдите в папку SLAM-Sim (вы можете использовать команду cd. Для получения более подробной информации перейдите по ссылке https://ss64.com/nt/cd.html );



1. Если у вас нет Virtualenv на вашем компьютере, вы можете легко установить его с помощью pip install virtualenv;
2. Введите venv\Scripts\activate. Если вы все сделали правильно, вы увидите (venv) в начале текущей строки;



1. Введите python server.py . Вы можете использовать аргументы $ python3 server.py -p=19998 -i=127.0.0.1, чтобы изменить порт или IP-адрес удаленного сервера API V-Rep.



1. После этого вы получите сообщение от сервера, заканчивающееся словами: \* Running on http://127.0.0.1:5000 /.

# **Запуск Slam Sim на Linux**

1. Откройте терминал (Ctrl+Alt+T);
2. Перейдите в папку SLAM-Sim командой cd;
3. Если у вас нет Virtualenv на вашем компьютере, вы можете легко установить его с помощью pip install virtualenv;
4. Введите команду source mypython/bin/activate. Если вы все сделали правильно, вы увидите (venv) в начале текущей строки.
5. Введите python server.py . Вы можете использовать аргументы $ python3 server.py -p=19998 -i=127.0.0.1, чтобы изменить порт или IP-адрес удаленного сервера API V-Rep.

# **API запросы**

|  |  |  |
| --- | --- | --- |
| Вид запроса | Описание | Пример |
| /start-v-rep-server?ip=<IP Address>&port=<Port> | Подключение к V-REP. Аргументы: ip - ip- адрес, с которого будет запущен удаленный API-сервер V-REP. Port - на котором будет запущен удаленный API-сервер V-REP. Способ запроса: POST. | http://127.0.0.1:5000/start-v-rep-server?ip=127.0.0.1&port=19997 |
| /start-simulation | Начнет симуляцию V-REP. Способ запроса: POST. | http://127.0.0.1:5000/start-simulation |
| /pause-simulation | Поставит на паузу симуляцию V-REP. Способ запроса: POST. | http://127.0.0.1:5000/pause-simulation |
| /stop-simulation | Выключит симуляцию V-REP.  Способ запроса: POST. | http://127.0.0.1:5000/stop-simulation |
| /print-message?text=<Text> | Выведет сообщение в консоль V-REP. Способ запроса: POST. | http://127.0.0.1:5000/print-message?Some text message you want to be printed |
| /load-scene?file-name=<FileName> | Загрузит существующую сцену. Способ: POST. Не забудьте добавить расширение .ttt к имени файла. | http://127.0.0.1:5000/load-scene?scene=Simple.ttt |
| /add-robot?model-name=<Model name> | Добавит в сцену робота, указанного по имени файла модели. Способ: POST. | http://127.0.0.1:5000/add-robot?model=KUKA YouBot.ttm |
| /forward?robot-id=<Robot ID>&speed=<Speed> | Заставит UBot двигаться вперед с заданной скоростью. Способ: POST. Передаваемая скорость должна быть в пределах до 1. | http://127.0.0.1:5000/forward?robot-id=0&speed=0.5 |
| /backward?robot-id=<Robot ID>&speed=<Speed> | Заставит UBot двигаться назад с заданной скоростью. Способ: POST. Передаваемая скорость должна быть в пределах до 1. | http://127.0.0.1:5000/backward?robot-id=0&speed=0.5 |
| /stop?robot-id=<Robot ID> | Заставит UBot остановиться. Способ: POST. | http://127.0.0.1:5000/stop?robot-id=0 |
| /replace?robot-id=<Robot ID>&x=<X>&y=<Y>&z=<Z> | Переместите робота в положение, указанное координатами.  Способ: POST. | http://127.0.0.1:5000/replace?robot-id=0&x=-1&y=0&z=0.1 |
| /get-image/<Robot ID> | Получите изображение с камеры RGB, установленной на роботе. Возвращает изображение в формате GIF. Метод: GET. | http://127.0.0.1:5000/get-image/0 |
| /get-depth/<Robot ID> | Получите данные о глубине, измеренные с помощью KINECT, установленного на роботе. Возвращает массив 480x640. Метод: GET. | http://127.0.0.1:5000/get-depth/0 |
| /get-lidar/<Robot ID> | Получите данные, измеренные с помощью лидара, установленного на роботе. Возвращает одномерный массив. Метод: GET. | http://127.0.0.1:5000/get-lidar/0 |
| /get-sensor-params-list/<int:robot\_id> | Получить список доступных для изменения параметров датчиков. Метод: GET. | http://127.0.0.1:5000/get-sensor-params-list/0 |
| /get-sensor-params/<robot-id>/<sensor-name>/<parameter-name> | Получить значение параметра датчика по его имени. Возможные значения "имя датчика": kinect\_rgb, kinect\_depth, fastHokuyo\_sensor 1. Метод: GET. | http://127.0.0.1:5000/get-sensor-params/1/kinect\_rgb/perspective\_angle |
| /set-sensor-params?robot-id=1&sensor-name=<Value>&parameter-name=<Value>&parameter-value=<Value> | Измените параметр датчика. Возможные значения "имя датчика": kinect\_rgb, kinect\_depth, fastHokuyo\_sensor 1. Метод: POST. | http://127.0.0.1:5000/set-sensor-params?robot-id=1&sensor-name=kinect\_rgb&parameter-name=perspective\_angle&parameter-value=110 |
| /get-standard-deviation/<Robot ID> | Возвращает стандартное отклонение - сигма. Для получения дополнительной информации посетите сайт <https://en.wikipedia.org/wiki/Standard_deviation>. | http://127.0.0.1:5000/get-standard-deviation/0 |
| /set-standard-deviation | Задает стандартное отклонение для движений робота. Метод: POST. | http://127.0.0.1:5000/set-standard-deviation/0 |
| /move-distance??robot-id=<Value>&distance=<Value>&speed=<Value> | Перемещает робота на заданное расстояние с определенной скоростью. Метод: POST. | http://127.0.0.1:5000/move-distance?robot-id=0&distance=1.0&speed=1.0 |
| /turn-angle?robot-id=<Value>&angle=<Value>&speed=<Value> | Поворачивает робота по часовой стрелке speed>0) или против часовой стрелки (speed<0) на заданный угол с заданной скоростью. Speed может принимать значения от -1 до 1. Метод: POST. | http://127.0.0.1:5000/turn-angle?robot-id=0& angle=90&speed=1 |

# **Работа со Slam Sim на Python**

Для работы Slam Sim с помощью python необходимо установить пакет requests, для этого необходимо в консоли написать pip install requests.

Работа со Slam Sim связанна с отправкой API запросов через библиотеку requests. Для нормальной работы необходимо выполнить следующий алгоритм действий:

1. Запустить Slam Sim (для удобного запуска на Windows есть start\_server.exe, который лежит в папке Slam Sim)
2. Запустить СoppeliaSim
3. Послать API запрос для подключения к CoppeliaSim
4. Послать API запрос для загрузки сцены
5. Послать API запрос для загрузки модели робота
6. Послать API запрос для старта симуляции
7. Далее необходимо посылать API запросы для задачи движения робота и опроса датчиков.

Рассмотрим, как выполнять API запросы начиная с 3 пункта данного алгоритма.

## **Подключение к CoppeliaSim**

Запустите Slam Sim (для удобного запуска на Windows есть start\_server.exe, который лежит в папке Slam Sim), далее запустите СoppeliaSim.

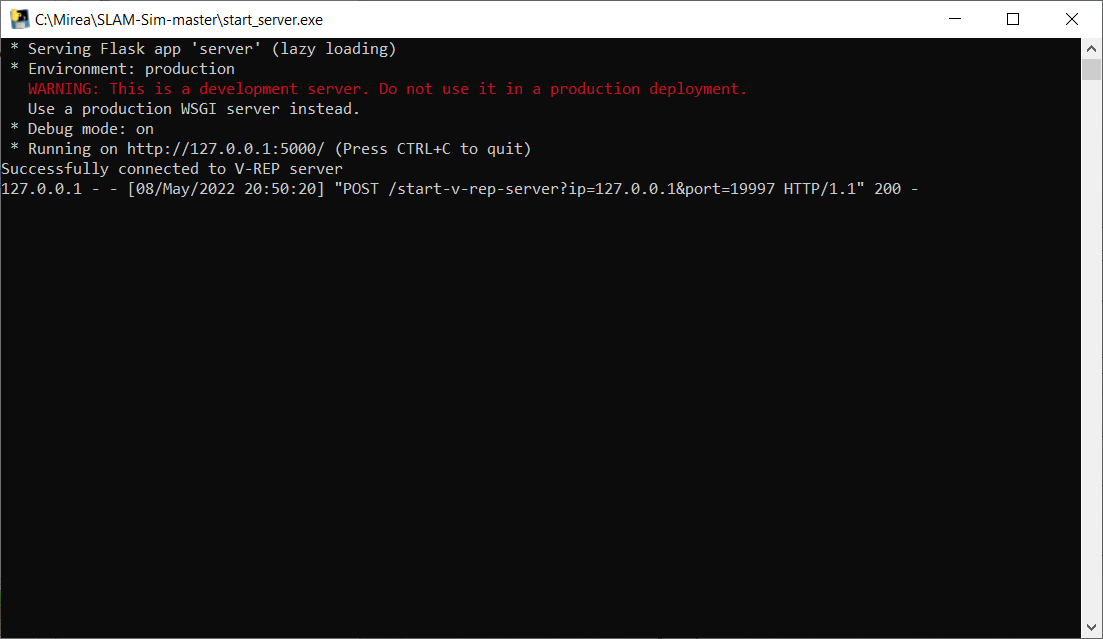
Алгоритм подключения к серверу после включение:

1) Создайте файл main.py в папке в Slam Sim

2) В данном файле напишите следующий код:

import requests  
  
def vrep\_connection():  
 connect = 'http://127.0.0.1:5000/start-v-rep-server?ip=127.0.0.1&port=19997'  
 requests.post(connect)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 vrep\_connection()

Если все правильно сделано, то Slam Sim напишет:



Последнее число 200, которое написал Slam Sim - это код ответа http запросов.

Код ответа (состояния) HTTP показывает, был ли успешно выполнен определённый HTTP запрос. Коды сгруппированы в 5 классов:

1)Информационные - 100

2)Успешные - 200

3)Перенаправления - 300

4)Клиентские ошибки - 400

5)Серверные ошибки - 500

В CoppeliaSim в консоли появиться надпись о подключении.

## **Загрузка сцены CoppeliaSim**

Загружаемые сцены хранятся в Slam Sim в папке Scenes.

Важно! При загрузке Slam Sim можно загрузить только 1 сцену. Для того чтобы загрузить другую необходимо перезапустить Slam Sim. За один сеанс нельзя загрузить несколько разных сцен.

Для того, чтобы загрузить сцену, необходимо добавить в код следующую функцию:

def load\_sceen():  
 load\_sceen = 'http://127.0.0.1:5000/load-scene?scene=Simple.ttt'  
 requests.post(load\_sceen)

В этой функции приведена загрузка сцены Simple.ttt для загрузки другой сцены необходимо записать название требуемой сцены.

Важно! Сцена должна быть сохранены в папке Scenes.

Далее эту функцию необходимо вызвать.

## **Загрузка модели робота в CoppeliaSim**

Загружаемые сцены хранятся в Slam Sim в папке Models.

Для того, чтобы загрузить робота, необходимо добавить в код следующую функцию:

def load\_robot():  
 add\_robot = 'http://127.0.0.1:5000/add-robot?model=KUKA YouBot.ttm'  
 requests.post(add\_robot)

В этой функции приведена загрузка робота KUKA YouBot.ttm. Для загрузки другого робота необходимо записать название требуемого робота.

Далее эту функцию необходимо вызвать.

## **Запуск, приостановка, и остановка симуляции в CoppeliaSim**

Для того, чтобы начать симуляцию, необходимо добавить в код следующую функцию:

def start\_sim():  
 start = 'http://127.0.0.1:5000/start-simulation'  
 requests.post(start)

Для того, чтобы приостановить симуляцию, необходимо добавить в код следующую функцию:

def pause\_sim():  
 pause = 'http://127.0.0.1:5000/pause-simulation’  
 requests.post(pause)

Для того, чтобы остановить симуляцию, необходимо добавить в код следующую функцию:

def stop\_sim():  
 stop = 'http://127.0.0.1:5000/stop-simulation’  
 requests.post(stop)

После запуска симуляции можно переходить к управлению движением робота.

## **Движение робота вперед, назад и повороты в CoppeliaSim**

Для того, чтобы заставить робота двигаться вперед, необходимо добавить следующую функцию:

def move\_forward(speed):  
 if speed <= 1:  
 move = 'http://127.0.0.1:5000/backward?robot-id=0&speed=' + str(speed)  
 requests.post(move)

где в качестве аргумента у этой функции выступает скорость, с которой поедет робот.

Для того, чтобы заставить робота двигаться назад, необходимо добавить следующую функцию:

def move\_back(speed):  
 if speed <= 1:  
 move = 'http://127.0.0.1:5000/forward?robot-id=0&speed=' + str(speed)  
 requests.post(move)

где в качестве аргумента у этой функции выступает скорость, с которой поедет робот.

Для того, чтобы заставить робота повернуть по часовой стрелке на определённый угол, необходимо добавить следующую функцию:

def turn\_right(angel):  
 turn = 'http://127.0.0.1:5000/turn-angle?robot-id=0&angle=' + str(angel) + '&speed=0.99'  
 requests.post(turn)

где в качестве аргумента у этой функции выступает угол поворота.

Для того, чтобы заставить робота повернуть против часовой стрелки на определённый угол, необходимо добавить следующую функцию:

def turn\_left(angel):  
 turn = 'http://127.0.0.1:5000/turn-angle?robot-id=0&angle=' + str(angel) + '&speed=-0.99'  
 requests.post(turn)

где в качестве аргумента у этой функции выступает угол поворота.

Для того, чтобы заставить робота проехать определенное расстояние, необходимо добавить следующую функцию:

def move\_distance(distance, speed):  
 distances = 'http://127.0.0.1:5000/move-distance?robot-id=0&distance=' + str(distance) + '&speed=' + str(-speed)  
 requests.post(distances)

где в качестве аргумента у этой функции выступает расстояние, которое необходимо проехать, и скорость.

## **Получение координат робота на сцене в CoppeliaSim**

Для того, чтобы получить координаты робота на сцене, необходимо добавить следующую функцию:

import re

def get\_position():  
 position = 'http://127.0.0.1:5000/get\_position/0'  
 position = requests.get(position)  
 pose = position.content.decode()  
 pose = [float(s) for s in re.findall(r'-?\d+\.?\d\*', pose)]  
 return pose

## **Опрос данных с лидара в CoppeliaSim**

Для того, чтобы получить данные с лидара, необходимо добавить следующую функцию:

import re

def get\_lidar():  
 lidar = 'http://127.0.0.1:5000/get-lidar/0'  
 lidar\_data = requests.get(lidar)  
 ranges = lidar\_data.content.decode()  
 ranges = [float(s) for s in re.findall(r'-?\d+\.?\d\*', ranges)]  
 return ranges

Данная функция возвращает float массив из 684 элементов. Каждый элемент массива – это длина луча в секторе 280 градусов. На один градус приходится 684/280 лучей.

Лидар лучше опрашивать в отдельном потоке и перед этим проверять пришел ли ответ от предыдущей команды.