**Управление движением робототехнического манипулятора с заданием скорости**

[**Github**](https://github.com/danissomo/mipt)

1. Установка симулятора URSim

Установил на Linux Mint 20, cкачал с официального сайта  [UR SIM FOR LINUX 5.11.1](https://www.universal-robots.com/download/software-e-series/simulator-linux/offline-simulator-e-series-ur-sim-for-linux-5111/)

Для установки необходима java 8, если версия выше, то выводится сообщение “Installed java is too old, exiting”.

sudo apt install openjdk-8-jdk

sudo update-alternatives --config java

Распаковать архив в домашнюю директорию, и отредактировать скрипт install.sh: в строке 71 заменить libcurl3 на libcurl4 (либо libcurl4:i386).

1. Установка библиотеки UR\_RTDE и настройка URSim для управления через UR\_RTDE

Установка Boost на Ubuntu.

sudo apt-get install libboost-all-dev

Сборка UR\_RTDE

git clone https://gitlab.com/sdurobotics/ur\_rtde.git

cd ur\_rtde

git submodule update --init --recursive

mkdir build

cd build

cmake ..

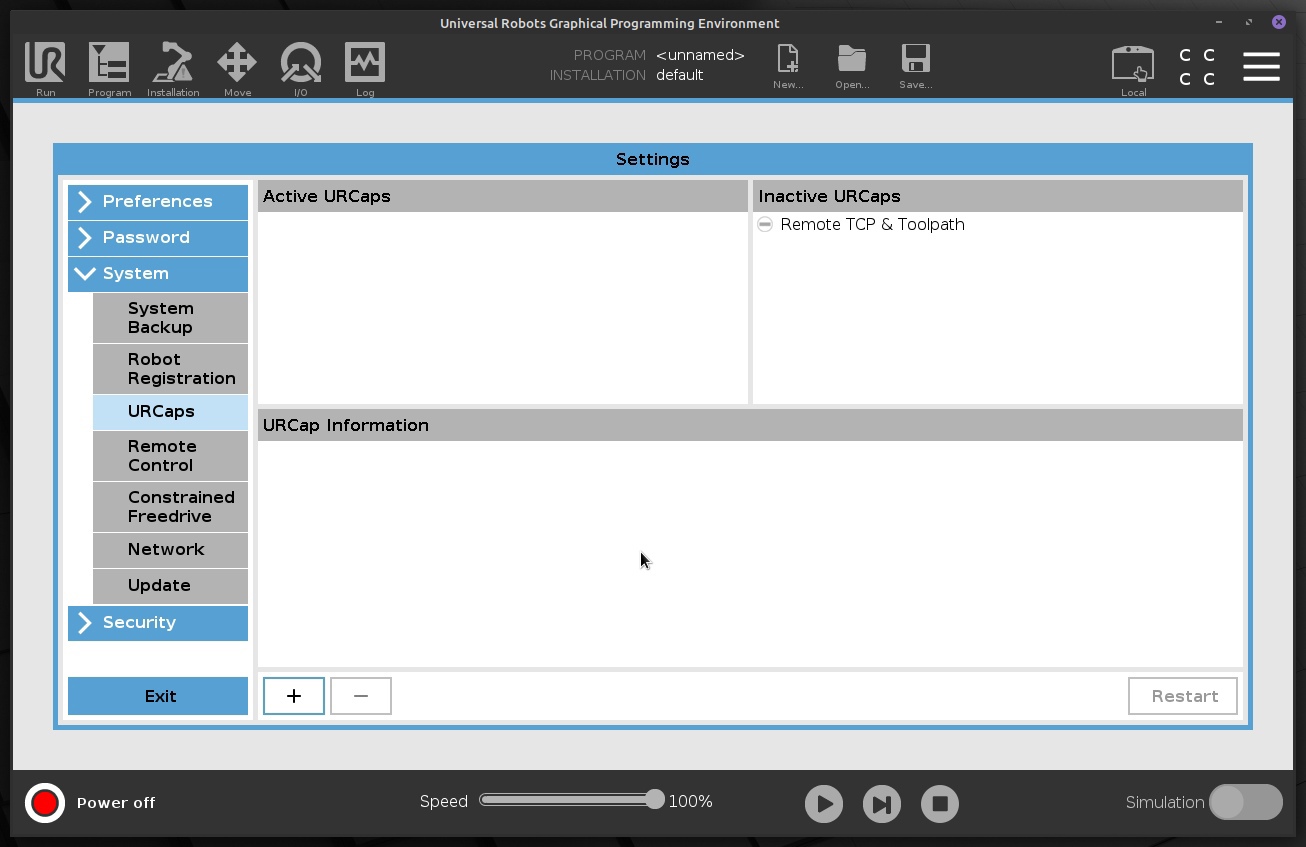
make

sudo make install

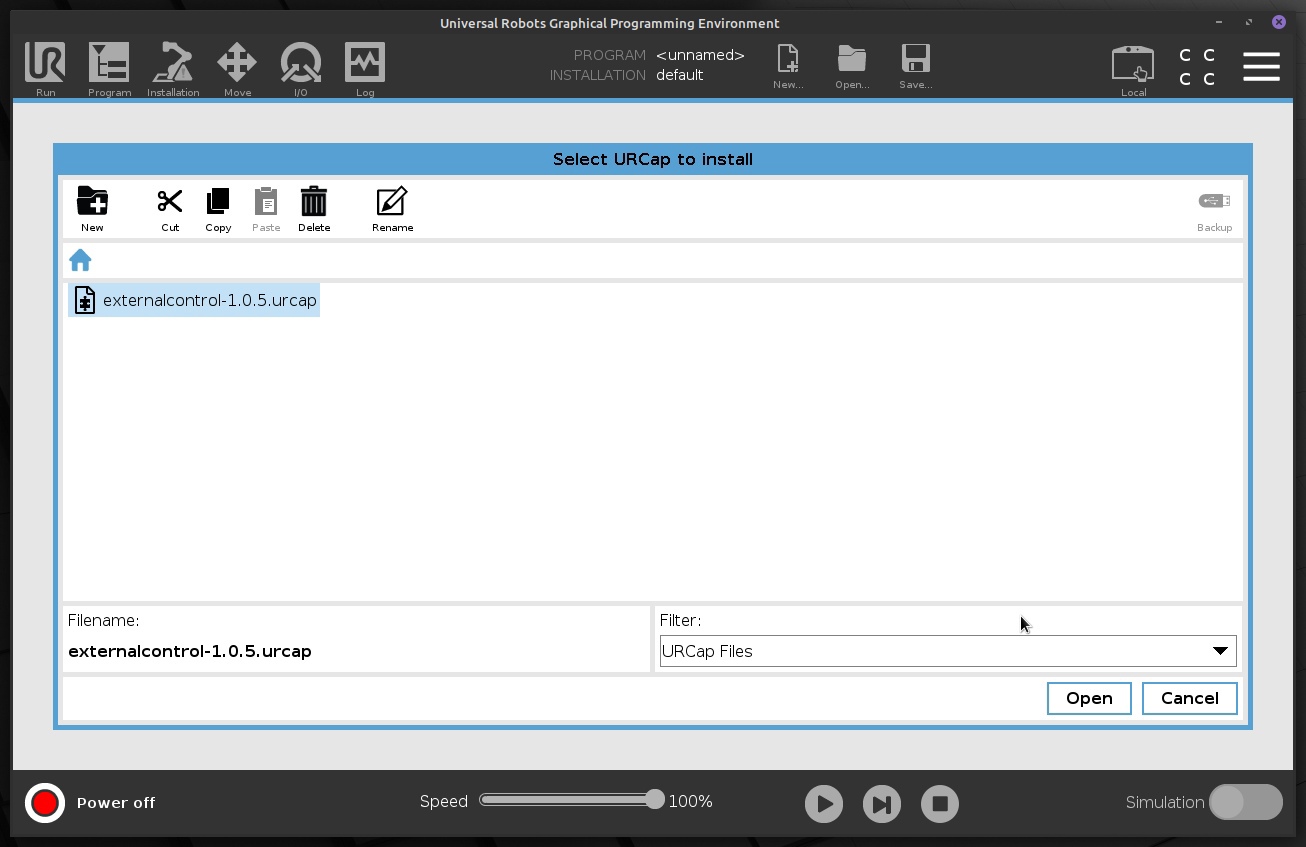
Настройка URSim

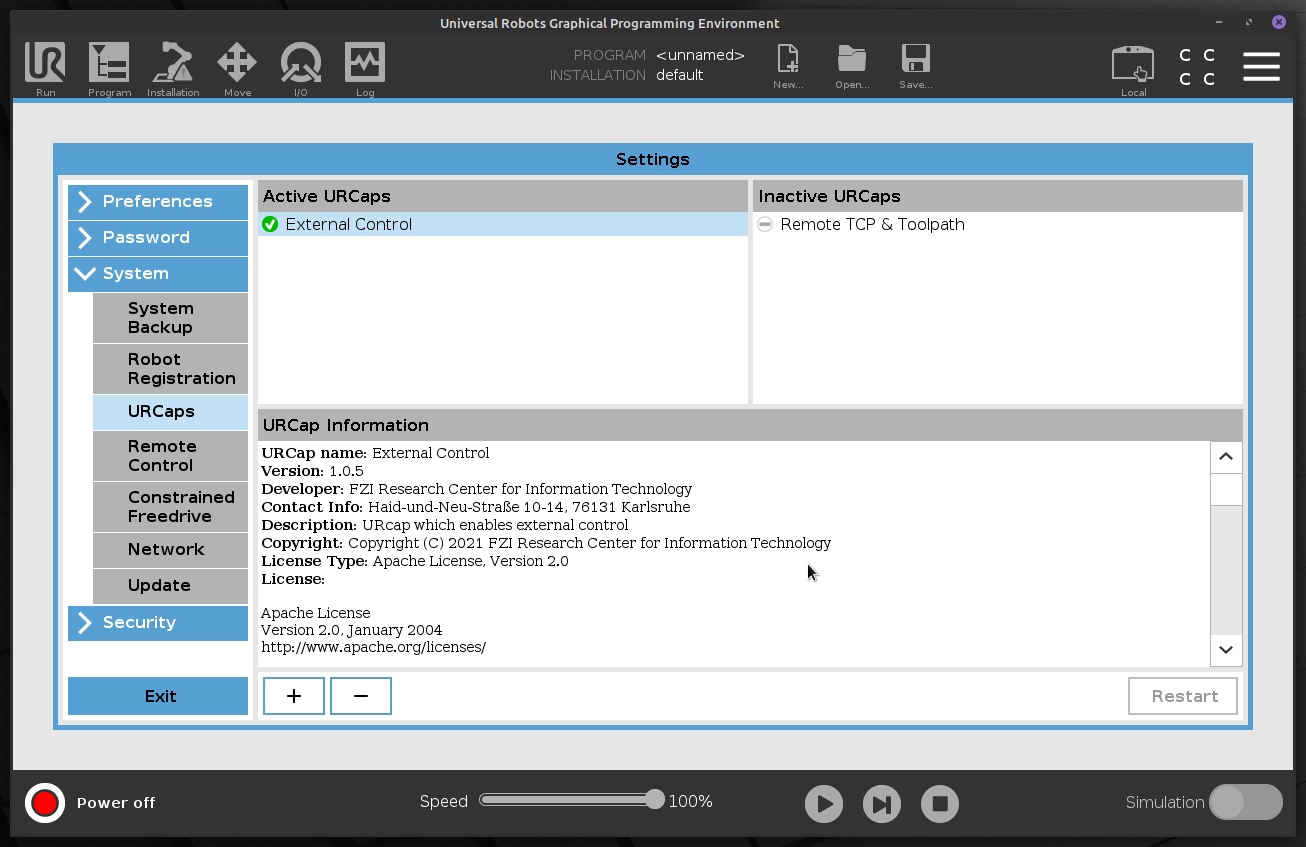
Для связи UR\_RTDE с URSim требуется UR Cap, который можно взять [здесь](https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/blob/master/ur_robot_driver/resources/externalcontrol-1.0.5.urcap). Файл нужно перенести в директорию программ робота.

Далее открыть настройки URSim >> System >> URCaps

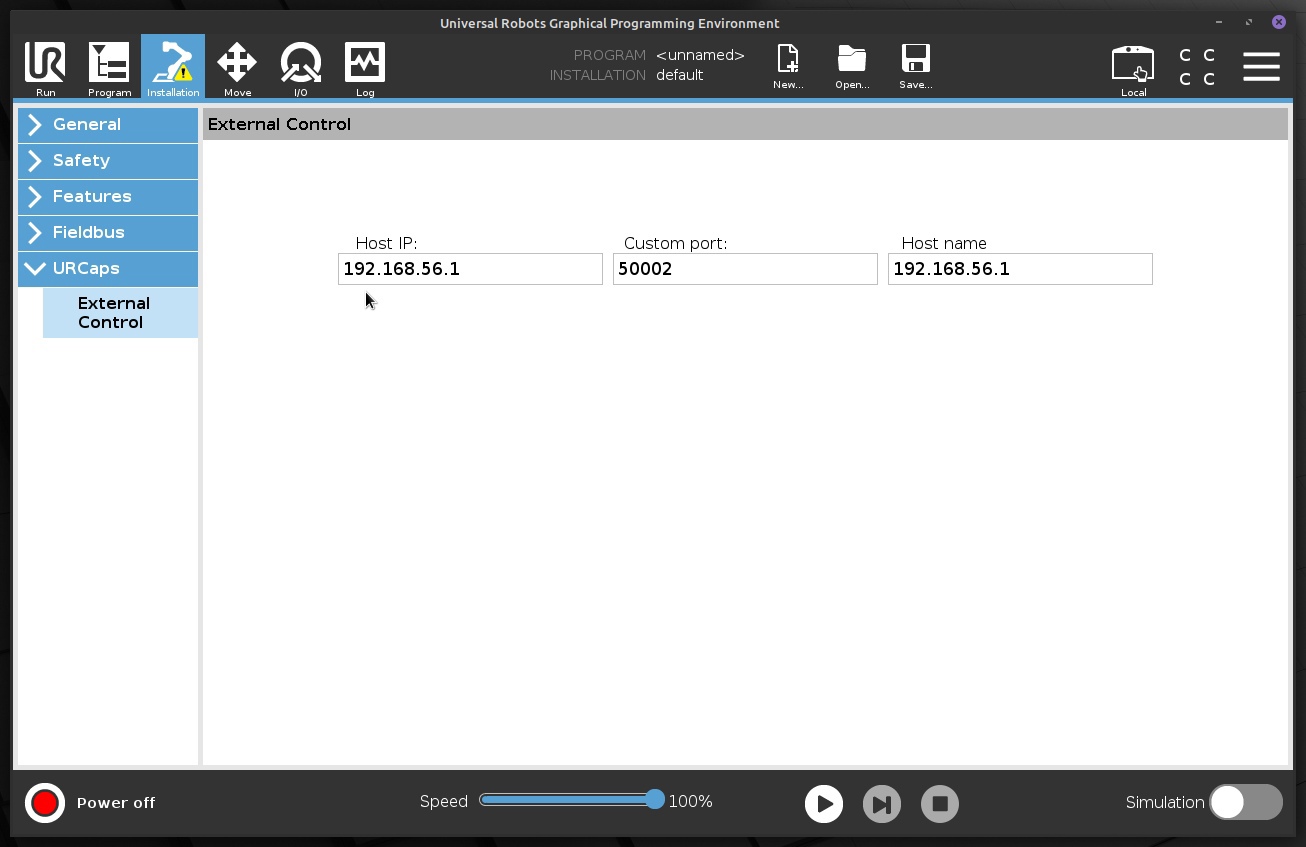


Кликнуть по плюсу, в списке должен быть скачанный файл, выбрать его и перезапустить URSim

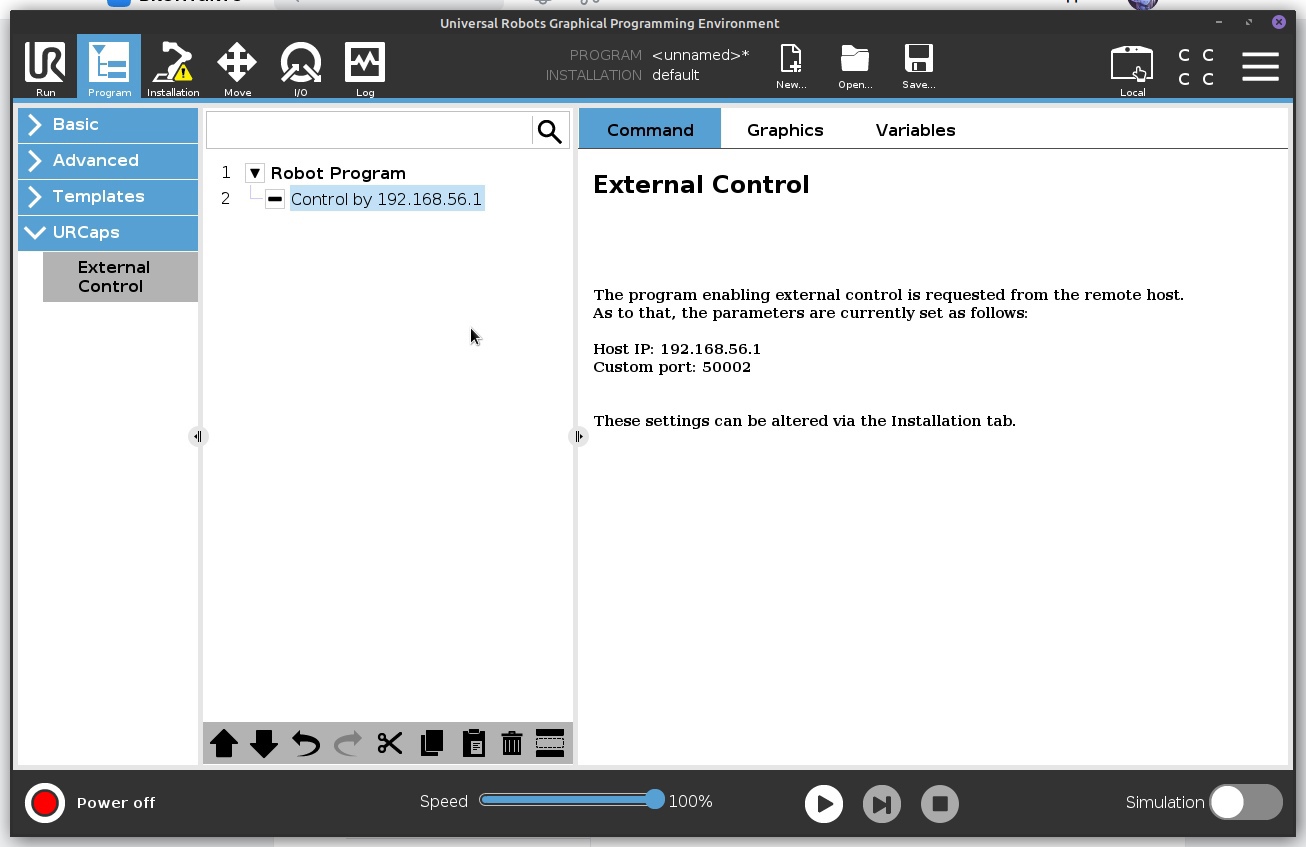




После перезапуска во вкладке installation должен появится пункт External Control URCaps. Теперь можно установить IP-адрес компьютера, на котором будет работать ROS driver.



Для использования нового URCaps нужно создать программу и вставить External Control ноду в дерево программы.



1. Исследование функции speedJ

На вход функция принимает вектор угловых скоростей для каждого из звеньев, а также ускорение, с которым звено будет выходить на заданную скорость.

При разном времени цикла обмена данными с симулятором получились следующие графики

У всех трех случаев ускорение для выхода на заданную скорость равно . Исходя из линейности графика скорости на последней диаграмме можно сделать вывод о том, что для ускорение недостаточно. Можно заметить это по графику ускорения.

При различном количестве движущихся звеньев реальные скорости получаются следующими:

При увеличении максимального ускорения и при параметре ускорения, передаваемого в функцию speedJ, равного , ускорение достигает установленного предела и скорость начинает возрастать линейно.

При

Выводы: наиболее стабильное следование траектории получается при ускорение передаваемое в функцию speedJ должно быть такое, что , где - задаваемая скорость, - текущая скорость.

1. Метод управления движением с вычислением управляющего воздействия на основе интегрирования ускорений

В статье «Trajectory Tracking Control of UR5 Robot: a PD with Gravity Compensation and Sliding Mode Control Comparison» рассматривается следующая схема управления движением.



В статье рассматривается 2 метода управления: PD-регулятор с компенсацией гравитации и Sliding mode control (управление со скользящим режимом). В Position controller происходит вычисления крутящего момента:

- крутящий момент для метода PD-регулятора

- крутящий момент для метода управления с со скользящим режимом

- положительно определенные матрицы, представляющие пропорциональные и дифференциальные коэффициенты усиления.

- оценка координат сочленений и ее производная

- расчетные координаты сочленений для следования по траектории и производная

- вектор нелинейных эффектов

-матрица инерции

UR5 имеет динамическую модель

Численно интегрируя можно получить , которые используются для следующего обновления динамической модели, после добавления белого шума получатся .

1. Реализация функции вычисления управляющего воздействия для заданного ускорения.

Положим -текущая скорость, – заданное ускорение.

Тогда для управляющее воздействие для функции speedJ будет вычисляться по формуле:

Реализация функции:

std::vector<*double*> calc\_speed(RTDEReceiveInterface \**rtde\_receive*, *double* *dt*,

                               std::vector<*double*> *qdd*) {

*auto* res = rtde\_receive->getActualQd();

  for (*size\_t* i = 0; i < res.size(); i++) res[i] += qdd[i] \* dt;

  return res;

}

Результат работы:

6.

На мой взгляд эту функцию можно применить в формуле для PD регулятора как . Т.е. как первую производную от угла поворота.