Документация LWC

Lightweight Cobot

Даниил Грабарь

Table of contents

1. Начало работы

1.1 Обзор

Lightweight Cobot (LWC) — открытая система управления коллаборативным роботом **KUKA LBR IIWA 7 R800** на базе **ROS 2 Jazzy**. Поддерживает работу как с физическим роботом (через протокол FRI), так и в виртуальной среде (симулятор Webots). В состав проекта входят CLI-инструмент `cobot`, аппаратный интерфейс ROS 2 Control, планировщик движений MoveIt 2 и REST/MCP API для интеграции с AI-агентами.

1.1.1 Репозитории

Платформа	Ссылка	Статус
GitVerse (предпочтительный)	daniel-robotics/lightweight-cobot	основной
GitHub	Daniel-Robotic/lightweight-cobot	зеркало

1.1.2 Онлайн документация

- GitVerse Pages — основная
- GitHub Pages — зеркало

1.1.3 Предпосылки

- Физический робот **KUKA LBR IIWA 7 R800** — если планируете работу с реальным оборудованием
- Либо желание работать только в симуляторе **Webots** — физический робот не нужен
- ПК или сервер с **Ubuntu 24.04** (или подключение к существующему серверу по SSH)
- Доступ к сети, в которой находится контроллер робота

1.1.4 Порядок шагов

1. **Настройка SunriseWorkbench** — подготовка контроллера KUKA: подключение кабелей, загрузка программы `ServerFriRos2`, настройка сетевых параметров. *(Только для физического робота)*
2. **Подключение к серверу** — выбор варианта развёртывания (локально или удалённо), SSH-подключение к серверу управления.
3. **Установка проекта** — скачивание и установка LWC через `curl`, `git` или вручную. После установки запустите `cobot setup` — мастер проведёт по оставшимся шагам автоматически.
4. **Конфигурация системы** — настройка `cobot-setting.yaml`: IP робота, порты, инструменты. Выполняется на шаге 2 мастера `cobot setup`, либо в любой момент позже через `cobot robot-setup`.
5. **CLI-инструмент cobot** — все команды управления проектом: запуск, сборка, обновление.

> **Только симуляция?**
>
> Если физический робот недоступен, пропустите шаг 1 и начните сразу с установки проекта. Запустите симулятор командой `cobot run --simulate`.

> **`cobot setup` сделает шаги 3–4 автоматически**
>
> После установки проекта достаточно выполнить `cobot setup` — мастер последовательно предложит: настройку документации, параметры робота (`cobot-setting.yaml`) и выбор среды сборки (ROS2 или Docker).

1.2 Настройка SunriseWorkbench

> **Только для физического робота**
>
> Этот раздел актуален только при работе с реальным роботом KUKA LBR IIWA 7. Для симуляции переходите к Установке проекта.

1.2.1 Настройка физического оборудования

Подключение Ethernet

Подключите кабель Ethernet от вашего ПК (или сервера управления) к одному из сетевых портов контроллера KUKA:

- **KLI** (KUKA Line Interface) — основной порт, используется для управления и программирования
- **KONI** (KUKA Optional Network Interface) — дополнительный порт, используется для FRI

Вы можете подключить оба порта одновременно — при конфигурации сервера будет выбор, какой интерфейс использовать.

Подключение KONI и KLI

1.2.2 Синхронизация проекта в SunriseWorkbench

Проверка наличия ServerFriRos2

Убедитесь, что в вашем Sunrise-проекте присутствует файл `ServerFriRos2.java`. Если файл отсутствует — скачайте его из репозитория. Он находится по пути `src/iiwa_sunrise/ServerFriRos2.java`.

curl wget

```
curl -fsSL https://gitverse.ru/api/repos/daniel-robotics/lightweight-cobot/raw/branch/master/src/iiwa_sunrise/ServerFriRos2.java \
     -o ServerFriRos2.java

wget -O ServerFriRos2.java \
     https://gitverse.ru/api/repos/daniel-robotics/lightweight-cobot/raw/branch/master/src/iiwa_sunrise/ServerFriRos2.java
```

После скачивания добавьте файл в Sunrise-проект и выполните синхронизацию с контроллером.

Синхронизация с контроллером

Откройте **SunriseWorkbench** и нажмите кнопку синхронизации проекта:

Кнопка синхронизации проекта

Перед синхронизацией убедитесь, что ПК и контроллер KUKA находятся в одной сети. Текущие сетевые параметры контроллера можно быстро проверить прямо в SunriseWorkbench:

Быстрый просмотр параметров сети

1.2.3 Настройка ServerFriRos2

Откройте файл `ServerFriRos2.java` в SunriseWorkbench и измените следующие параметры в соответствии с вашей сетевой конфигурацией:

```
// IP-адрес интерфейса KONI
KONI_IP = "192.170.10.10";
```

```
// IP-адрес интерфейса KLI
KLI_IP  = "192.168.21.31";

// Нулевая позиция (все суставы в 0°)
ZERO_POSITION = {0, 0, 0, 0, 0, 0, 0};

// Рабочая позиция для мониторинга
MONITOR_WORKING_POSITION = {0, 0, 0, -1.57, 0, 1.57, 0};

// Инструмент, используемый по умолчанию
@Named("tool1")
```

Важно

IP-адреса должны совпадать с реальными адресами интерфейсов вашего контроллера. Неверный IP приведёт к тому, что FRI-соединение не установится.

После внесения изменений повторно выполните синхронизацию проекта с контроллером.

Следующий шаг: Подключение к серверу управления

1.3 Подключение к серверу

1.3.1 Варианты развёртывания

Проект можно развернуть двумя способами:

Вариант	Описание	Когда использовать
Локально	Установка на вашем ПК	Разработка, симуляция, отладка
Удалённо (сервер)	Установка на выделенном сервере, подключённом к контроллеру KUKA	Работа с реальным роботом

При удалённом варианте вы управляете сервером через **SSH-соединение** со своего ПК.

1.3.2 Программы для SSH-подключения

Выберите любую из программ:

- **Termius** — кроссплатформенный SSH-клиент с удобным GUI
- **MobaXterm** — многофункциональный терминал для Windows
- **PuTTY** — классический SSH-клиент для Windows
- **Встроенный терминал / командная строка** — описано ниже

Инструкции по настройке Termius, MobaXterm и PuTTY смотрите в их официальной документации.

1.3.3 Данные для подключения

```
IP-адрес:       192.168.21.1
Имя пользователя: cobot
Пароль:         12345678
```

> ⚠ **Требование к сети**
>
> Ваш ПК должен находиться в **одной сети/подсети с роботом** (например, в сети КнАГУ). Без этого соединение установить не удастся.

1.3.4 Подключение через встроенный терминал

Linux Windows

Подойдёт любой дистрибутив. Откройте терминал и выполните:

```
ssh cobot@192.168.21.1
```

Требуется **Windows 10** или новее (встроенный SSH-клиент). Откройте **Командную строку** или **PowerShell** и выполните:

```
ssh cobot@192.168.21.1
```

После выполнения команды появится запрос пароля:

```
cobot@192.168.21.1's password:
```

Введите пароль `12345678` — символы не отображаются во время набора, это нормальное поведение из соображений безопасности. Нажмите `Enter ⏎`.

При успешном подключении вы увидите приглашение командной строки сервера:

```
cobot@server:~$
```

Следующий шаг: Установка проекта

1.4 Установка проекта

1.4.1 Быстрая установка

Самый простой способ — установить через `curl` одной командой. Убедитесь, что `curl` установлен:

```
sudo apt update && sudo apt upgrade -y && sudo apt install curl
```

Перейдите в домашнюю директорию и запустите скрипт установки:

Стабильная версия (master) **Dev-версия** (dev)

```
cd ~
curl -fsSL https://gitverse.ru/api/repos/daniel-robotics/lightweight-cobot/raw/branch/master/install.sh | bash

cd ~
curl -fsSL https://gitverse.ru/api/repos/daniel-robotics/lightweight-cobot/raw/branch/dev/install.sh | bash -s dev
```

1.4.2 Установка через Git

Проект доступен как на GitVerse (предпочтительно), так и на GitHub.

> **Не знакомы с Git?**
>
> Если вы впервые работаете с Git и GitHub, рекомендуем прочитать эту статью на Хабре — там всё объясняется с нуля.

Клонируйте репозиторий и запустите скрипт установки:

GitVerse GitHub

```
cd ~
git clone https://gitverse.ru/daniel-robotics/lightweight-cobot.git
cd ~/lightweight-cobot
sudo chmod +x ./install.sh
./install.sh

cd ~
git clone https://github.com/Daniel-Robotic/lightweight-cobot.git
cd ~/lightweight-cobot
sudo chmod +x ./install.sh
./install.sh
```

1.4.3 Ручная установка

Если ни `curl`, ни `git` недоступны — скачайте архив проекта вручную со страницы репозитория (кнопка «Скачать ZIP»), разархивируйте и выполните:

```
cd ~/lightweight-cobot
sudo chmod +x ./install.sh
./install.sh
```

1.4.4 Процесс установки

Скрипт `install.sh` автоматически установит:

- **Git** — система контроля версий
- **Docker** — контейнеризация для изолированного запуска
- **CLI-инструмент** `cobot` — основной инструмент управления проектом
- Системные зависимости Ubuntu

Перезагрузка после установки

После завершения скрипта может потребоваться перезагрузка, чтобы заработал Docker:

```
sudo reboot now
```

Следите за выводом в терминале — скрипт сам сообщит, нужна ли перезагрузка.

Если лог пуст

Если после запуска скрипта не выводится никакой информации, обновите среду bash и продолжите настройку вручную:

```
source ~/.bashrc   # обновление переменных окружения
cobot setup        # продолжение настройки системы
```

1.4.5 Проверка установки

После завершения убедитесь, что `cobot` доступен:

```
cobot -h
```

Если команда выводит список доступных подкоманд — установка прошла успешно.

Следующий шаг: Конфигурация системы

1.5 Конфигурация системы

1.5.1 Главный конфигурационный файл

Все параметры системы хранятся в одном файле — `cobot-setting.yaml` в корне проекта. Это единственный источник истины для IP-адреса робота, портов, путей к конфигам, настроек планировщика и веб-сервера.

> **Не нужно настраивать вручную до установки**
>
> При выполнении `cobot setup` мастер автоматически предложит настроить этот файл на **шаге 2** — вам не нужно делать это заранее. Возвращайтесь к этому разделу когда захотите изменить параметры после первоначальной установки.

> **Не редактируйте файл вручную**
>
> Используйте только команду `cobot robot-setup` — интерактивный мастер валидирует значения и не допускает синтаксических ошибок. Ручное редактирование YAML может привести к ошибкам парсинга и невозможности запуска системы.

```
cobot robot-setup
```

1.5.2 Блок `robot` — параметры робота

Отвечает за соединение с физическим контроллером KUKA по протоколу FRI.

```yaml
robot:
  name: "iiwa7"
  ip: "192.170.10.2"
  port: 30200
  fri_cycle_ms: 10
  joint_position_tau: 0.04
  joint_velocity_tau: 0.01
  active_controller: "jtc"
  description: pkg://iiwa_description/urdf/iiwa7.urdf.xacro
```

Параметр	Описание	Рекомендации
`name`	Модель робота	Не менять — `iiwa7`
`ip`	IP-адрес контроллера KUKA	**Изменить** на реальный IP вашего контроллера
`port`	UDP-порт FRI	По умолчанию `30200` — менять только при конфликте портов
`fri_cycle_ms`	Период цикла FRI: `5` мс = 200 Гц, `10` мс = 100 Гц	`10` для стабильной работы, `5` для высокоточных задач
`joint_position_tau`	EMA-фильтр позиций [с] — сглаживает команды перед отправкой	Уменьшать для более быстрой реакции, увеличивать при вибрациях
`joint_velocity_tau`	EMA-фильтр скорости [с] — убирает выбросы конечных разностей	Аналогично `joint_position_tau`
`active_controller`	Режим управления: `jtc` (MoveIt / JointTrajectory) или `forward` (прямое управление)	`jtc` для большинства задач
`description`	Путь к URDF-описанию робота	Не менять

1.5.3 Блок `digital_twin` — симулятор

Настройки виртуальной среды Webots и визуализации в RViz.

```yaml
digital_twin:
  webots:
    world: pkg://iiwa_description/worlds/iiwa.wbt
    transform: "-0.25 0 0.79"
    rotation: "0 0 1 0"
    controller_timer: "50"
    cameras:
      - pkg://iiwa_config/config/cameras/d455_top.yaml
  rviz:
    config: pkg://iiwa_config/config/rviz/rviz_moveit.rviz
```

Параметр	Описание
`webots.world`	Путь к `.wbt`-миру симулятора
`webots.transform`	Смещение базы робота в мире `[x y z]` в метрах
`webots.rotation`	Ориентация базы `[x y z угол]` в радианах
`webots.cameras`	Список YAML-конфигов подключённых камер
`rviz.config`	Путь к конфигурации RViz

1.5.4 Блок `tool` — активный инструмент

Указывает, какой захват или инструмент закреплён на роботе.

```yaml
tool:
  active: "patron"
```

Значение	Описание
`none`	Без инструмента
`patron`	Захват «Patron» (патрон)

Список доступных инструментов находится в файле `src/iiwa_config/config/tools.yaml`. Для добавления нового инструмента необходимо описать его там, после чего указать имя в `tool.active`.

1.5.5 Блок `planning` — планирование движений

Параметры MoveIt 2 и конфигурации траекторного планировщика.

```yaml
planning:
  pose_link: "tcp"
  planning_group: "iiwa_arm"
  default_frame: "base_link"
```

```
  default_planner: "ompl"
  planning_attempts: 3
```

Параметр	Описание	Рекомендации
`pose_link`	TCP-линк для декартовых целей	Соответствует фрейму в URDF — не менять без изменения URDF
`planning_group`	Группа планирования из SRDF	Не менять — `iiwa_arm`
`default_frame`	Система отсчёта по умолчанию	Не менять — `base_link`
`default_planner`	Планировщик: `ompl`, `pilz_industrial_motion_planner`	`ompl` — универсальный; `pilz` — для предсказуемых траекторий
`planning_attempts`	Число попыток планирования при неудаче	Увеличивать при сложных траекториях

1.5.6 Блок `web` — REST API и MCP-сервер

FastAPI-сервер для управления роботом через HTTP и MCP (интеграция с AI-агентами).

```
web:
  enabled: true
  host: "0.0.0.0"
  port: 8007
  endpoints: pkg://iiwa_config/config/api_endpoints.yaml
  joint_limits: pkg://iiwa_config/config/moveit/joint_limits.yaml
```

Параметр	Описание
`enabled`	Включить (`true`) или отключить (`false`) веб-сервер
`host`	Адрес прослушивания: `0.0.0.0` — все интерфейсы, `127.0.0.1` — только локально
`port`	Порт HTTP API (по умолчанию `8007`)
`endpoints`	Путь к описанию REST-эндпоинтов
`joint_limits`	Путь к файлу ограничений суставов для валидации команд

После запуска REST API доступен по адресу `http://<host>:8007`, MCP — по пути `/mcp`.

1.5.7 Блок `foxglove` — мониторинг через Foxglove Studio

Foxglove Studio — инструмент визуализации и мониторинга ROS 2 топиков в реальном времени.

```
foxglove:
  enabled: true
  port: 8765
  debug: false
  address: 0.0.0.0
```

Параметр	Описание
`enabled`	Включить или отключить Foxglove Bridge
`port`	WebSocket-порт для подключения Foxglove Studio (по умолчанию `8765`)
`debug`	Подробное логирование bridge-процесса
`address`	Адрес прослушивания WebSocket

Остальные параметры блока (`tls`, `topic_whitelist`, `min_qos_depth` и др.) относятся к продвинутой конфигурации и в большинстве случаев менять их не требуется.

1.5.8 Что менять, что оставить

	Параметр	Действие
✅	`robot.ip`	**Обязательно изменить** на IP вашего контроллера
✅	`robot.fri_cycle_ms`	Выбрать `10` (стандарт) или `5` (высокая частота)
✅	`tool.active`	Указать активный инструмент
✅	`web.enabled`	Выключить (`false`), если веб-интерфейс не нужен
⚠️	`robot.active_controller`	Менять только при намеренном переключении режима управления
⚠️	`planning.*`	Менять при необходимости другого планировщика или параметров
❌	`robot.description`	Не трогать — путь к URDF
❌	`controller.moveit.*`	Не трогать — пути к конфигам MoveIt внутри пакетов
❌	`digital_twin.webots.world`	Не трогать без знания структуры Webots-миров

1.6 Основные концепции

1.6.1 Архитектура системы

> **Раздел в разработке**
>
> Подробное описание архитектуры системы готовится.

Система LWC построена как набор взаимосвязанных ROS 2 пакетов. Ключевые компоненты:

- **iiwa_bringup** — launch-файлы, точка входа для запуска всей системы
- **iiwa_controller** — hardware interface, реализует связь с контроллером KUKA по протоколу FRI
- **iiwa_planning** — планирование движений на базе MoveIt 2
- **iiwa_web** — REST API и MCP-сервер для внешнего управления
- **iiwa_description** — URDF-описание робота и миры Webots
- **iiwa_config** — все конфигурационные файлы (MoveIt, контроллеры, камеры)
- **iiwa_utils** — вспомогательные Python-утилиты, загрузка конфигов
- **iiwa_msgs** — кастомные ROS 2 типы сообщений (action, srv)

1.6.2 CLI-команды cobot

`cobot` — единая точка входа для управления всем проектом. Все операции с роботом, симулятором, Docker-контейнерами и документацией выполняются через эту команду.

Справка

```
cobot -h
```

Вывод справки разделён на 4 группы команд:

- **Setup commands** — настройка системы
- **Run commands** — запуск проекта
- **Build commands** — сборка ROS 2
- **Management commands** — управление пакетом

Некоторые команды имеют собственные подкоманды. Пример:

```
cobot doc-setup rebuild   # подкоманда rebuild для doc-setup
cobot doc-setup --help    # справка по подкомандам конкретной команды
```

Setup commands

Команды первичной и повторной настройки системы.

`cobot setup`

Мастер-команда первого запуска. Проводит через три шага:

1. Настройка сервера документации
2. Параметры робота (`cobot-setting.yaml`) — IP, порт FRI, инструмент
3. Выбор среды сборки: ROS2 Jazzy нативно или Docker-образ

```
cobot setup
```

Используйте эту команду при первой установке вместо ручного запуска каждой setup-команды.

`cobot local-setup`

Локальная установка ROS 2 Jazzy без Docker: скачивает зависимости через `rosdep` , собирает workspace через `colcon` .

```
cobot local-setup
```

> **Note**
>
> После выполнения обязательно выполните `source ~/.bashrc` или откройте новый терминал.

`cobot docker-setup`

Сборка или скачивание готовых Docker-образов для запуска проекта в изоляции. Доступны два варианта:

- **Сборка из Dockerfile** — дольше, но даёт актуальную версию
- **Скачивание готового образа** — быстрее, использует pre-built образ

```
cobot docker-setup
```

`cobot doc-setup`

Разворачивает локальный MkDocs-сервер документации — полную копию онлайн-документации.

```
cobot doc-setup          # запустить / собрать документацию
cobot doc-setup rebuild  # пересобрать документацию
```

После запуска документация будет доступна по адресу `http://localhost:8000`.

`cobot robot-setup`

Интерактивный мастер настройки файла `cobot-setting.yaml`. Запрашивает IP-адрес робота, порт FRI, активный инструмент и другие параметры.

```
cobot robot-setup
```

> **Tip**
>
> Используйте именно эту команду для изменения конфигурации — она валидирует введённые значения и исключает синтаксические ошибки. Подробнее о параметрах — в разделе Конфигурация системы.

Run commands

`cobot run`

Запускает весь стек: hardware interface, MoveIt 2, RViz и опциональные компоненты (Foxglove, REST API). При запуске предлагает выбор:

- **Docker или локально**
- **Симулятор (Webots) или реальный робот**

```
cobot run                # интерактивный выбор режима
cobot run --simulate     # принудительно запустить симулятор
```

Build commands

`cobot rebuild`

Пересборка ROS 2 workspace через `colcon`. Используется при изменении исходного кода пакетов.

```
cobot rebuild
```

> **Note**
>
> Работает только для локальной установки (не Docker). Эквивалент `colcon build --mixin release`.

`cobot clean`

Удаляет сгенерированные папки сборки. Предлагает выбор, какие именно удалить:

- `build/` — артефакты компиляции
- `install/` — установленные файлы пакетов
- `log/` — логи сборки

```
cobot clean
```

Management commands

`cobot update`

Скачивает последнюю версию проекта с GitVerse и переустанавливает CLI `cobot`.

```
cobot update
```

`cobot delete`

Удаляет компоненты проекта с системы. Предложит выбор: удалить только проект, Docker-образы и контейнеры, или также ROS 2.

```
cobot delete
```

> **Danger**
>
> Операция необратима. Удалённые файлы и Docker-образы потребуют повторной установки.

Управление роботом: Управление через REST API

1.6.3 FRI-протокол

> **Раздел в разработке**
>
> Подробное описание FRI-протокола готовится.

FRI (Fast Robot Interface) — UDP-протокол низкоуровневого управления роботом KUKA в реальном времени. Работает поверх Ethernet и обеспечивает детерминированный цикл обмена данными между внешним ПК и контроллером KUKA.

Основные характеристики

- **Транспорт:** UDP (без гарантии доставки — критично для RT)
- **Период цикла:** 5 мс (200 Гц) или 10 мс (100 Гц), задаётся в `cobot-setting.yaml` → `robot.fri_cycle_ms`
- **Режимы управления:** позиция, момент (крутящий момент), импеданс

Требования к сетевому подключению

> **Важно: цикл 5 мс требует KONI**
>
> Для работы с периодом цикла **5 мс (200 Гц)** необходимо подключение через порт **KONI** (KUKA Optional Network Interface). Порт KLI поддерживает только 10 мс цикл. При использовании KLI устанавливайте `fri_cycle_ms: 10`.

Порт	Мин. цикл	Назначение
KONI	5 мс	Высокочастотное управление, рекомендуется для FRI
KLI	10 мс	Стандартное управление и программирование

1.6.4 Симуляция (Webots)

> **Раздел в разработке**
>
> Подробное описание работы в симуляторе готовится.

Webots — симулятор роботов с открытым исходным кодом. В LWC используется как цифровой двойник робота KUKA LBR IIWA 7: позволяет разрабатывать и отлаживать алгоритмы управления без доступа к физическому роботу.

Ключевые особенности

- Симулятор использует те же ROS 2 топики и интерфейсы, что и реальный робот
- Мир симуляции задаётся в `cobot-setting.yaml` → `digital_twin.webots.world`
- Для запуска: `cobot run --simulate`

Отличия от реального робота

- Нет ограничений по безопасности (можно двигаться быстрее)
- Физика не идеальна — инерция, трение и упругость приближённые
- FRI-протокол не задействован — связь идёт через Webots-драйвер ROS 2

1.6.5 Планирование движений

> **Раздел в разработке**
>
> Подробное описание планирования движений готовится.

LWC использует **MoveIt 2** — стандартный фреймворк планирования движений для ROS 2.

Основные понятия

- **Группа планирования** (`iiwa_arm`) — набор суставов, для которых строится план. Задаётся в SRDF.
- **Планировщик** — алгоритм построения траектории. Доступны:
- `ompl` — универсальный вероятностный планировщик (по умолчанию)
- `pilz_industrial_motion_planner` — детерминированные траектории типа PTP, LIN, CIRC
- **TCP (Tool Center Point)** — точка инструмента, для которой задаётся целевая поза. Настраивается в `cobot-setting.yaml` → `planning.pose_link`
- **Система отсчёта** — координатная система целей. По умолчанию: `base_link`

Настройки в `cobot-setting.yaml`

Параметр	Описание
`planning.default_planner`	Планировщик по умолчанию: `ompl` или `pilz_industrial_motion_planner`
`planning.planning_attempts`	Число попыток при неудаче планирования
`planning.pose_link`	TCP-линк для декартовых целей

1.7 Использование LWC

1.7.1 Управление через ROS2

> **Раздел в разработке**
>
> Подробная документация по управлению через ROS2-топики и action-серверы готовится.

Этот способ предполагает прямую отправку команд в ROS2 через CLI-инструменты (`ros2 topic pub`, `ros2 action send_goal`) или написание собственных ROS2-нод на Python/C++.

Дополнительные ресурсы

- Документация ROS2 Jazzy — официальная документация: топики, сервисы, action-серверы, написание нод
- MATLAB Robotics System Toolbox — управление роботом через ROS2 из MATLAB

1.7.2 Управление через Foxglove Studio

> **Раздел в разработке**
>
> Подробная документация по управлению и мониторингу через Foxglove Studio готовится.

Foxglove Studio — инструмент визуализации и мониторинга ROS2-данных в реальном времени. Подключается к запущенному стеку через WebSocket-bridge (порт `8765` по умолчанию, настраивается в `cobot-setting.yaml` → блок `foxglove`).

Скачать Foxglove Studio

Перейдите на официальный сайт Foxglove и скачайте приложение для своей ОС.

1.7.3 Управление через REST API

REST API предоставляет HTTP-интерфейс для управления роботом из любой среды: Bash, Python, MATLAB, Postman и других инструментов.

Веб-панель (Swagger UI)

Интерактивная документация со встроенным тест-клиентом доступна по адресу:

- **Локально:** http://localhost:8007/docs
- **Удалённо:** `http://ip-сервера:8007/docs`

Подготовка к работе

> **Не знакомы с инструментами?**
>
> - **Bash / curl** — введение в Bash-скрипты
> - **Python** — руководство для начинающих или официальная документация
> - **MATLAB** — официальная документация MATLAB

КОНСТАНТЫ

Для Python и MATLAB определите константы в начале скрипта. Для Bash они указываются непосредственно в командах.

Python MATLAB

```python
import httpx, math

HOST   = "http://localhost:8007"
T_FAST = 10   # тайм-аут для чтения состояния [с]
T_MOVE = 60   # тайм-аут для команд движения [с]
```

```matlab
HOST   = 'http://localhost:8007';
T_FAST = 10;
T_MOVE = 60;
```

Эндпоинты

1. GET `/robot/joint_states` — УГЛЫ СУСТАВОВ

Возвращает текущие значения углов (в радианах) всех 7 суставов. Используйте для мониторинга положения робота перед отправкой команд.

Bash Python MATLAB

```bash
curl -s --max-time 10 http://localhost:8007/robot/joint_states | python3 -m json.tool
```

```python
r = httpx.get(f"{HOST}/robot/joint_states", timeout=T_FAST)
print(r.json()["position"])
```

```matlab
r = webread([HOST '/robot/joint_states'], weboptions('Timeout', T_FAST));
disp(r.position)
```

---

2. GET `/robot/pose` — ТЕКУЩАЯ ПОЗА TCP

Возвращает декартову позу инструментального центра (TCP): координаты `x, y, z` [м] и ориентацию в углах Эйлера [градусы]. Используйте для контроля положения инструмента в пространстве.

Bash **Python** **MATLAB**

```bash
curl -s --max-time 10 http://localhost:8007/robot/pose | python3 -m json.tool
```

```python
r = httpx.get(f"{HOST}/robot/pose", timeout=T_FAST)
pose = r.json()
print(pose)
```

```matlab
r = webread([HOST '/robot/pose'], weboptions('Timeout', T_FAST));
fprintf('x=%.4f  y=%.4f  z=%.4f\n', r.position.x, r.position.y, r.position.z);
fprintf('A=%.4f  B=%.4f  C=%.4f\n', r.orientation.euler_deg.a, r.orientation.euler_deg.b, r.orientation.euler_deg.c);
```

---

3. GET `/robot/positions` — ИМЕНОВАННЫЕ ПОЗИЦИИ

Возвращает список всех сохранённых именованных позиций (name + description). Используйте для получения допустимых значений для `/robot/move/named`.

Bash **Python** **MATLAB**

```bash
curl -s --max-time 10 http://localhost:8007/robot/positions
```

```python
r = httpx.get(f"{HOST}/robot/positions", timeout=T_FAST)
for pos in r.json():
    print(pos["name"], "—", pos["description"])
```

```matlab
r = webread([HOST '/robot/positions'], weboptions('Timeout', T_FAST));
for i = 1:numel(r)
    fprintf('%s — %s\n', r(i).name, r(i).description);
end
```

---

4. POST `/robot/stop` — ЭКСТРЕННАЯ ОСТАНОВКА

Немедленно останавливает текущее движение робота. Используйте при необходимости прервать выполняемую команду.

Bash **Python** **MATLAB**

```bash
curl -s --max-time 10 -X POST http://localhost:8007/robot/stop
```

```python
r = httpx.post(f"{HOST}/robot/stop", timeout=T_FAST)
print(r.json())
```

```matlab
opts = weboptions('RequestMethod', 'post', 'MediaType', 'application/json', 'Timeout', T_FAST);
r = webwrite([HOST '/robot/stop'], struct(), opts);
disp(r.success)
```

---

5. POST `/robot/move/named` — ДВИЖЕНИЕ В ИМЕНОВАННУЮ ПОЗИЦИЮ

Перемещает робота в одну из заранее сохранённых позиций (см. `/robot/positions`). Используйте для воспроизводимых переходов между рабочими позициями.

Параметр	Тип	Описание
`name`	string	Имя позиции
`speed`	float	Масштаб скорости (0.0–1.0)
`accel_scale`	float	Масштаб ускорения (0.0 = по умолчанию)

Bash Python MATLAB

```
curl -s --max-time 60 -X POST http://localhost:8007/robot/move/named \
  -H "Content-Type: application/json" \
  -d '{"name": "home", "speed": 0.1, "accel_scale": 0.0}'
```

```
r = httpx.post(f"{HOST}/robot/move/named",
               json={"name": "home", "speed": 0.1, "accel_scale": 0.0},
               timeout=T_MOVE)
print(r.json())
```

```
opts = weboptions('RequestMethod', 'post', 'MediaType', 'application/json', 'Timeout', T_MOVE);
body = struct('name', 'home', 'speed', 0.1, 'accel_scale', 0.0);
r = webwrite([HOST '/robot/move/named'], body, opts);
disp(r.success)
```

---

6. POST `/robot/move/pose` — ДВИЖЕНИЕ ПО ДЕКАРТОВОЙ ПОЗЕ

Перемещает TCP в заданную точку пространства. Ориентация задаётся углами Эйлера (A, B, C) в радианах. Используйте когда нужно задать точное положение и ориентацию инструмента.

Параметр	Тип	Описание
`x, y, z`	float	Координаты [м]
`a, b, c`	float	Углы Эйлера [рад]
`speed`	float	Масштаб скорости (0.0–1.0)
`planner`	string	Планировщик: `ptp`, `lin`, `circ`
`frame_id`	string	Система отсчёта (пусто = `base_link`)

Bash Python MATLAB

```
curl -s --max-time 60 -X POST http://localhost:8007/robot/move/pose \
  -H "Content-Type: application/json" \
  -d '{"x": 0.4, "y": 0.0, "z": 0.5, "a": 0.0, "b": 3.14159, "c": 0.0, "speed": 0.1, "planner": "ptp"}'
```

```
r = httpx.post(f"{HOST}/robot/move/pose",
               json={"x": 0.4, "y": 0.0, "z": 0.5,
                     "a": 0.0, "b": math.pi, "c": 0.0,
                     "speed": 0.1, "planner": "ptp", "frame_id": ""},
               timeout=T_MOVE)
print(r.json())
```

```
opts = weboptions('RequestMethod', 'post', 'MediaType', 'application/json', 'Timeout', T_MOVE);
body = struct('x', 0.4, 'y', 0.0, 'z', 0.5, ...
              'a', 0.0, 'b', pi,   'c', 0.0, ...
              'speed', 0.1, 'planner', 'ptp', 'frame_id', '');
r = webwrite([HOST '/robot/move/pose'], body, opts);
disp(r.success)
```

---

7. POST `/robot/move/joints` — ДВИЖЕНИЕ ПО УГЛАМ СУСТАВОВ

Перемещает робота в позицию, заданную углами всех 7 суставов (в радианах). Используйте когда нужен прямой контроль над конфигурацией робота без планирования в декартовом пространстве.

Параметр	Тип	Описание
`joints`	float[7]	Углы суставов J1–J7 [рад]
`speed`	float	Масштаб скорости (0.0–1.0)

Bash Python MATLAB

```bash
curl -s --max-time 60 -X POST http://localhost:8007/robot/move/joints \
  -H "Content-Type: application/json" \
  -d '{"joints": [0.0, 0.5, 0.0, -1.5708, 0.0, 1.5708, 0.0], "speed": 0.1}'
```

```python
r = httpx.post(f"{HOST}/robot/move/joints",
               json={"joints": [0.0, 0.5, 0.0, -math.pi/2, 0.0, math.pi/2, 0.0],
                     "speed": 0.1},
               timeout=T_MOVE)
print(r.json())
```

```matlab
opts = weboptions('RequestMethod', 'post', 'MediaType', 'application/json', 'Timeout', T_MOVE);
body = struct('joints', {{0.0, 0.5, 0.0, -pi/2, 0.0, pi/2, 0.0}}, 'speed', 0.1);
r = webwrite([HOST '/robot/move/joints'], body, opts);
disp(r.success)
```

8. POST `/trajectory/send` — ОТПРАВКА ТРАЕКТОРИИ

Выполняет многоточечную траекторию, заданную набором waypoints с метками времени. Используйте для плавного воспроизведения записанных или синтезированных движений.

Параметр	Тип	Описание
`points[].positions`	float[7]	Углы суставов в точке [рад]
`points[].time_from_start`	float	Время от старта [с]
`validate_limits`	bool	Проверять ли ограничения суставов

Bash Python MATLAB

```bash
curl -s --max-time 60 -X POST http://localhost:8007/trajectory/send \
  -H "Content-Type: application/json" \
  -d '{
    "points": [
      {"positions": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "time_from_start": 0.0},
      {"positions": [0.0, 0.5, 0.0, -1.0, 0.0, 1.0, 0.0], "time_from_start": 3.0},
      {"positions": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "time_from_start": 6.0}
    ],
    "validate_limits": true
  }'
```

```python
r = httpx.post(f"{HOST}/trajectory/send",
               json={
                   "points": [
                       {"positions": [0.0]*7,                                "time_from_start": 0.0},
                       {"positions": [0.0, 0.5, 0.0, -1.0, 0.0, 1.0, 0.0], "time_from_start": 3.0},
                       {"positions": [0.0]*7,                                "time_from_start": 6.0},
                   ],
                   "validate_limits": True,
               },
               timeout=T_MOVE)
print(r.json())
```

```matlab
opts = weboptions('RequestMethod', 'post', 'MediaType', 'application/json', 'Timeout', T_MOVE);
p1 = struct('positions', {{0,0,0,0,0,0,0}},        'time_from_start', 0.0);
p2 = struct('positions', {{0,0.5,0,-1.0,0,1.0,0}}, 'time_from_start', 3.0);
p3 = struct('positions', {{0,0,0,0,0,0,0}},        'time_from_start', 6.0);
body = struct('points', {{p1, p2, p3}}, 'validate_limits', true);
r = webwrite([HOST '/trajectory/send'], body, opts);
disp(r)
```

9. POST `/trajectory/send_csv` — ЗАГРУЗКА ТРАЕКТОРИИ ИЗ CSV

Загружает траекторию из CSV-файла и выполняет её. Формат файла: строки — точки, столбцы — углы суставов + время. Используйте для воспроизведения траекторий, подготовленных во внешних инструментах.

Bash Python MATLAB

```bash
curl -s --max-time 60 -X POST http://localhost:8007/trajectory/send_csv \
  -F "file=@trajectory.csv" \
  -F "separator=," \
  -F "validate_limits=true"
```

```python
with open("trajectory.csv", "rb") as f:
    r = httpx.post(f"{HOST}/trajectory/send_csv",
                   files={"file": ("trajectory.csv", f, "text/csv")},
                   data={"separator": ",", "validate_limits": "true"},
                   timeout=T_MOVE)
print(r.json())
```

> **Требует уточнения**
>
> Код ниже использует `matlab.net.http` и может потребовать доработки в зависимости от вашей версии MATLAB.

```matlab
import matlab.net.http.*
import matlab.net.http.io.*

csvData = fileread('trajectory.csv');
part = FormProvider(FormField('file', csvData, 'filename', 'trajectory.csv', ...
                              'content-type', 'text/csv'), ...
                    FormField('separator', ','), ...
                    FormField('validate_limits', 'true'));
req = RequestMessage('POST', [], part);
opts = matlab.net.http.HTTPOptions('ConnectTimeout', T_MOVE, 'ResponseTimeout', T_MOVE);
[resp, ~] = req.send([HOST '/trajectory/send_csv'], opts);
disp(resp.Body.Data)
```

---

10. POST `/trajectory/stop` — ОСТАНОВКА ТРАЕКТОРИИ

Прерывает выполнение текущей траектории. Используйте для аварийной остановки во время воспроизведения.

Bash Python MATLAB

```bash
curl -s --max-time 10 -X POST http://localhost:8007/trajectory/stop
```

```python
r = httpx.post(f"{HOST}/trajectory/stop", timeout=T_FAST)
print(r.json())
```

```matlab
opts = weboptions('RequestMethod', 'post', 'Timeout', T_FAST);
r = webread([HOST '/trajectory/stop'], opts);
disp(r)
```

---

11. GET `/trajectory/logs` — ЛОГ ВЫПОЛНЕНИЯ ТРАЕКТОРИИ

Возвращает последние `n` строк лога выполнения траектории. Используйте для диагностики после выполнения команды.

Bash Python MATLAB

```bash
curl -s --max-time 10 "http://localhost:8007/trajectory/logs?n=20"
```

```python
r = httpx.get(f"{HOST}/trajectory/logs", params={"n": 20}, timeout=T_FAST)
for line in r.json()["lines"]:
    print(line)
```

```matlab
r = webread([HOST '/trajectory/logs'], weboptions('Timeout', T_FAST), 'n', 20);
disp(r.lines)
```

12. POST `/sequences/start` — ЗАПУСК ПОСЛЕДОВАТЕЛЬНОСТИ ДВИЖЕНИЙ

Запускает автоматическую последовательность движений из JSON-конфига (`motion_sequence_config.json`). Используйте для воспроизведения повторяющихся циклов работы.

Параметр	Тип	Описание
`config`	file	JSON-файл конфигурации последовательности
`n_iterations`	int	Число повторений (0 = бесконечно)
`delay_between_iterations`	float	Пауза между итерациями [с]

Bash Python MATLAB

```bash
curl -s --max-time 10 -X POST http://localhost:8007/sequences/start \
  -F "config=@motion_sequence_config.json" \
  -F "n_iterations=3" \
  -F "delay_between_iterations=5.0"
```

```python
with open("motion_sequence_config.json", "rb") as f:
    r = httpx.post(f"{HOST}/sequences/start",
                   files={"config": ("config.json", f, "application/json")},
                   data={"n_iterations": "3", "delay_between_iterations": "5.0"},
                   timeout=T_FAST)
print(r.json())
```

> ⚠ **Требует уточнения**
>
> Код ниже использует `matlab.net.http` и может потребовать доработки в зависимости от вашей версии MATLAB.

```matlab
import matlab.net.http.*
import matlab.net.http.io.*

jsonData = fileread('motion_sequence_config.json');
part = FormProvider(FormField('config', jsonData, 'filename', 'config.json', ...
                              'content-type', 'application/json'), ...
                    FormField('n_iterations', '3'), ...
                    FormField('delay_between_iterations', '5.0'));
req = RequestMessage('POST', [], part);
opts = matlab.net.http.HTTPOptions('ConnectTimeout', T_FAST, 'ResponseTimeout', T_FAST);
[resp, ~] = req.send([HOST '/sequences/start'], opts);
disp(resp.Body.Data)
```

13. GET `/sequences/status` — СТАТУС ПОСЛЕДОВАТЕЛЬНОСТИ

Возвращает текущий статус выполнения последовательности. Используйте для polling-мониторинга выполнения.

Bash Python MATLAB

```bash
curl -s --max-time 10 http://localhost:8007/sequences/status
```

```python
r = httpx.get(f"{HOST}/sequences/status", timeout=T_FAST)
print(r.json())
```

```matlab
r = webread([HOST '/sequences/status'], weboptions('Timeout', T_FAST));
disp(r.status)
```

14. GET `/sequences/logs` — ЛОГ ПОСЛЕДОВАТЕЛЬНОСТИ

Возвращает последние `n` строк лога выполнения последовательности.

Bash **Python** **MATLAB**

```
curl -s --max-time 10 "http://localhost:8007/sequences/logs?n=50"

r = httpx.get(f"{HOST}/sequences/logs", params={"n": 50}, timeout=T_FAST)
for line in r.json()["lines"]:
    print(line)

r = webread([HOST '/sequences/logs'], weboptions('Timeout', T_FAST), 'n', 50);
disp(r.lines)
```

---

15. POST `/sequences/stop` — ОСТАНОВКА ПОСЛЕДОВАТЕЛЬНОСТИ

Прерывает выполнение текущей последовательности движений.

Bash **Python** **MATLAB**

```
curl -s --max-time 10 -X POST http://localhost:8007/sequences/stop

r = httpx.post(f"{HOST}/sequences/stop", timeout=T_FAST)
print(r.json())

opts = weboptions('RequestMethod', 'post', 'Timeout', T_FAST);
r = webread([HOST '/sequences/stop'], opts);
disp(r)
```

2. Sunrise Workbench

2.1 Обзор Sunrise Workbench

Для программирования KUKA LBR IIWA 7 необходимо использовать специализированное программное обеспечение **SunriseWorkbench** — интегрированную среду разработки на базе Eclipse, предназначенную для написания, отладки и развёртывания управляющих программ на контроллере KUKA Sunrise Cabinet.

Об установщике

Дистрибутив SunriseWorkbench не находится в открытом доступе. Обратитесь к системному администратору за получением установочного пакета и уточнением его расположения.

В данном руководстве представлены инструкции по установке SunriseWorkbench на следующие операционные системы:

- Windows
- Linux

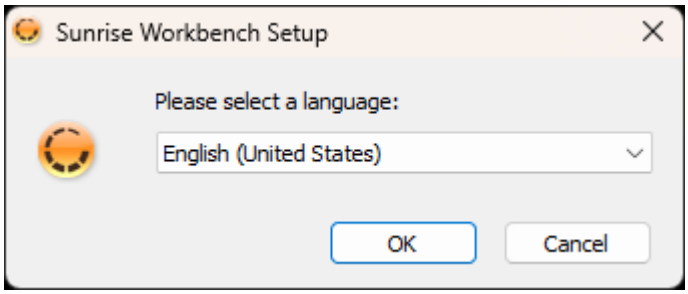
2.2 Установка

2.2.1 Установка на Windows

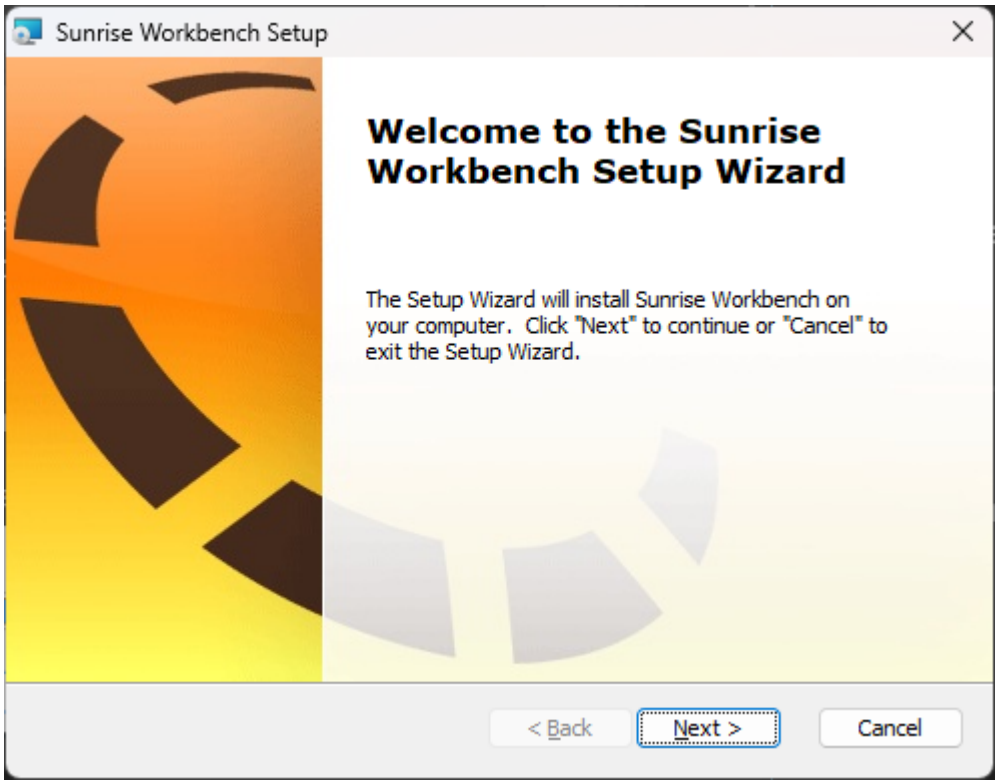
В данном разделе описан процесс установки SunriseWorkbench на операционную систему Windows.

Установка SunriseWorkbench

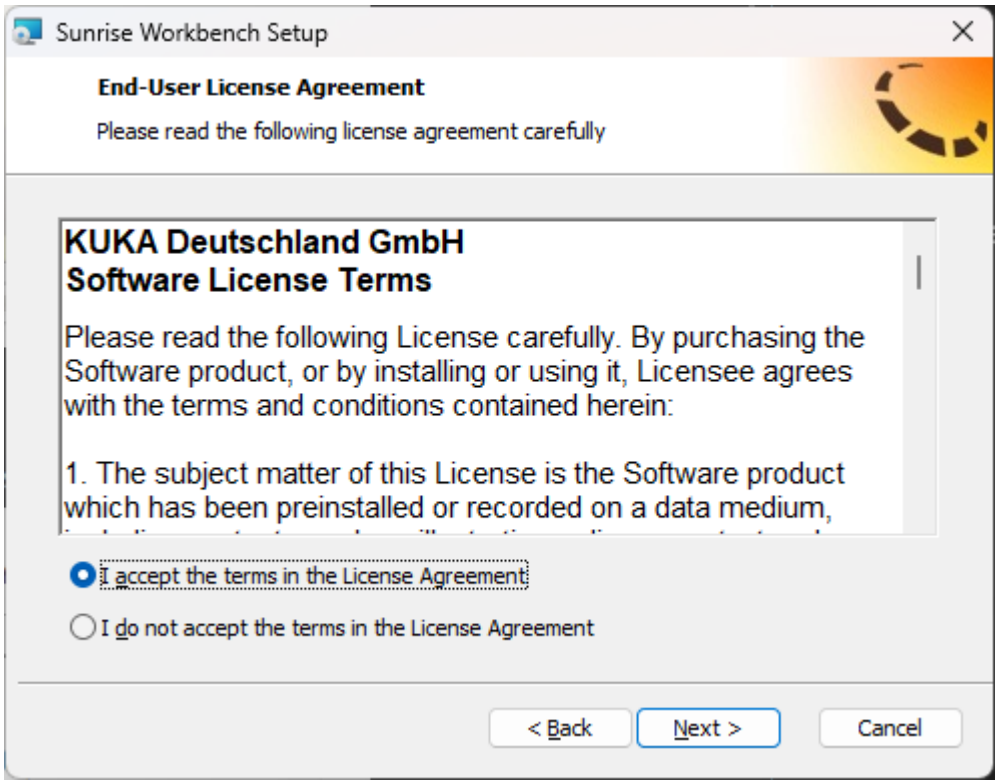
Шаг 1. Запустите установщик SunriseWorkbench. В появившемся окне выберите язык установки (по умолчанию – английский) и нажмите **OK**.



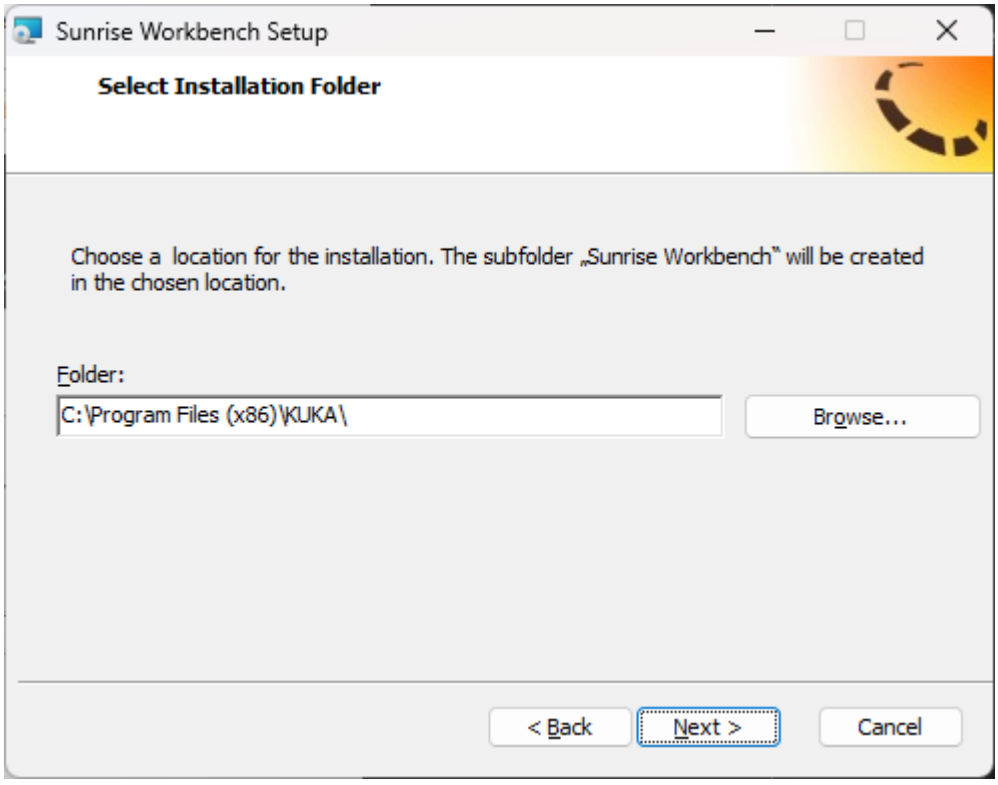
Шаг 2. В приветственном окне мастера установки нажмите **Next**.



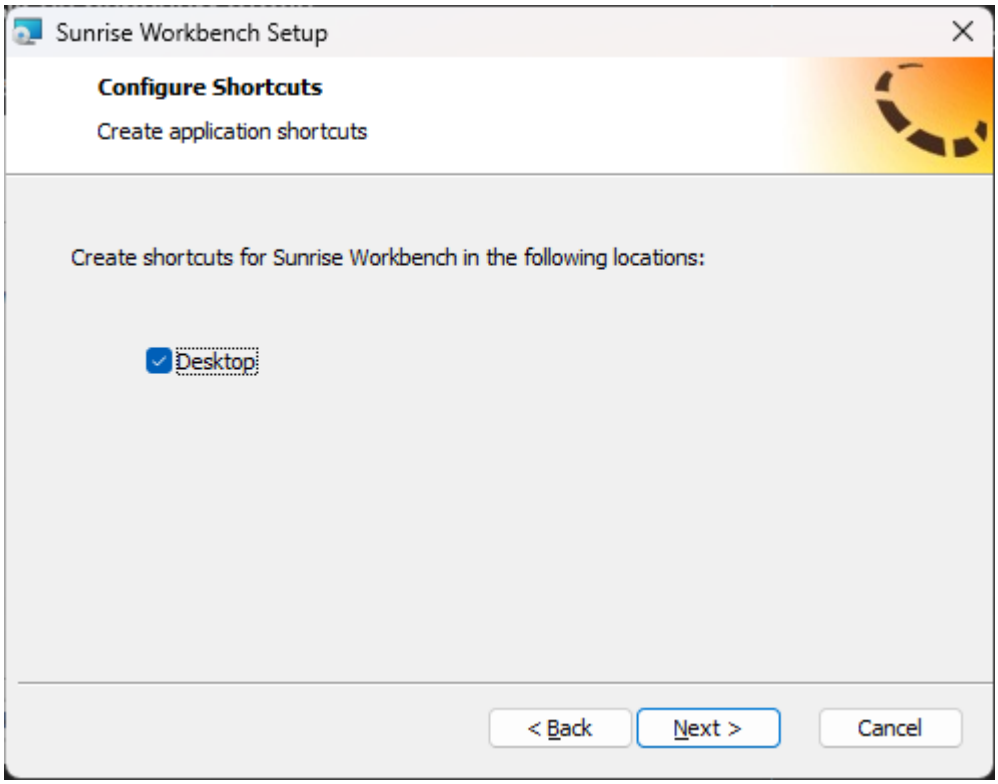
Шаг 3. Ознакомьтесь с условиями лицензионного соглашения и подтвердите принятие.



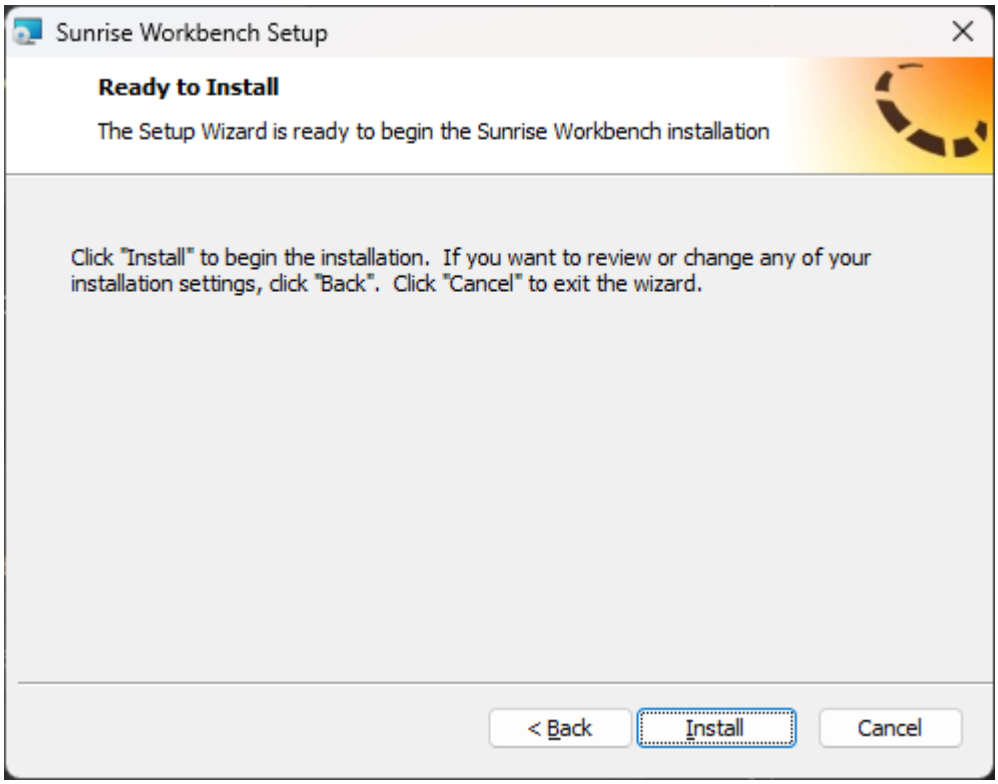
Шаг 4. Путь установки рекомендуется оставить по умолчанию.



Шаг 5. Для удобства дальнейшей работы установите флажок **Create Desktop Shortcut**.

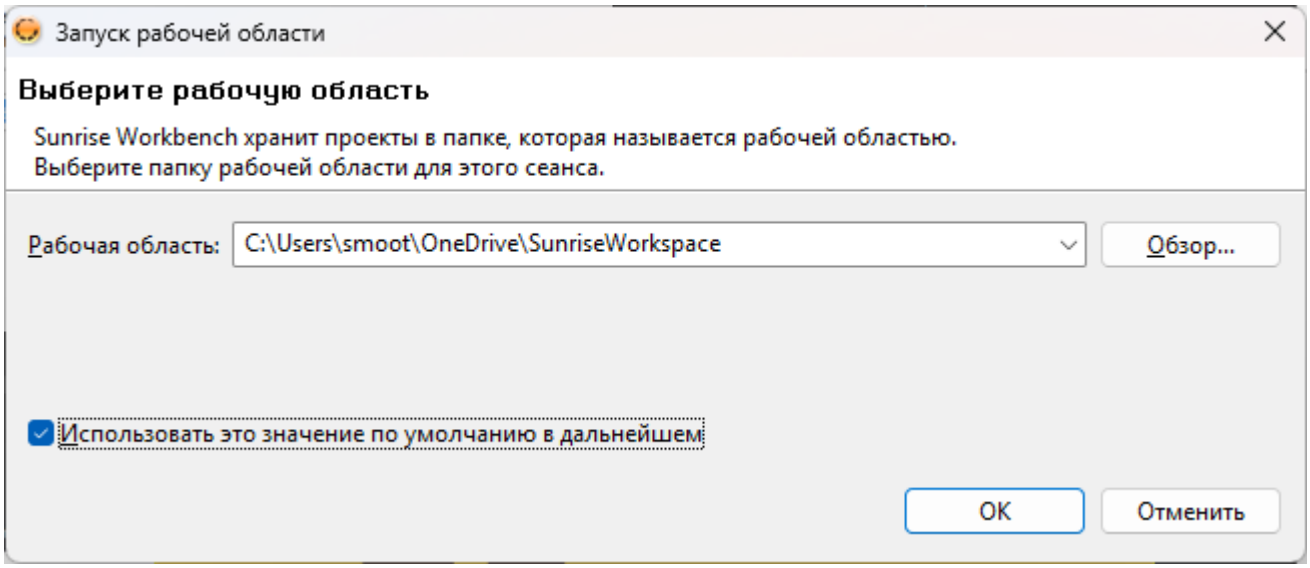


Шаг 6. Нажмите **install** и дождитесь завершения установки всех необходимых компонентов.



Первый запуск

Шаг 7. При первом запуске приложение предложит указать путь к рабочему пространству (workspace). Рекомендуется оставить путь по умолчанию и установить флажок **Использовать это значение по умолчанию в дальнейшем**.



Шаг 8. После загрузки главного окна нажмите **New Sunrise Project** для создания нового проекта.

Дальнейшая настройка

Подробные инструкции по конфигурации проекта представлены в разделе Создание нового проекта. Для полноценной работы также рекомендуется выполнить Установку библиотек.

2.2.2 Linux

Установка на Linux

Поскольку SunriseWorkbench совместима только с операционной системой Windows, для работы на Linux необходимо установить утилиту-эмулятор, обеспечивающую запуск Windows-приложений.

На сегодняшний день существует несколько популярных решений:

- **PortProton**
- **ProtonPlus**
- **Bottles**

Установка любого из перечисленных эмуляторов описана в разделе Установка эмулятора Windows.

Установка эмулятора Windows

Для запуска Windows-приложений на Linux используется эмулятор **PortProton**, устанавливаемый через пакетный менеджер **Flatpak** из репозитория **Flathub**.

УСТАНОВКА FLATPAK

Для Ubuntu 18.10 и более поздних версий выполните:

Bash

```
sudo apt update && sudo apt upgrade -y
sudo apt install flatpak
```

УСТАНОВКА ПЛАГИНА ДЛЯ GNOME SOFTWARE

Для поддержки Flatpak-пакетов в Центре приложений GNOME:

Bash

```
sudo apt install gnome-software-plugin-flatpak
```

ПОДКЛЮЧЕНИЕ РЕПОЗИТОРИЯ FLATHUB

Bash

```
flatpak remote-add --if-not-exists flathub https://dl.flathub.org/repo/flathub.flatpakrepo
```

> ⚠ **Перезагрузка**
>
> После добавления репозитория необходимо перезагрузить систему для применения изменений.

УСТАНОВКА PORTPROTON

Установить PortProton можно двумя способами: через терминал или через Центр приложений GNOME.

Через терминал:

Bash

```
flatpak install flathub ru.linux_gaming.PortProton
```

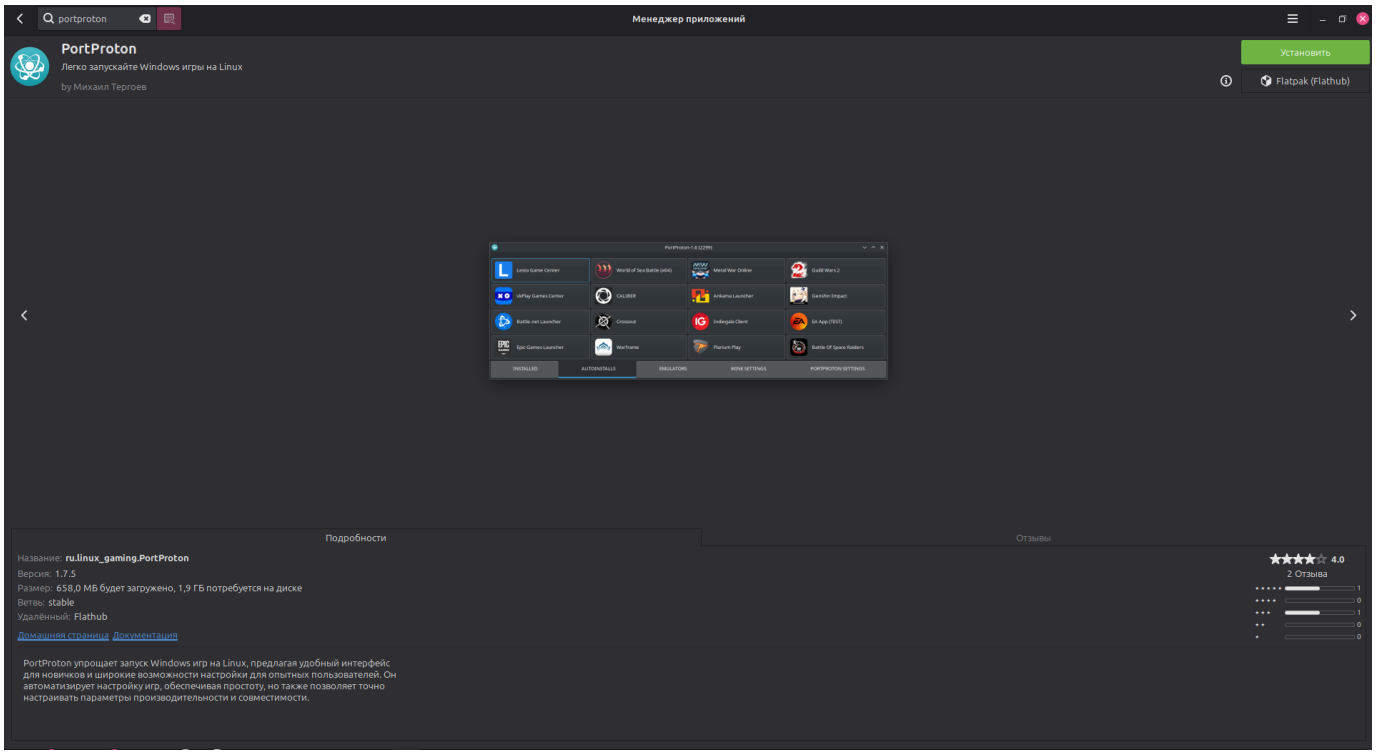
Запуск производиться командой:

Bash

```
flatpak run ru.linux_gaming.PortProton
```

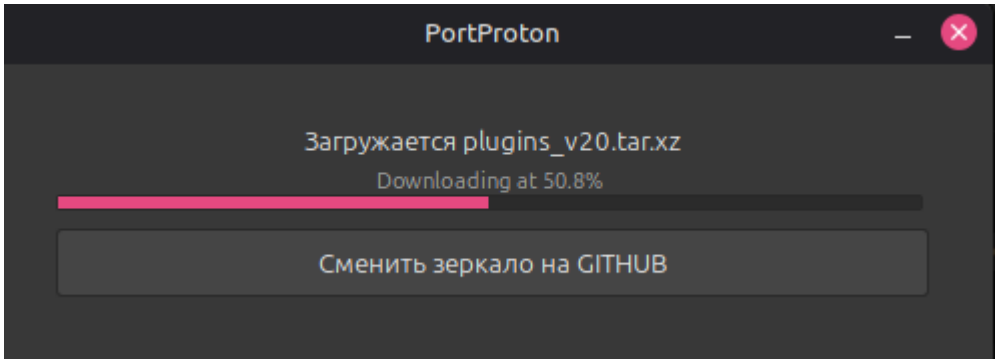
Через центр приложений:

Также PortProton доступен в Центре приложений GNOME после подключения Flathub.



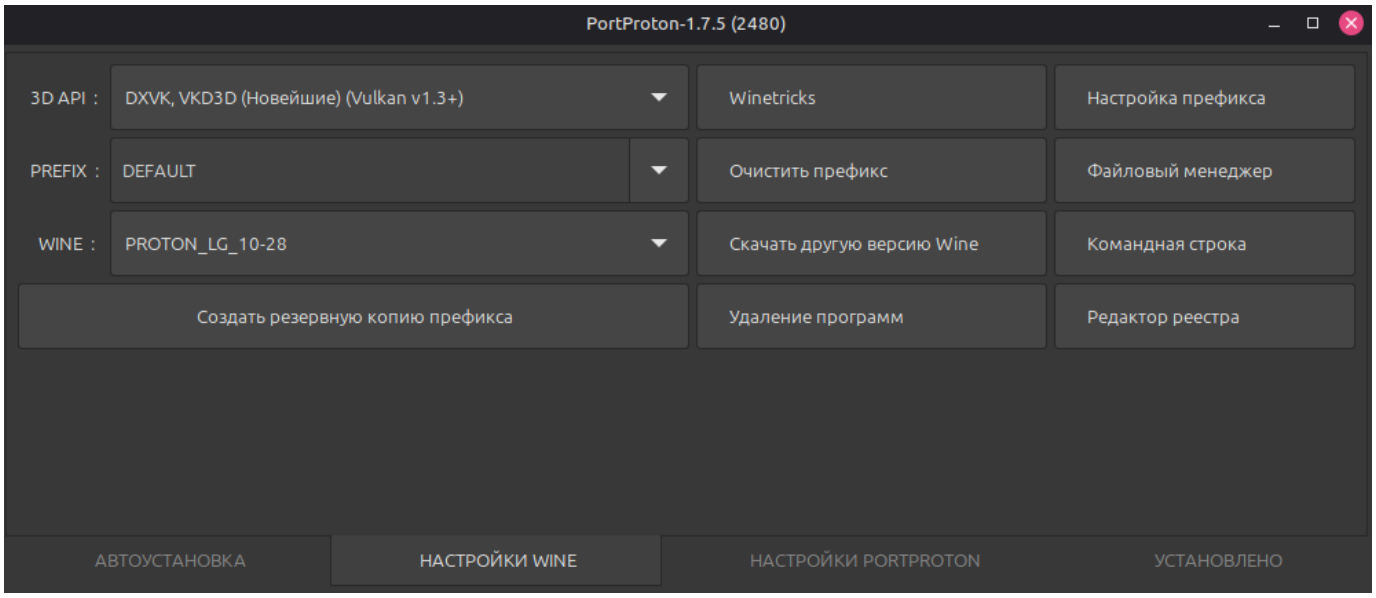
ПЕРВОНАЧАЛЬНАЯ НАСТРОЙКА

При первом запуске PortProton автоматически устанавливает необходимые зависимости Wine и вспомогательные компоненты. Процесс занимает несколько минут.



После завершения инициализации станут доступны основные функции приложения, в том числе:

- **Настройки Wine** — управление конфигурацией Wine-окружения;
- **Командная строка Windows** — запуск cmd.exe внутри Wine;
- **Файловый менеджер** — доступ к виртуальной файловой системе Windows.



Следующий шаг

После установки PortProton перейдите к разделу Установка SunriseWorkbench.

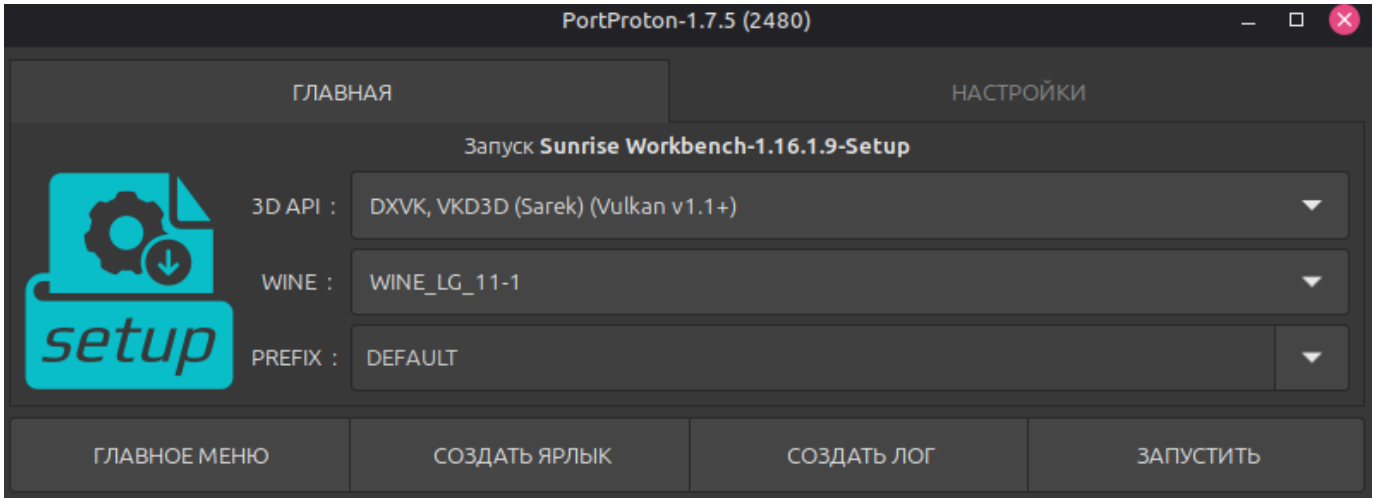
Установка SunriseWorkbench

В данном разделе описана установка SunriseWorkbench на Linux с использованием эмулятора PortProton.

> **Предварительное требование**
>
> Перед началом установки убедитесь, что PortProton установлен и настроен. Инструкции приведены в разделе Установка эмулятора Windows.

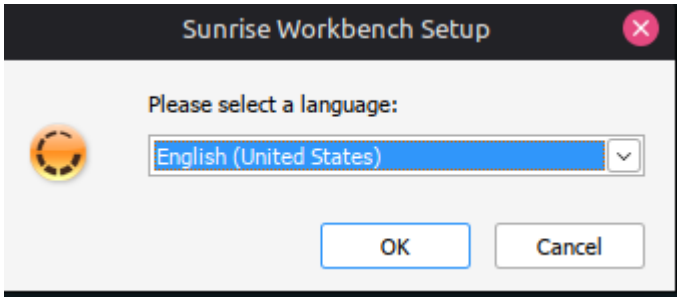
ЗАПУСК УСТАНОВЩИКА

Шаг 1. Перейдите в папку, содержащую установщик SunriseWorkbench. Нажмите правой кнопкой мыши на файл `.exe` и выберите **Открыть с помощью → PortProton**. В диалоговом окне оставьте настройки по умолчанию и нажмите **Запустить**.

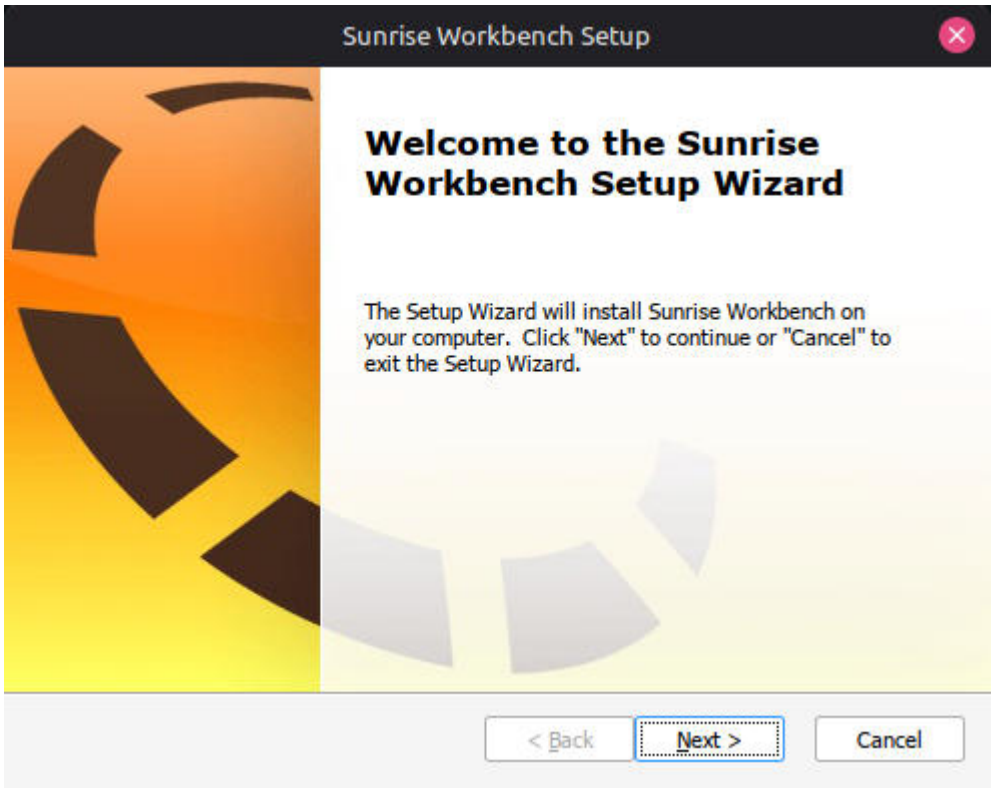


ПРОЦЕСС УСТАНОВКИ

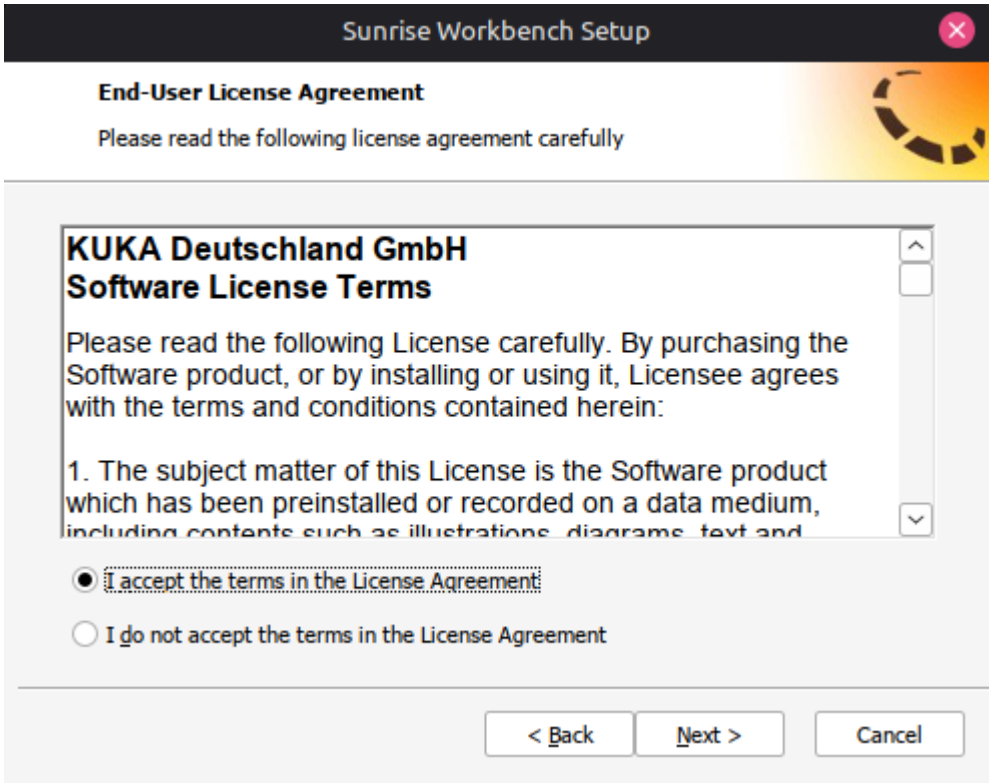
Шаг 2. В появившемся окне выберите язык установки (по умолчанию — английский) и нажмите **OK**.



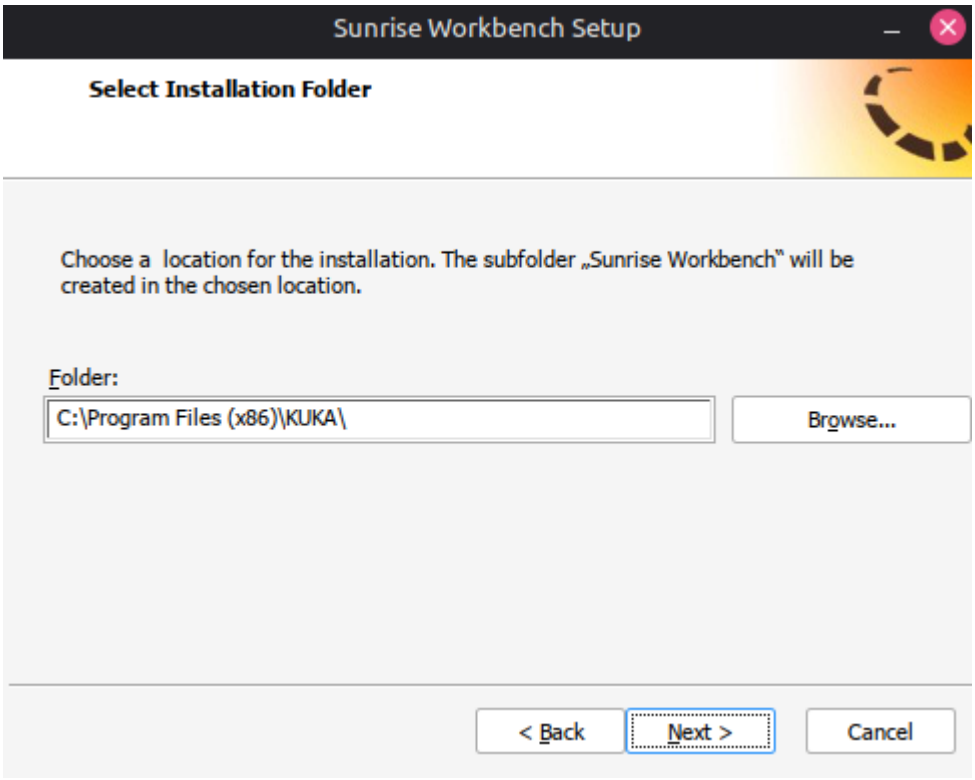
Шаг 3. В приветственном окне мастера установки нажмите **Next**.



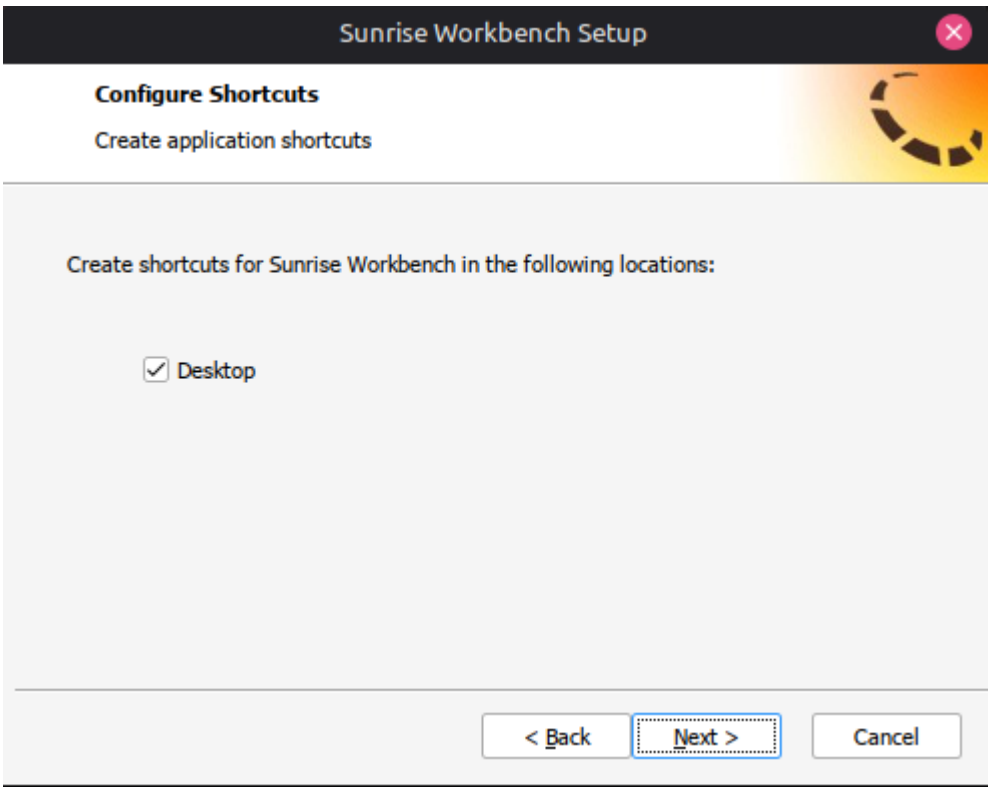
Шаг 4. Ознакомьтесь с условиями лицензионного соглашения и подтвердите принятие.



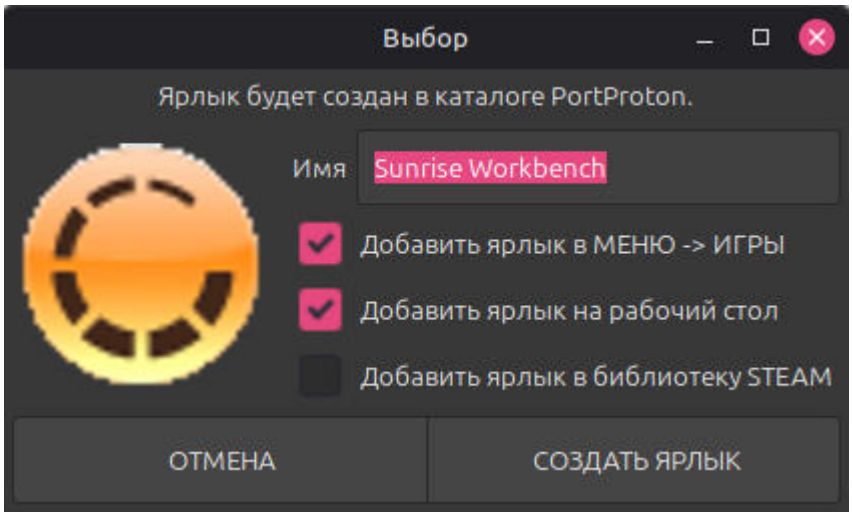
Шаг 5. Путь установки рекомендуется оставить по умолчанию.



Шаг 6. Установите флажок **Desktop** – ярлык SunriseWorkbench появится на рабочем столе.



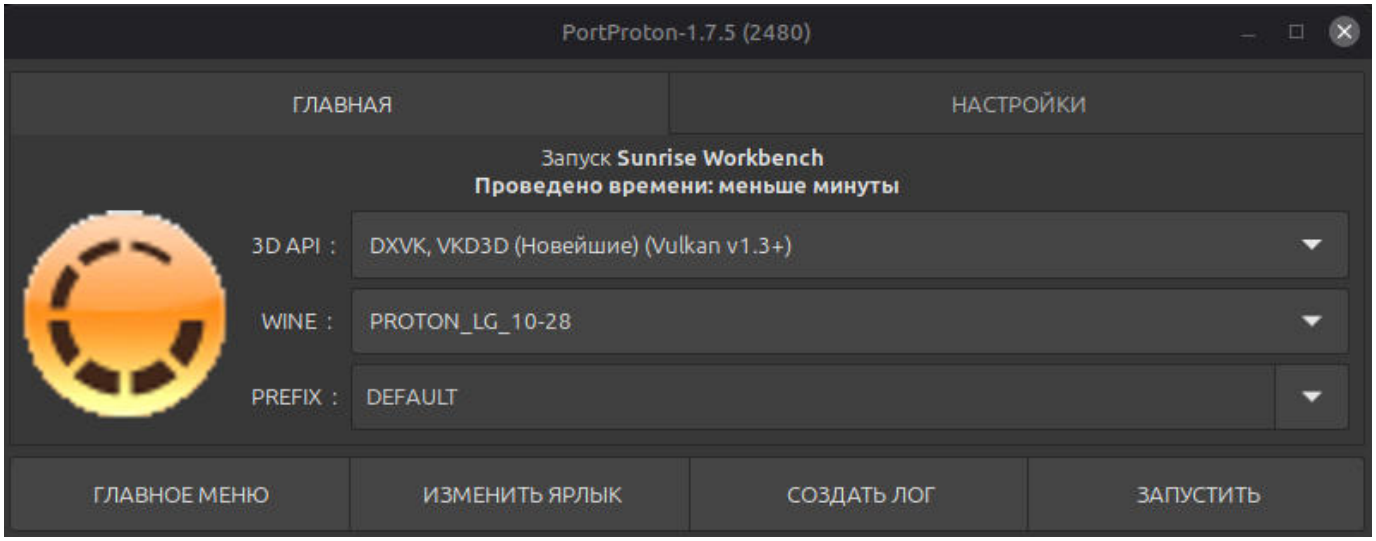
Шаг 7. Нажмите **install** и дождитесь завершения установки всех необходимых компонентов.



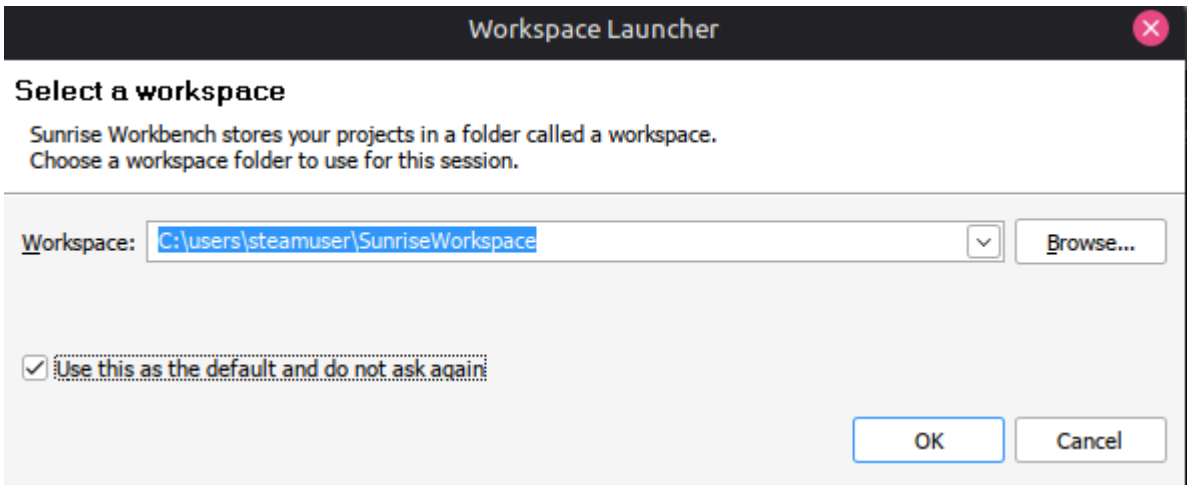
Нажмите на кнопку **Создать ярлык** чтобы он появился на рабочем столе.

ПЕРВЫЙ ЗАПУСК

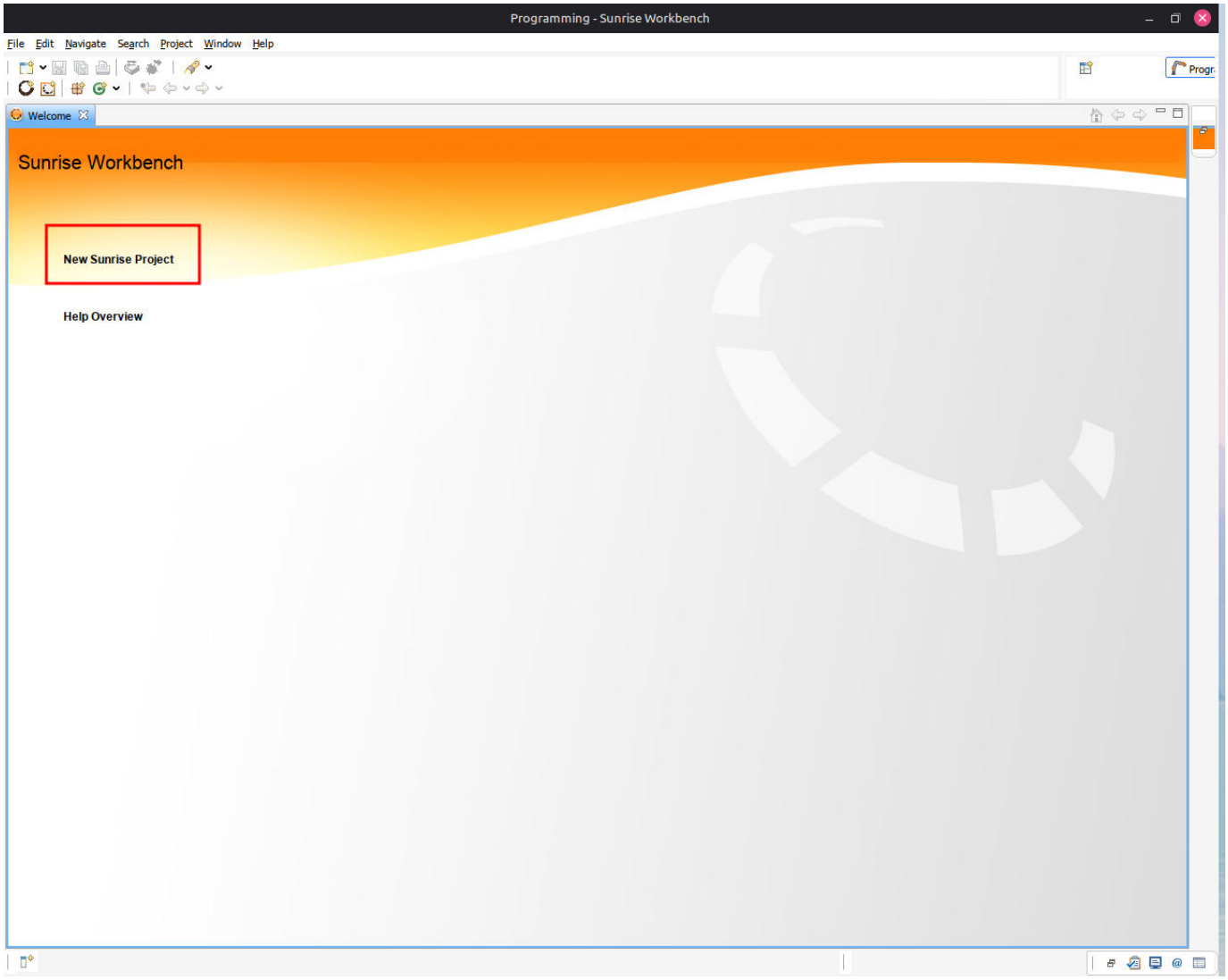
Шаг 8. После завершения установки запустите SunriseWorkbench через созданный ярлык на рабочем столе. Необходимо будет нажать на кнопку **Запустить** для того чтобы PortProton инициализировал окружение для работы приложения.



Шаг 9. Приложение предложит указать путь к рабочему пространству. Рекомендуется оставить значение по умолчанию и установить флажок **Use this as the default and do not ask again**.



Шаг 10. После загрузки главного окна нажмите **New Sunrise Project**.



Дальнейшая настройка

Подробные инструкции по конфигурации проекта представлены в разделе Создание нового проекта. Для полноценной работы также рекомендуется выполнить Установку библиотек.

2.3 Конфигурация проекта

2.3.1 Создание нового проекта

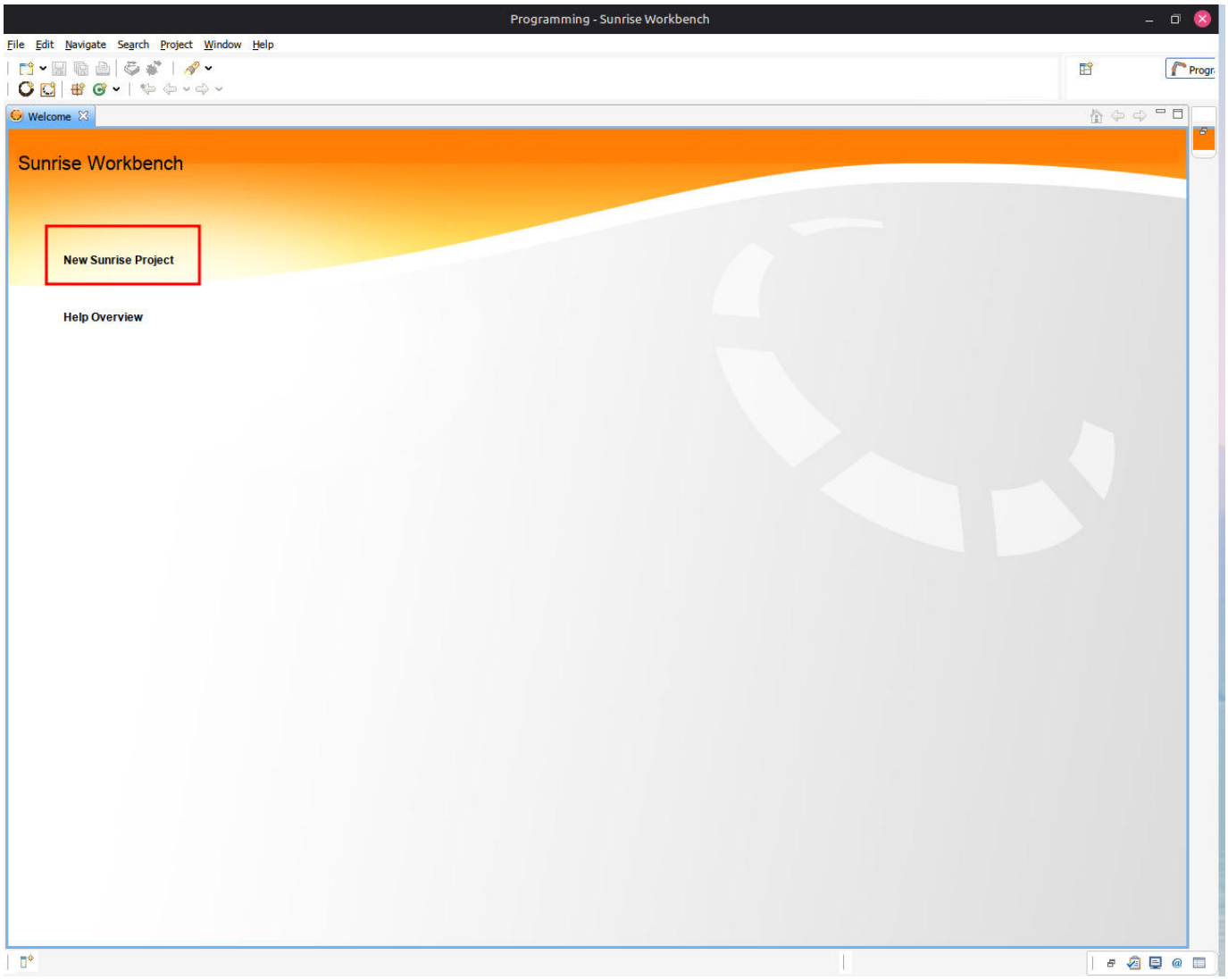
В данном разделе описан процесс создания нового проекта в SunriseWorkbench и его первоначальная конфигурация для работы с роботом KUKA LBR IIWA 7.

Предварительное требование

Перед созданием проекта убедитесь, что SunriseWorkbench установлен. Инструкции по установке приведены в разделах Windows и Linux.

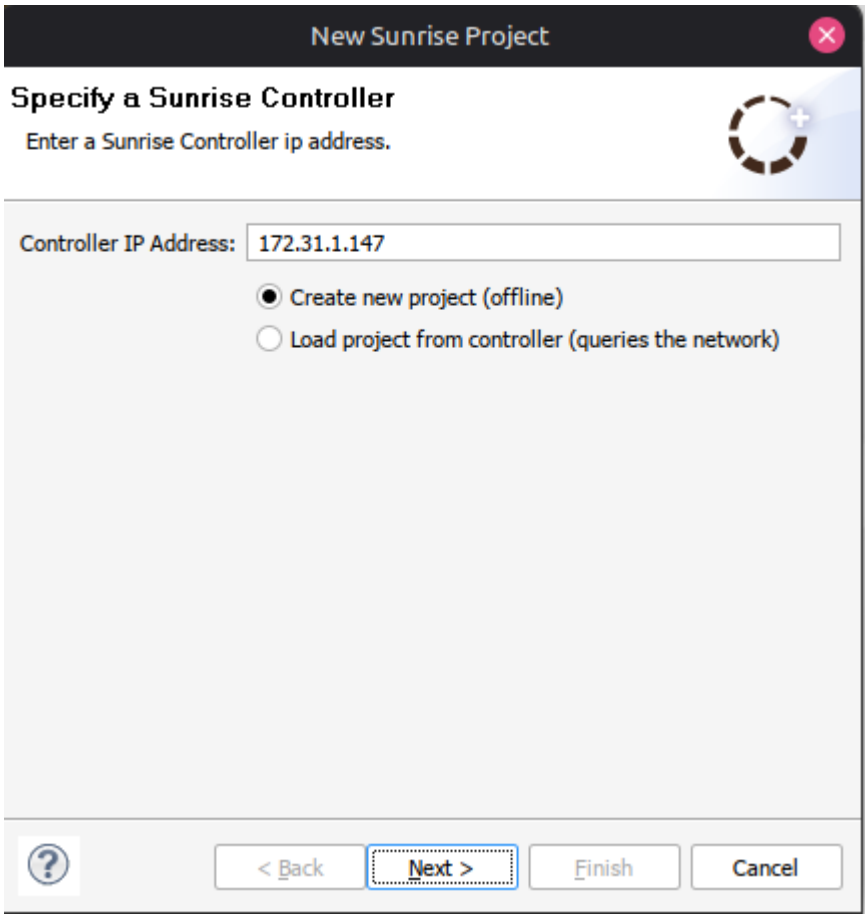
Запуск мастера создания проекта

В главном окне SunriseWorkbench нажмите кнопку **New Sunrise Project**.



Настройка подключения к контроллеру

В появившемся диалоговом окне укажите IP-адрес контроллера KUKA Sunrise Cabinet.

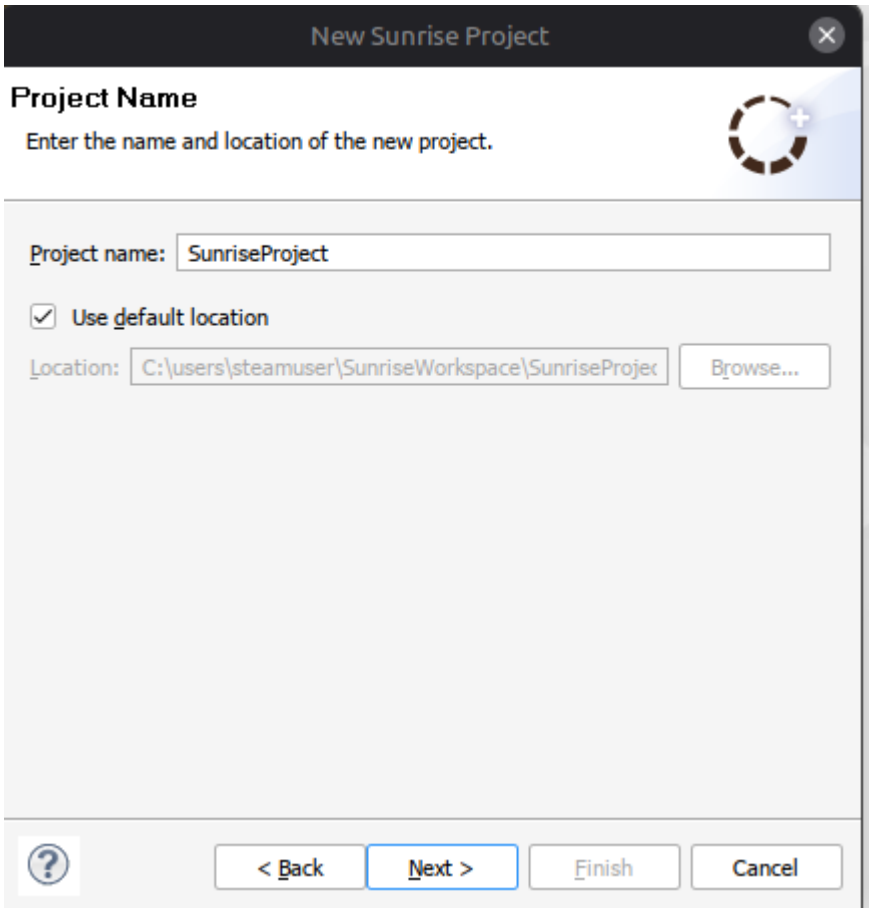


IP-адрес контроллера

IP-адрес контроллера по умолчанию: `172.31.1.147`. Если в вашей конфигурации используется другой адрес, обязательно измените его на актуальный.

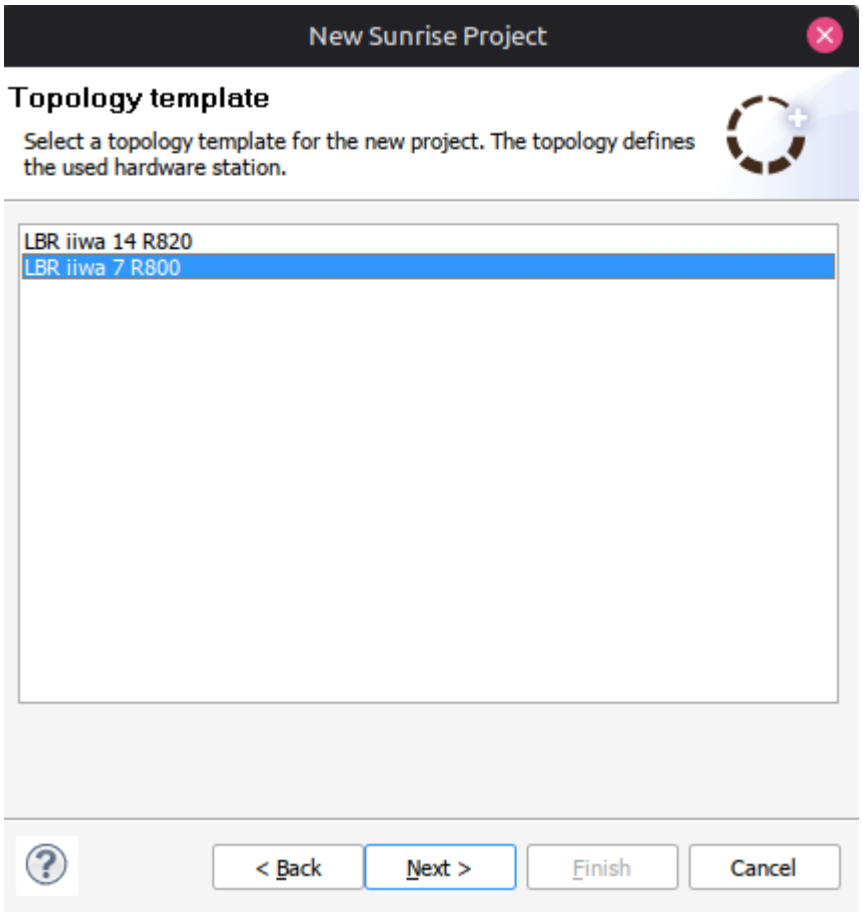
Название проекта

Задайте наименование проекта в соответствующем поле.



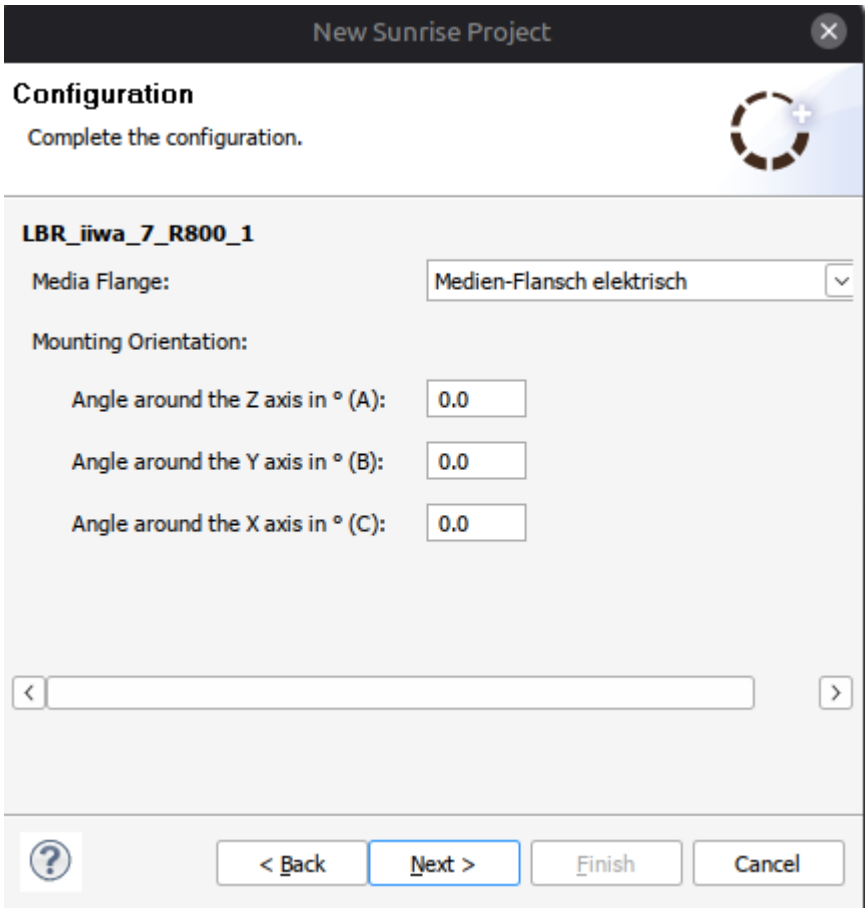
Выбор модели робота

В выпадающем списке выберите модель робота. Для KUKA LBR IIWA 7 выберите **LBR iiwa 7 R800**.



Выбор фланца

Выберите тип фланца в соответствии с конфигурацией вашего робота.

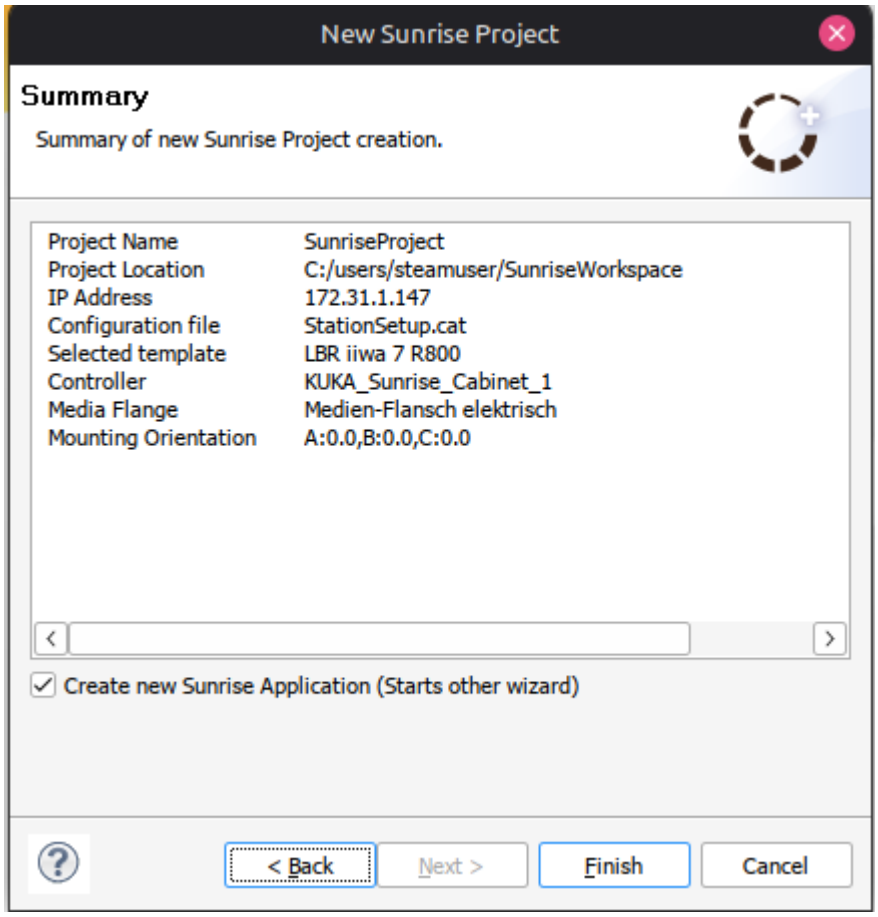


Выбор фланца

Тип фланца должен точно соответствовать физической конфигурации робота. В данной конфигурации используется **Medien-Flansch elektrisch**. Ориентацию оставьте по умолчанию (0°).

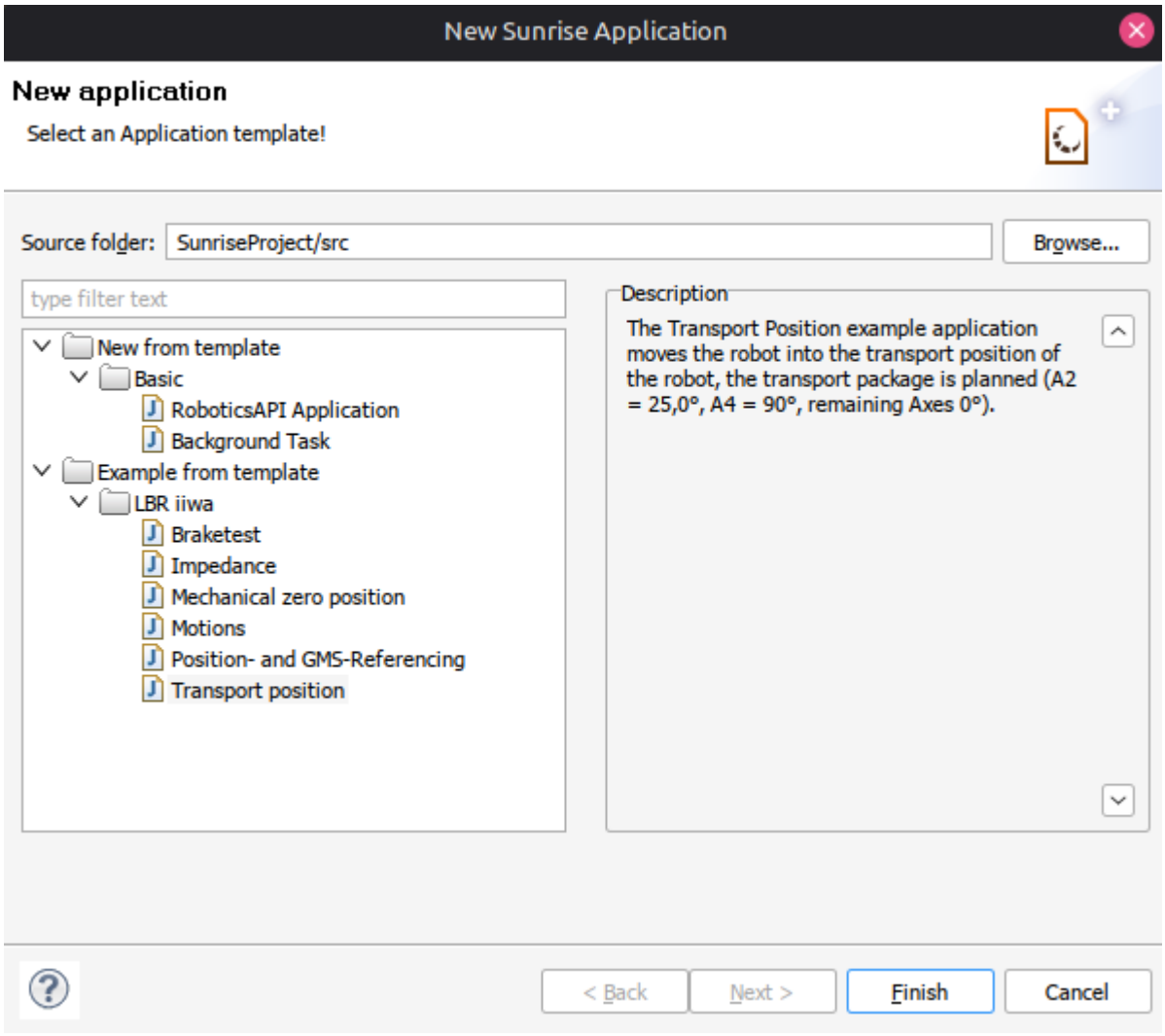
Проверка конфигурации

В итоговом окне проверьте все указанные параметры. Убедившись в их корректности, нажмите **Finish**.



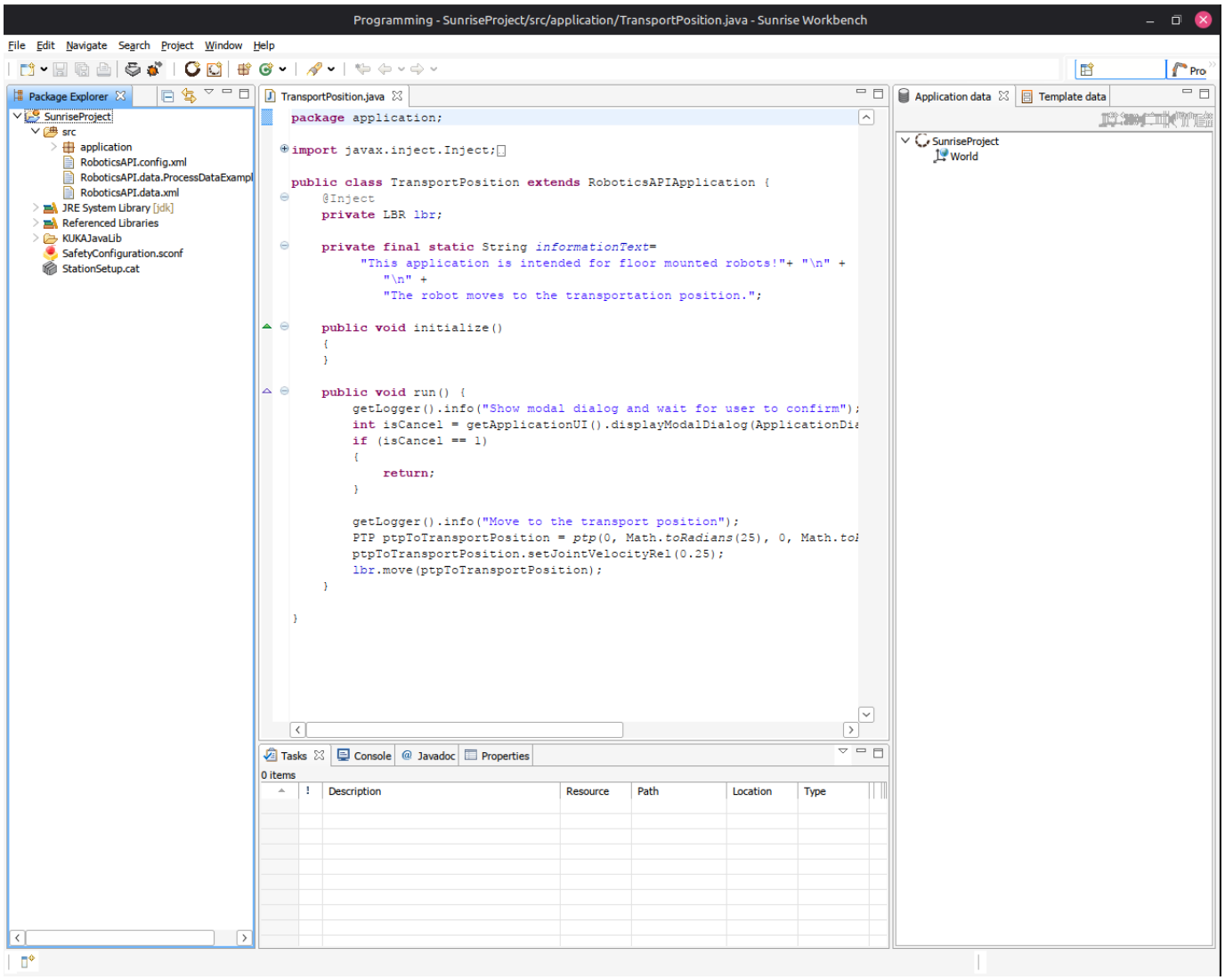
Выбор шаблона приложения

После создания проекта откроется диалог выбора шаблона. Выберите один из предложенных примеров и нажмите **Finish**.



Главное окно редактора

После успешного завершения мастера откроется основное окно редактора SunriseWorkbench с созданным проектом.



Следующий шаг

Для полноценной работы с роботом необходимо установить все требуемые библиотеки. Инструкции приведены в разделе Установка библиотек.

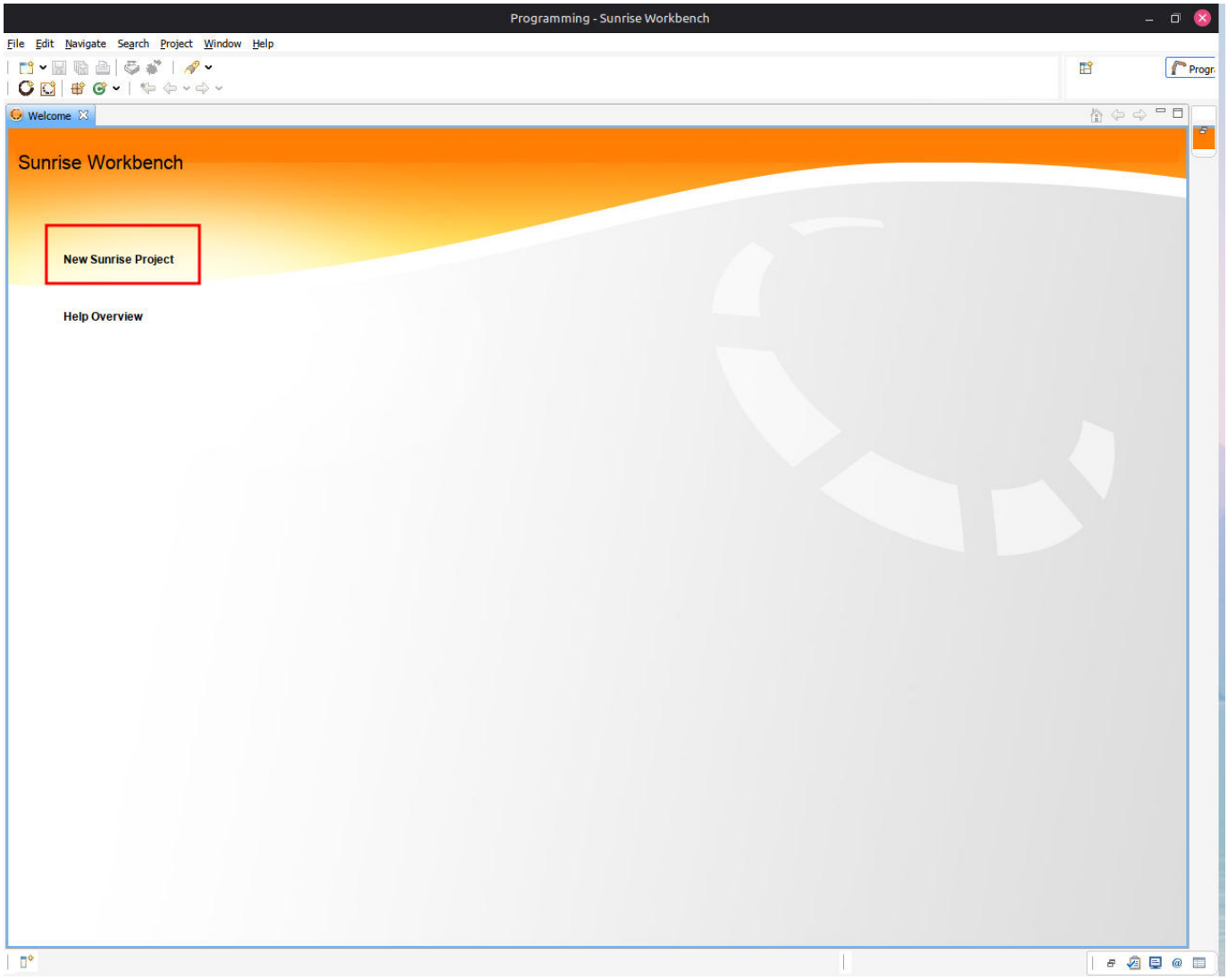
2.3.2 Загрузка проекта с контроллера

В данном разделе описана процедура импорта существующего проекта непосредственно с контроллера KUKA в среду SunriseWorkbench.

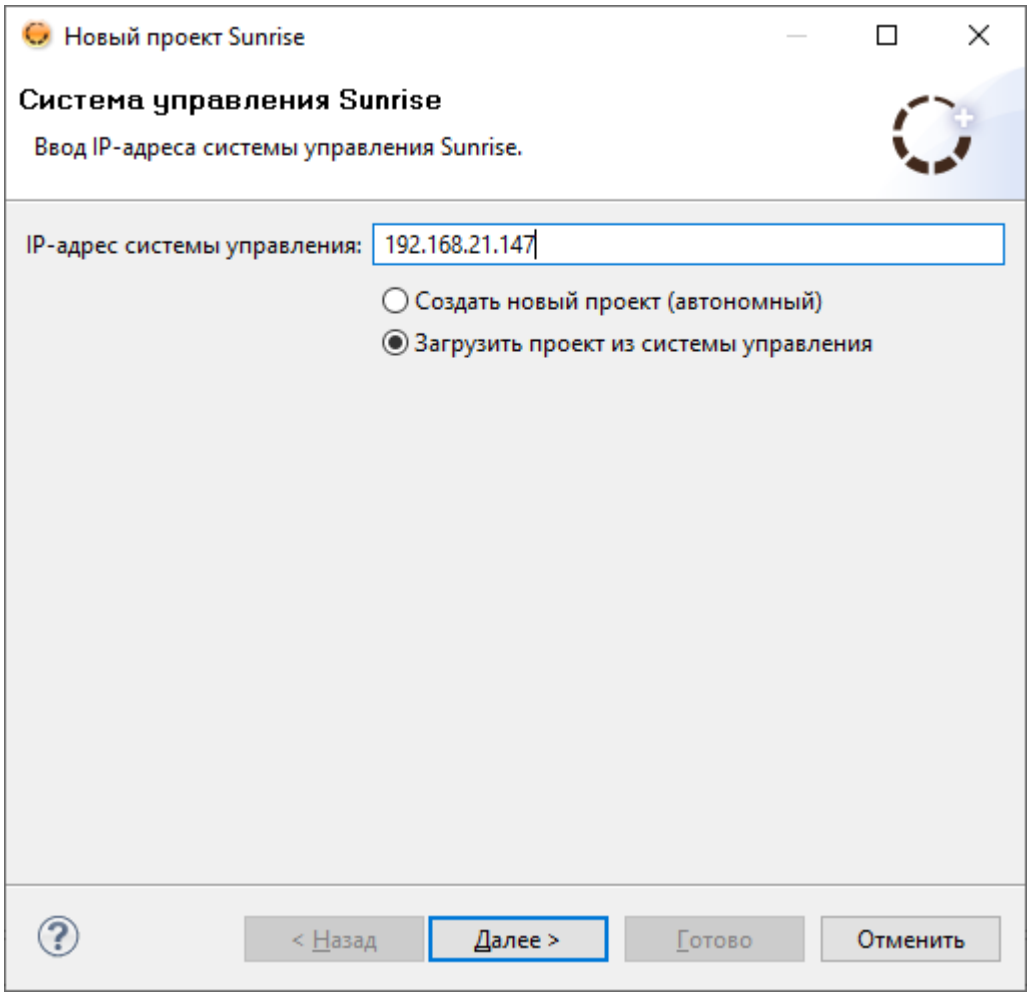
> **Предварительное требование**
>
> Убедитесь, что SunriseWorkbench установлен и запущен. Инструкции по установке приведены в разделах Windows и Linux.

Запуск мастера импорта проекта

В главном окне SunriseWorkbench нажмите кнопку **New Sunrise Project**.



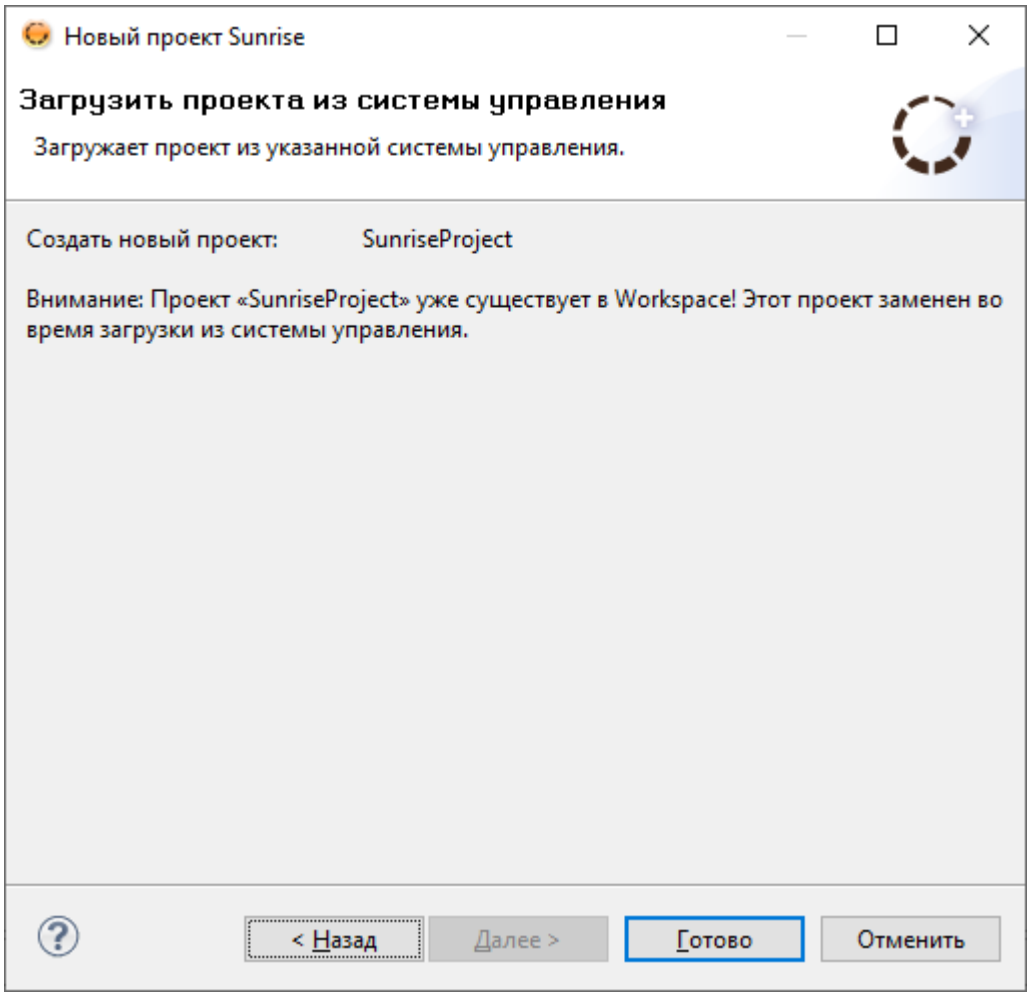
В открывшемся диалоговом окне выберите опцию **Load project from controller** и введите IP-адрес контроллера KUKA Sunrise Cabinet.



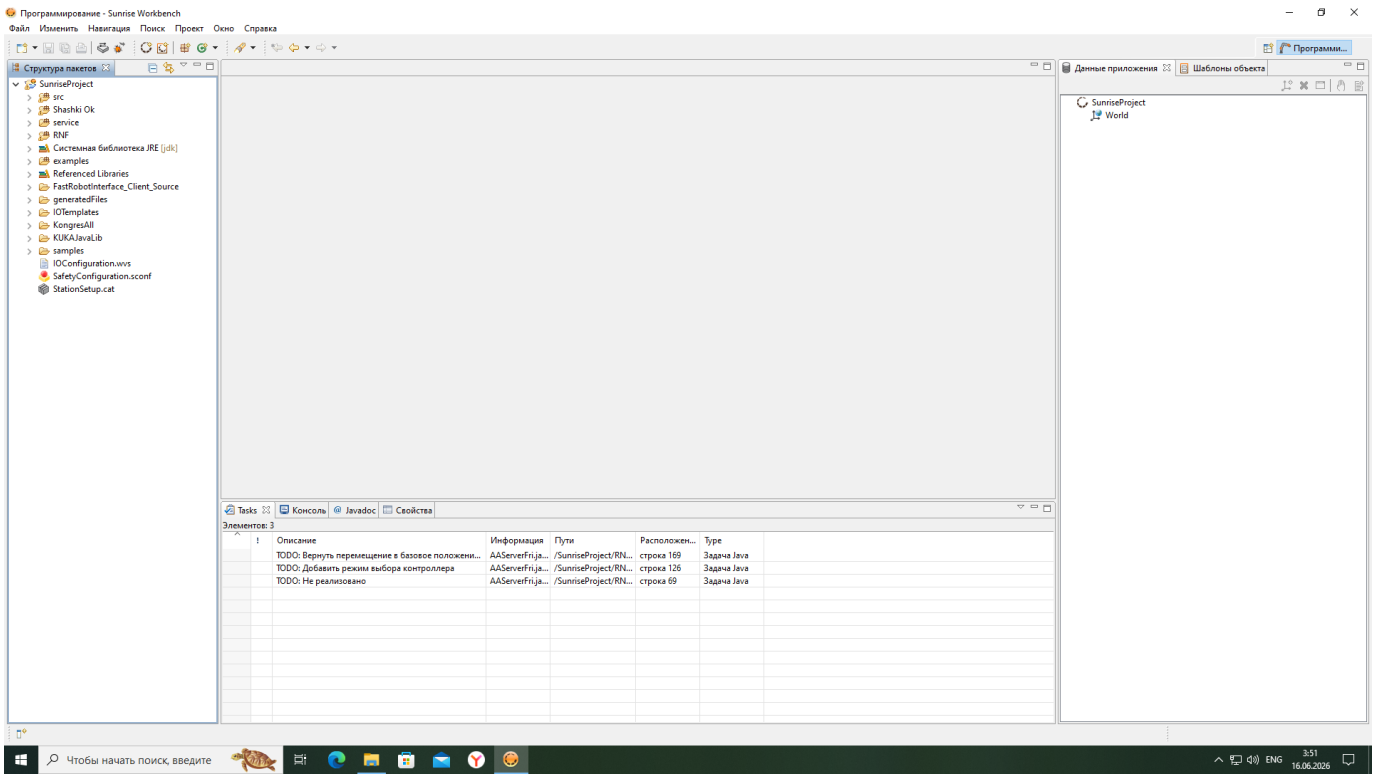
IP-адрес контроллера

IP-адрес контроллера по умолчанию: `172.31.1.147`. Если контроллер был предварительно перенастроен, необходимо указать актуальный IP-адрес. В данной конфигурации используется адрес `192.168.21.147`. Для получения текущего IP-адреса обратитесь к разделу Настройка станции.

Нажмите **Next** — начнётся загрузка проекта с контроллера.



После завершения загрузки импортированный проект отобразится в дереве проектов SunriseWorkbench. В данном примере проект называется `SunriseProject`.



Следующий шаг

Для полноценной работы с роботом необходимо установить все требуемые библиотеки. Инструкции приведены в разделе Установка библиотек.

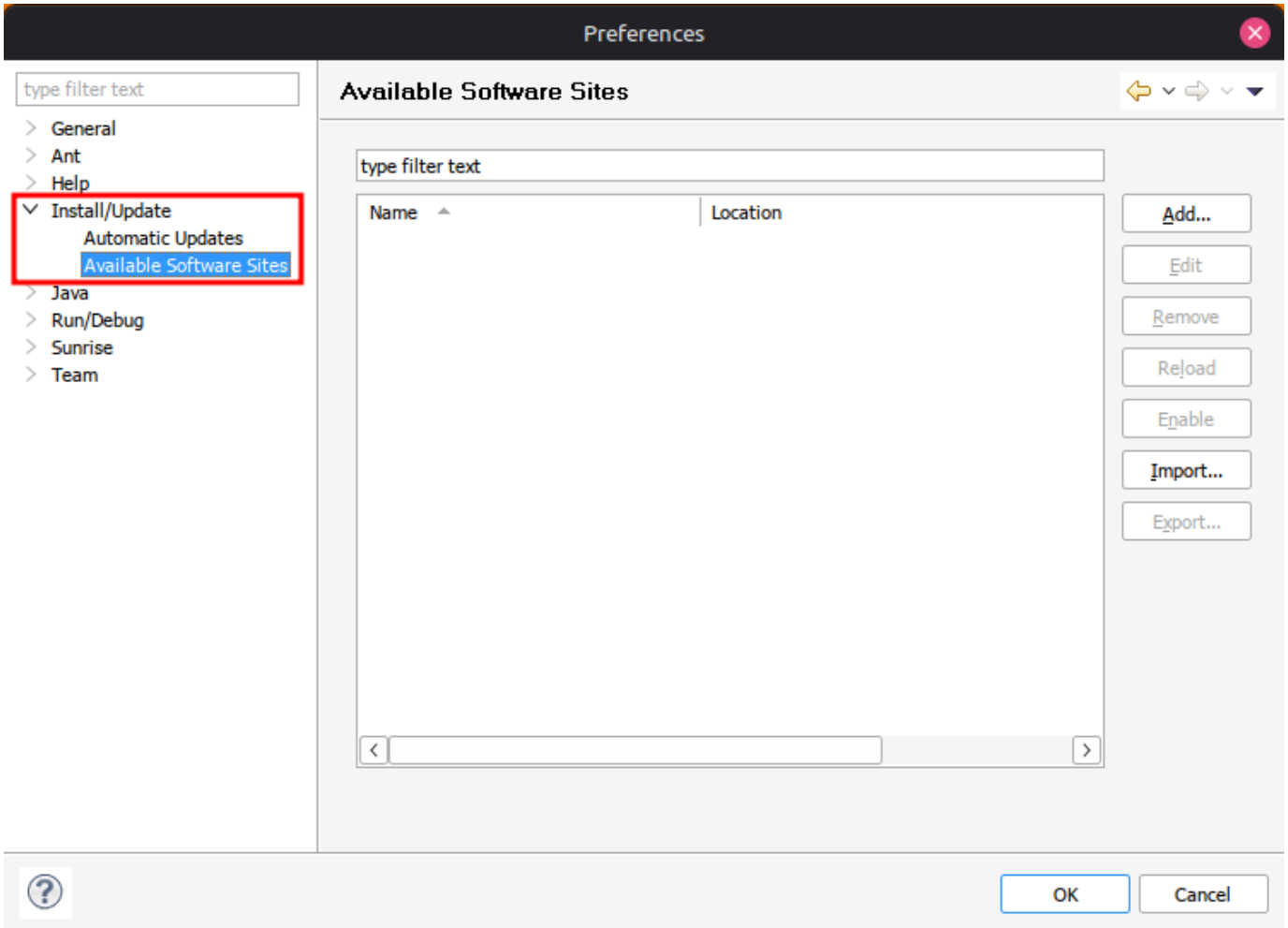
2.3.3 Установка библиотек

Для обеспечения полноценной работы проекта SunriseWorkbench необходимо установить пакет дополнительных библиотек, поставляемых в виде `.zip`-архивов.

> **Предварительное требование**
>
> Перед установкой библиотек убедитесь, что проект создан или загружен. Обратитесь к разделу Создание нового проекта или Загрузка готового проекта.

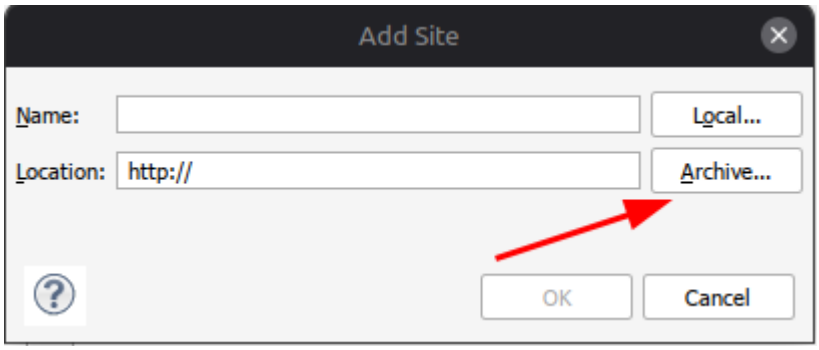
Открытие настроек

В строке меню главного окна SunriseWorkbench выберите **Window → Preferences**. В открывшемся окне перейдите в раздел **Install/Update → Available Software Sites**.



Добавление архивов библиотек

Нажмите кнопку **Add**. В появившемся диалоговом окне добавьте `.zip`-архив с библиотеками через кнопку **Archive...**.



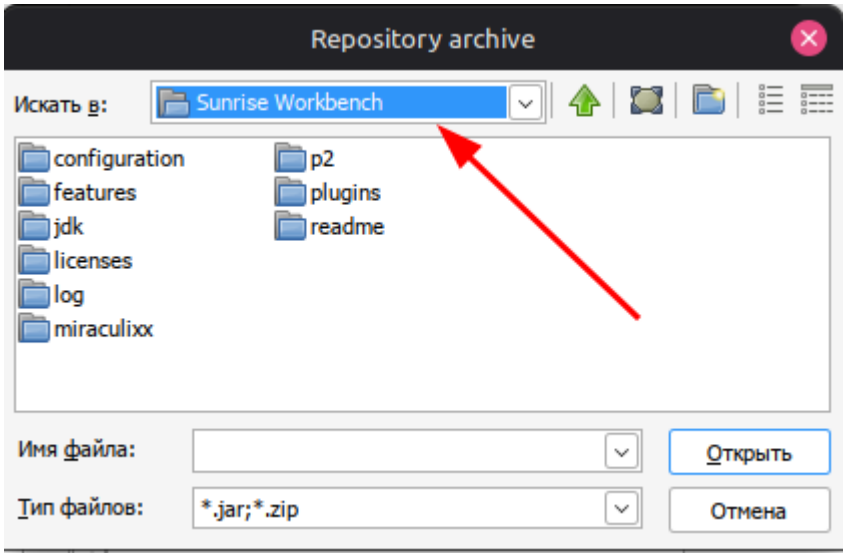
ДЛЯ ПОЛЬЗОВАТЕЛЕЙ WINDOWS

Текст...

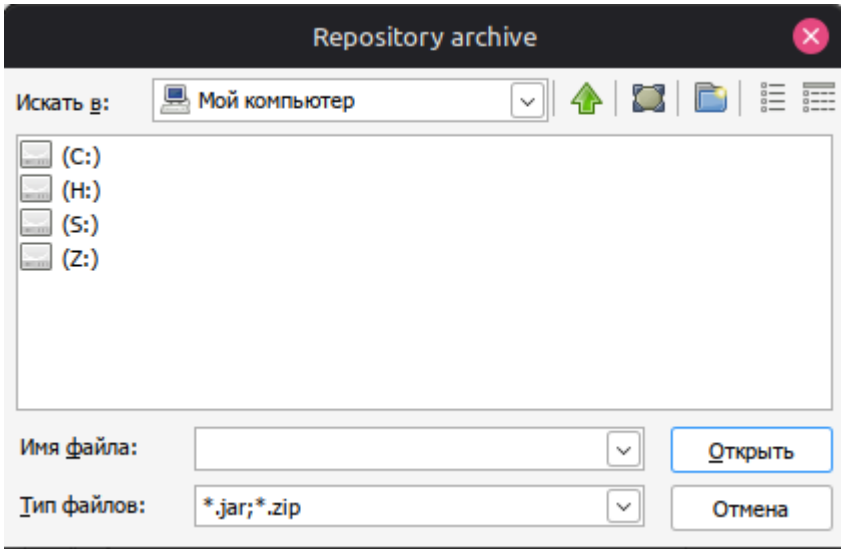
ДЛЯ ПОЛЬЗОВАТЕЛЕЙ LINUX

Поскольку SunriseWorkbench работает в эмулированной Windows-среде, для доступа к файловой системе Linux необходимо выполнить следующие действия:

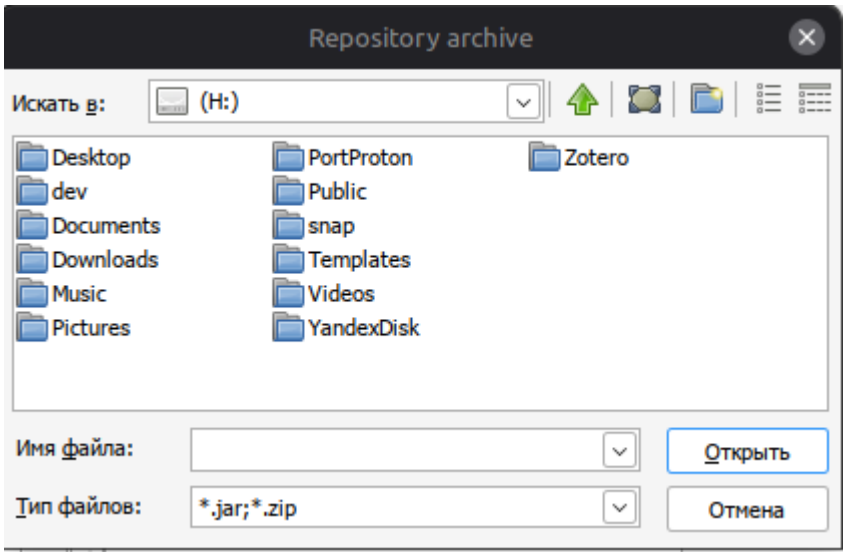
1. В диалоге выбора файла нажмите поле **Искать в:** и выберите **Мой компьютер**.



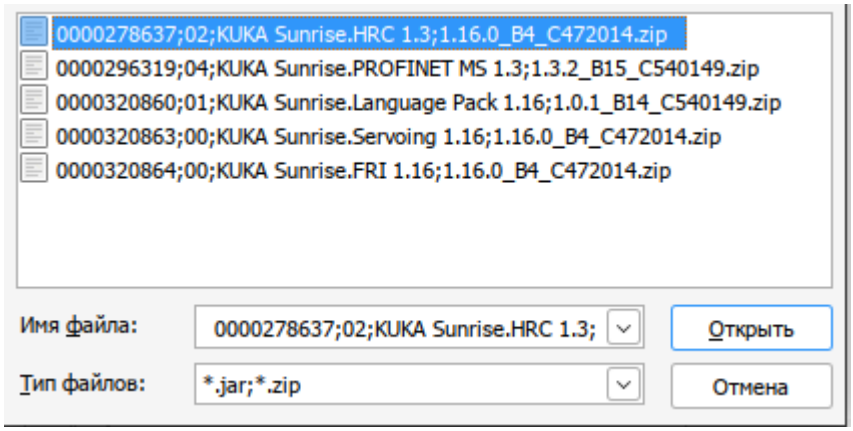
2. Отобразится список всех смонтированных дисков. Их количество может превышать фактическое число физических накопителей — это особенность работы эмулятора.



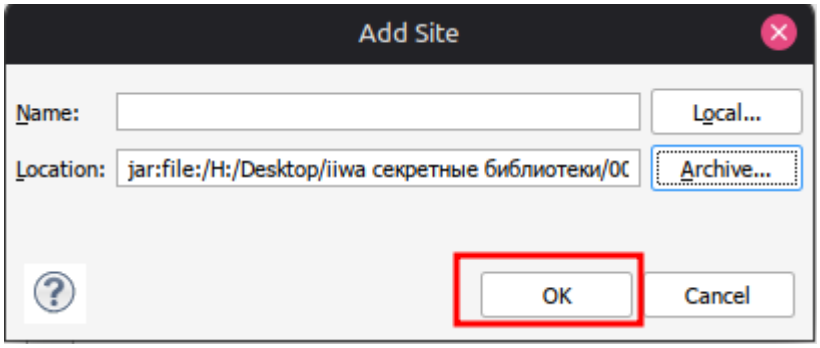
3. Последовательно проверьте каждый диск. В одном из них будет отображена файловая система Linux (в примере — диск H).



4. Перейдите в папку с библиотеками, выберите один из `.zip`-архивов и нажмите **OK**.

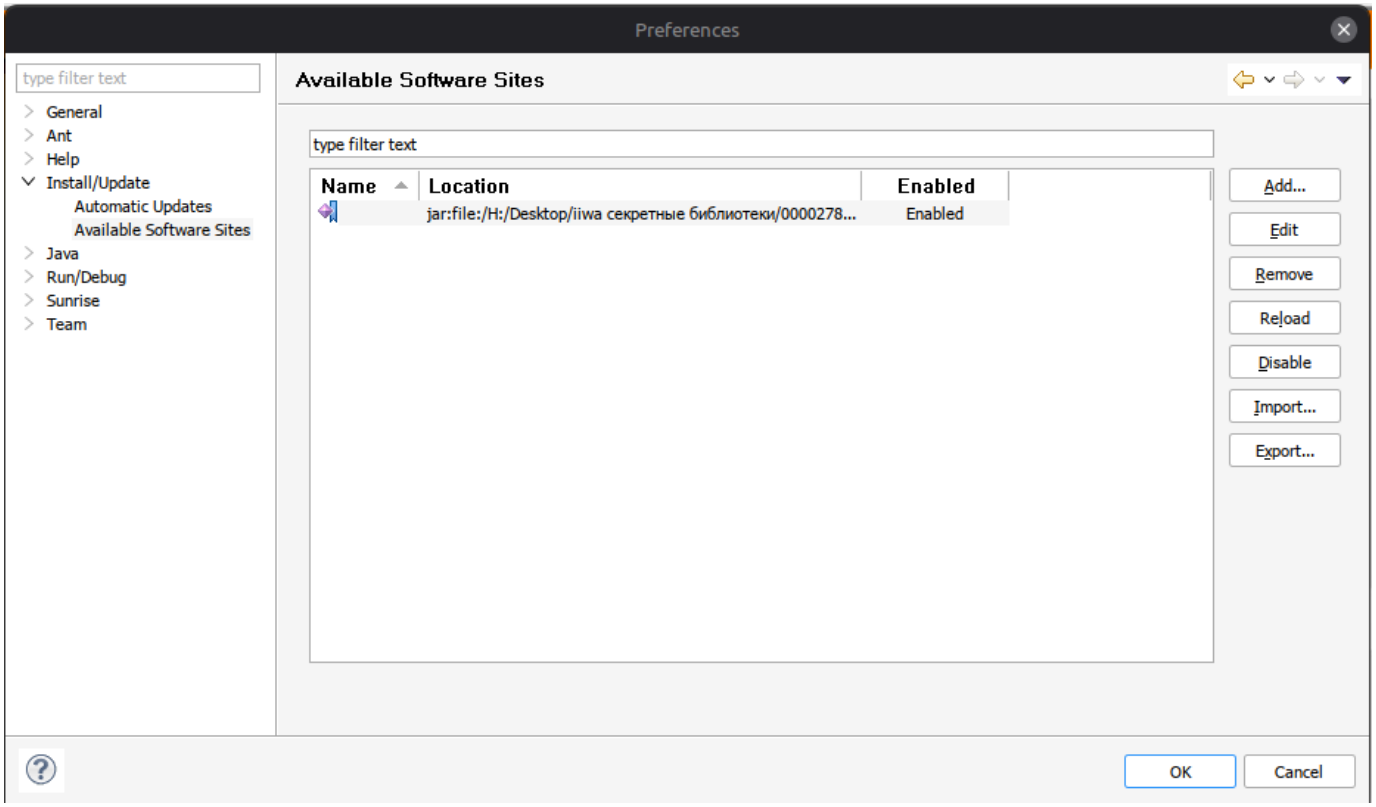


5. В открывшемся окне подтвердите выбор архива, нажав **OK**.



Добавление оставшихся архивов

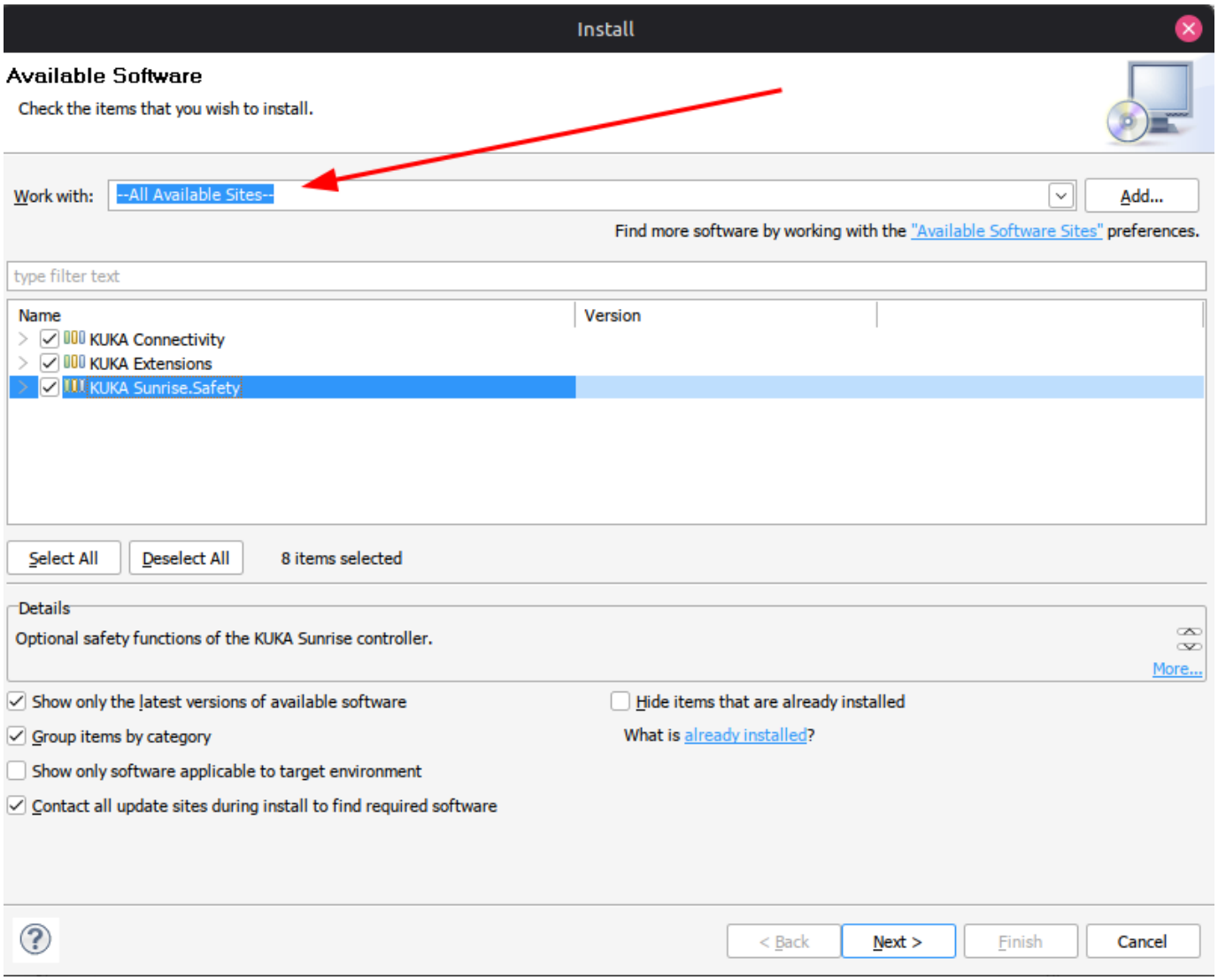
В списке **Available Software Sites** должна появиться запись с добавленным архивом. Повторите процедуру для всех оставшихся `.zip`-файлов библиотек.



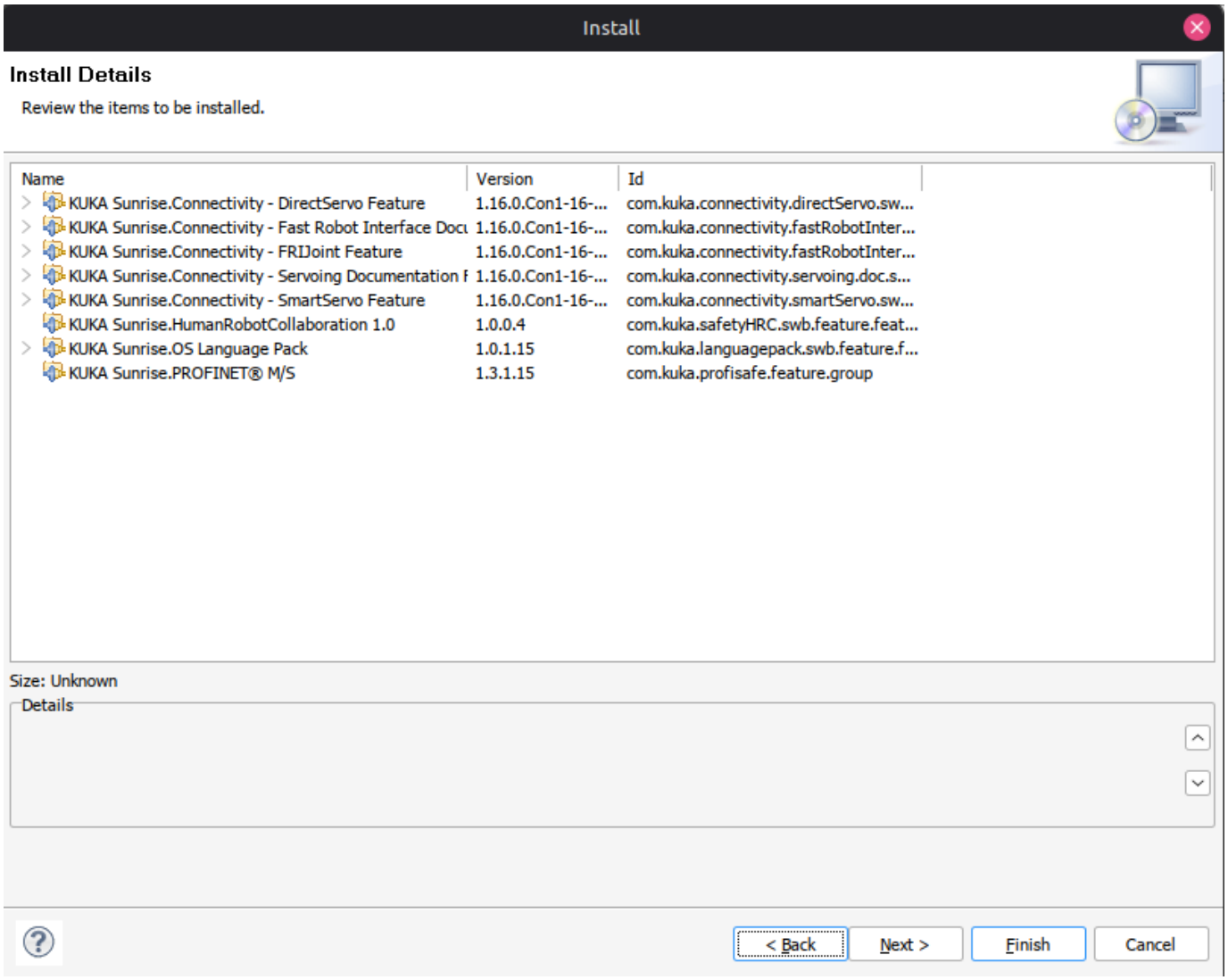
После добавления всех архивов нажмите **OK** для сохранения настроек.

Установка библиотек

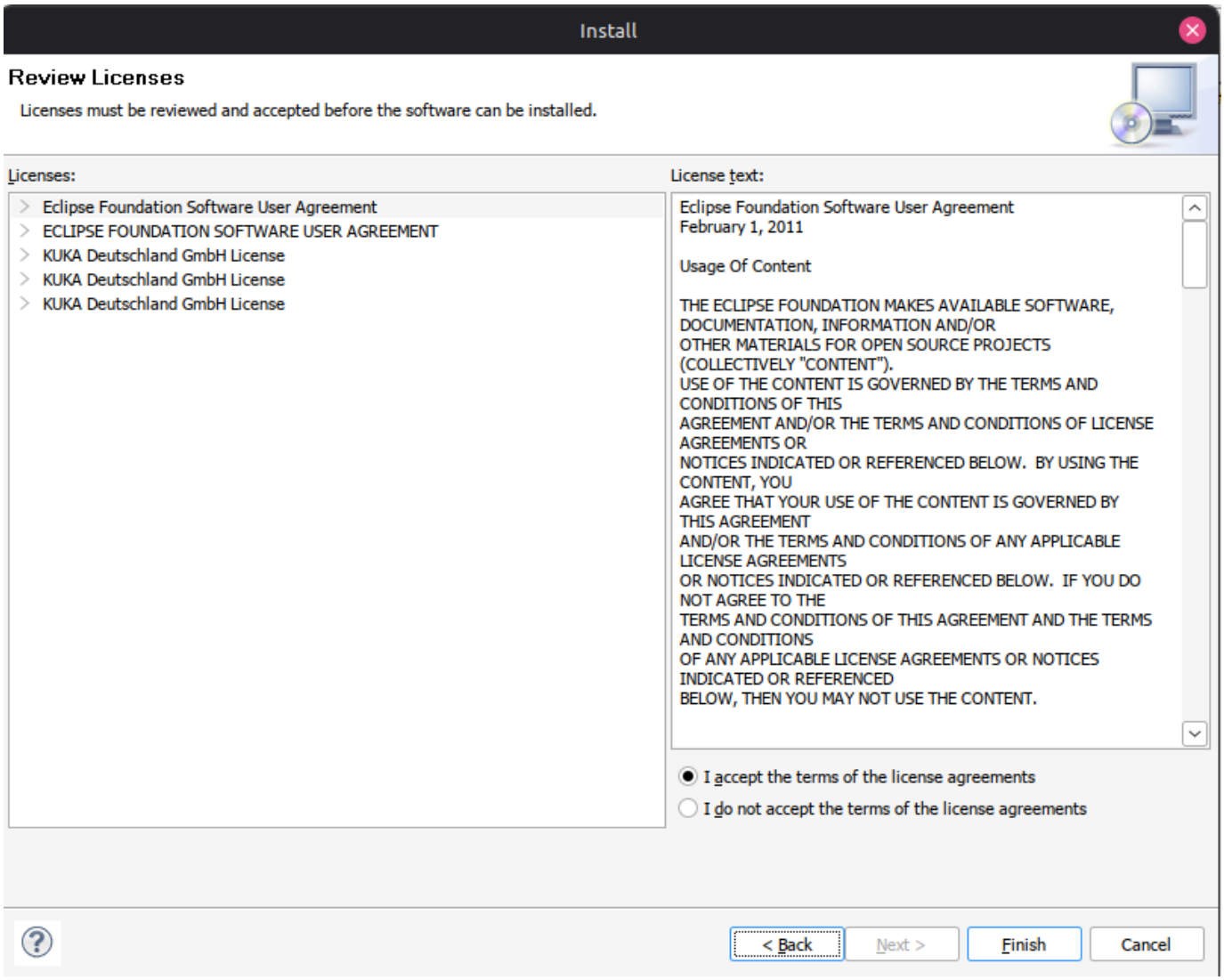
В строке меню выберите **Help → Install New Software...**. В открывшемся окне в поле **Work with** выберите **All Available Sites** — в списке отобразятся компоненты из всех добавленных архивов.



Установите флажки напротив всех доступных компонентов и нажмите **Next**. Ознакомьтесь со сводкой устанавливаемых компонентов и нажмите **Next**.

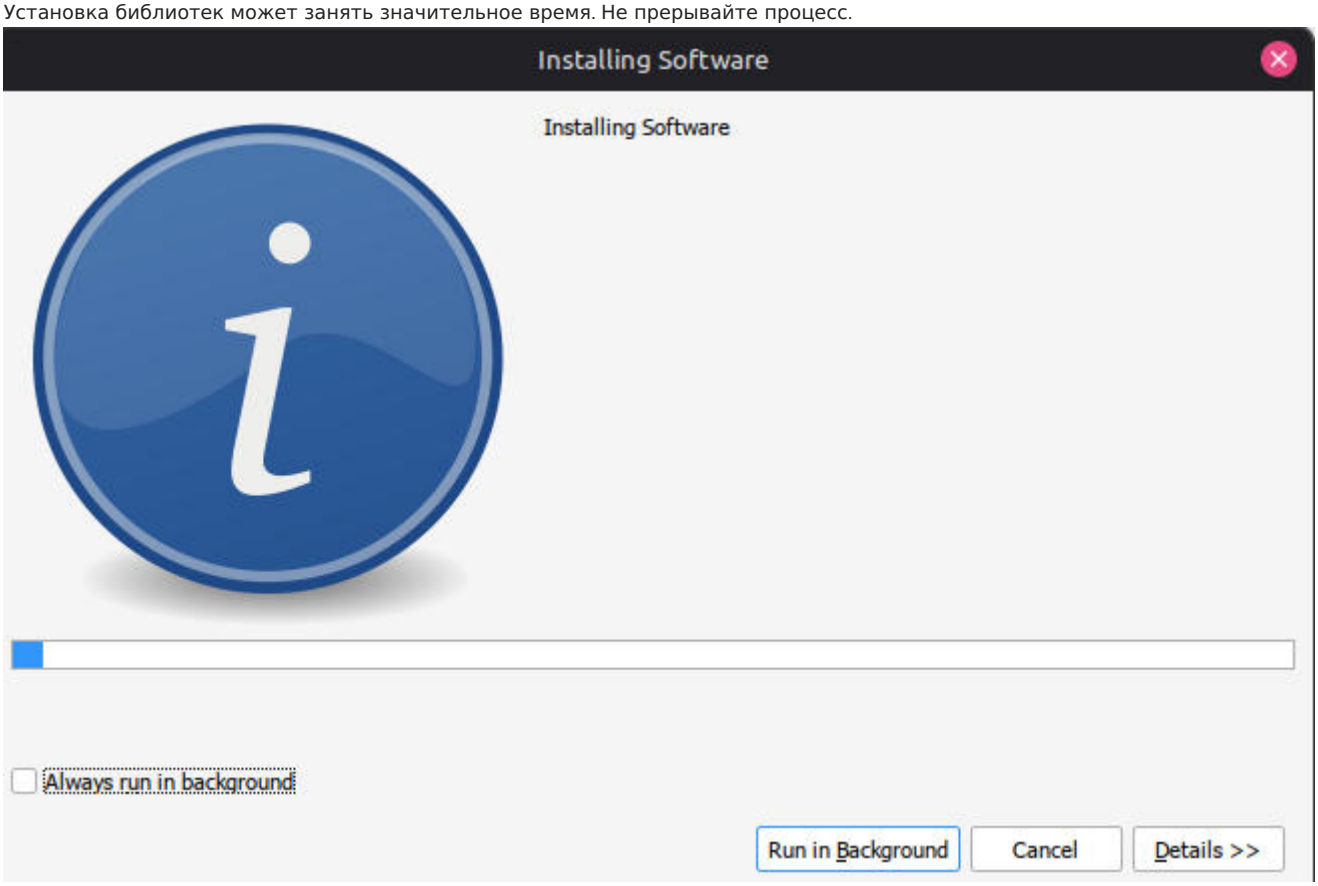


Примите условия лицензионных соглашений и нажмите **Finish** — начнётся процесс установки.


Install
Review Licenses
Licenses must be reviewed and accepted before the software can be installed.
Licenses:
Eclipse Foundation Software User Agreement
ECLIPSE FOUNDATION SOFTWARE USER AGREEMENT
KUKA Deutschland GmbH License
KUKA Deutschland GmbH License
KUKA Deutschland GmbH License
License text:
Eclipse Foundation Software User Agreement
February 1, 2011
Usage Of Content
THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR
OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT").
USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS
AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR
NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU
AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT
AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS
OR NOTICES INDICATED OR REFERENCED BELOW. IF YOU DO NOT AGREE TO THE
TERMS AND CONDITIONS OF THIS AGREEMENT AND THE TERMS AND CONDITIONS
OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED
BELOW, THEN YOU MAY NOT USE THE CONTENT.
I accept the terms of the license agreements
I do not accept the terms of the license agreements
< Back
Next >
Finish
Cancel

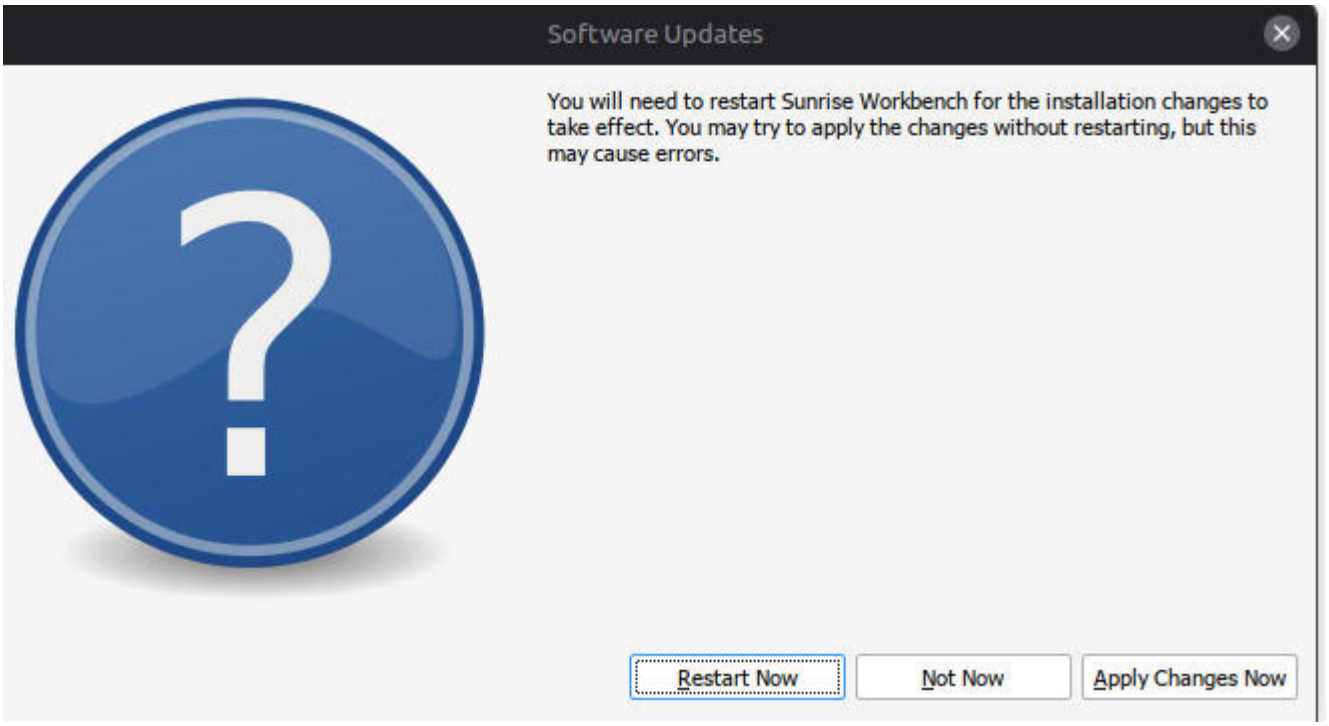
Продолжительность установки

Установка библиотек может занять значительное время. Не прерывайте процесс.

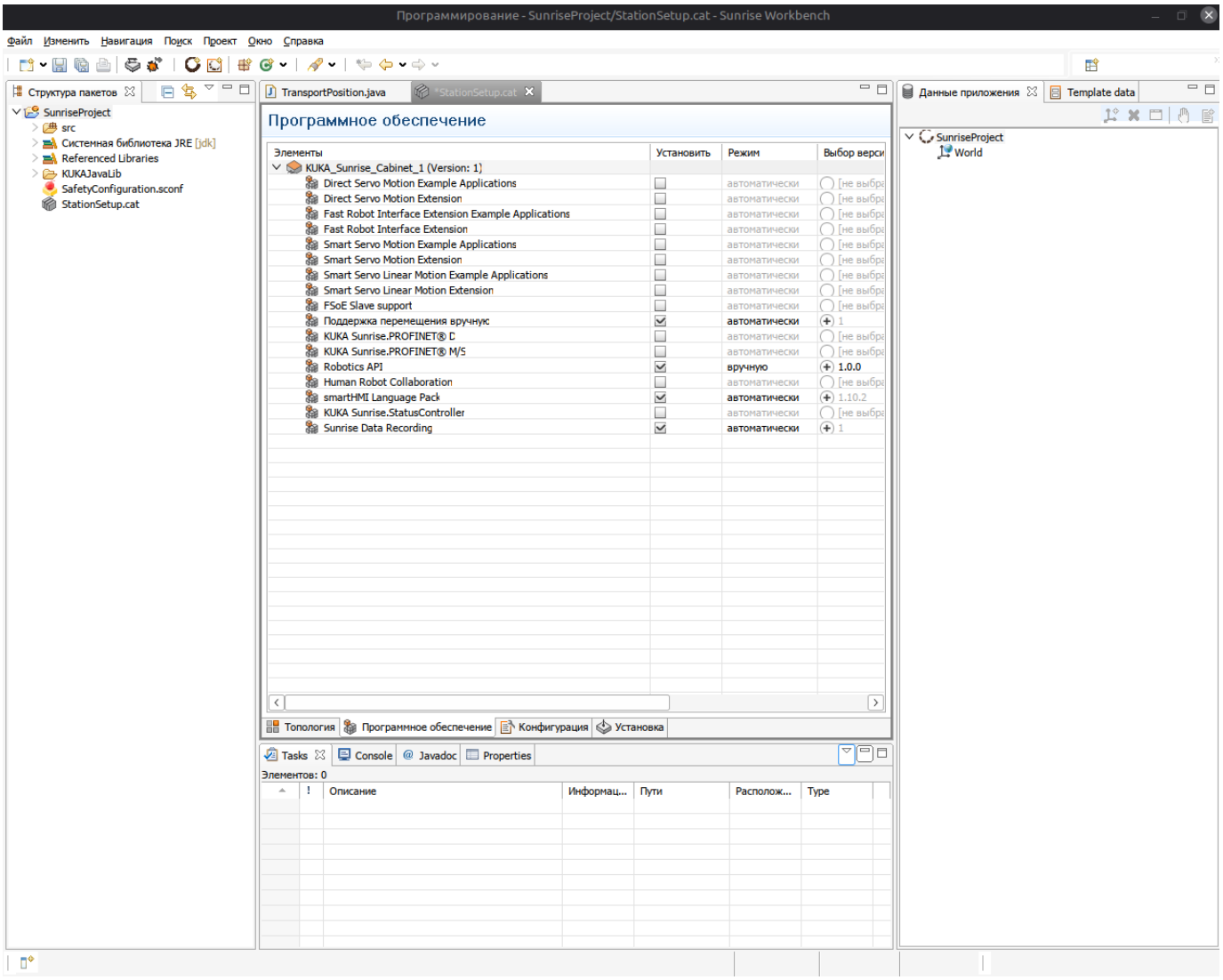


Перезапуск приложения

По завершении установки SunriseWorkbench предложит выполнить перезапуск. Нажмите **Restart Now**.



После перезапуска интерфейс переключится на русский язык, а в файле `StationSetup.cat` появятся все установленные библиотеки.



Следующий шаг

Если проект ещё не настроен, перейдите к разделу Создание нового проекта или Загрузка готового проекта.

2.4 KUKA LBR IIWA 7

2.4.1 Описание программ

TeachKuka

В данном разделе будет представлено описание управляющей программы **TeachKuka**, предназначенной для обучения позиций робота.

> **Раздел в разработке**
>
> Содержимое данного раздела находится в процессе подготовки.

LBRserver

В данном разделе будет представлено описание управляющей программы **LBRserver**, обеспечивающей связь контроллера с ROS 2 по протоколу FRI.

> **Раздел в разработке**
>
> Содержимое данного раздела находится в процессе подготовки.

RobotPowerControl

В данном разделе будет представлено описание управляющей программы **RobotPowerControl**, предназначенной для управления состоянием питания робота.

> **Раздел в разработке**
>
> Содержимое данного раздела находится в процессе подготовки.

2.4.2 Функциональные возможности

Станция

Раздел **Станция** является главным уровнем навигации интерфейса KUKA smartHMI. Он открывается нажатием кнопки **Станция** в строке навигации smartPAD и предоставляет доступ к основным функциям управления роботизированной ячейкой.

СТРУКТУРА МЕНЮ

Интерфейс раздела Станция включает четыре функциональных блока:

Блок	Описание
Меню навигации	Станция, Приложения, Меню робота, Меню IO Group
Меню станции	Данные процесса, Safety, Фреймы, KUKA_Sunrise_cabinet, Состояние HMI, Информация, Протокол
Дополнительное меню	Вид перемещения, Часы, Пользовательские кнопки
Функциональные кнопки smartPAD	Управление программой и перемещением

ДАННЫЕ ПРОЦЕССА

Раздел **Данные процесса** отображает текущее состояние активного приложения (например, `Ok`). Используется для мониторинга параметров выполняемой программы в режиме реального времени.

SAFETY

Раздел **Safety** предоставляет доступ к настройкам и состоянию системы безопасности робота.

Функции блока Safety

Функция	Описание
Состояние	Отображает текущее состояние конфигурации безопасности
Активация	Управление активацией и деактивацией конфигурации безопасности

Доступные действия на странице Активация

Действие	Описание
Активизировать	Применить и активировать текущую конфигурацию безопасности
Деактивировать	Отключить активную конфигурацию безопасности
Сброс	Сбросить конфигурацию безопасности к предыдущему состоянию

Поле **Id безопасной конфигурации** отображает уникальный идентификатор загруженной конфигурации (например, `2BCAB6DD`).

ФРЕЙМЫ

Раздел **Фреймы** открывает редактор систем координат.

Здесь отображается список всех фреймов, определённых в проекте Sunrise, с возможностью просмотра, корректировки и навигации по иерархии.

Структура таблицы фреймов

Столбец	Описание
Имя фрейма	Наименование фрейма в проекте
X, Y, Z	Смещение по осям в мм
A, B, C	Ориентация в градусах

Данные фреймов также доступны в приложении SunriseWorkbench.

Навигация и корректировка

Для перехода в дочерние фреймы нажмите кнопку **>** рядом с нужным фреймом. Путь навигации в строке «хлебных крошек» обновляется автоматически. Для возврата на предыдущий уровень нажмите соответствующий элемент в строке навигации.

При нажатии **Корректировать** открывается диалог с таблицей сравнения текущих и новых значений. Для подтверждения нажмите **Сохранить**, для отмены — **Отменить**.

Фреймы поддерживают многоуровневую вложенность. Полный путь иерархии отображается в строке навигации (например, `Универсальный > grant_RNF > P4`).

KUKA_SUNRISE_CABINET

Раздел **KUKA_Sunrise_Cabinet** отображает информацию о состоянии аппаратных компонентов контроллера.

Компонент	Описание
Состояние загрузки	Статус загрузки контроллера
Магистральные шины	Состояние шины EtherCAT

СОСТОЯНИЕ HMI

Раздел **Состояние HMI** отображает статус соединения между smartHMI и контроллером Sunrise Cabinet.

ПРОТОКОЛ

Раздел **Протокол** открывает журнал системных событий.

Фильтры журнала

Фильтр	Описание
Источник(и)	Станция, LBR_iiwa_7_R800 или оба
Уровень	Информация, Предупреждение, Ошибка
Промежуток времени	Временной диапазон выборки

Каждая запись содержит: иконку уровня, дату и время события, источник, наименование и описание события.

ИНФОРМАЦИЯ

Раздел **Информация** содержит подробные системные сведения о контроллере и подключённом роботе.

ФУНКЦИОНАЛЬНЫЕ КНОПКИ SMARTPAD

Физические кнопки smartPAD разделены на три группы: кнопки управления программой (левая панель), кнопки ручного управления осями (правая панель) и пользовательские кнопки.

Кнопки управления программой

Кнопка	Описание
Редактировать	Переход в режим обучения (Teach). Активирует возможность изменения точек программы в ручном режиме.
Стоп	Останавливает выполнение программы или движение робота.
Шаг назад	Выполняет один шаг программы в обратном направлении. Используется для отладки.
Старт	Запускает выбранное приложение или возобновляет остановленную программу. В режиме T1/T2 требует удержания enabling device.

Note

Режим редактирования со smartPAD не используется — все программы написаны на Java и изменяются только в SunriseWorkbench.

Кнопки управления осями (режимы T1 и T2)

Кнопка	Описание
A1 − / A1 +	Перемещение оси 1 в отрицательном / положительном направлении
A2 − / A2 +	Перемещение оси 2 в отрицательном / положительном направлении
A3 − / A3 +	Перемещение оси 3 в отрицательном / положительном направлении
A4 − / A4 +	Перемещение оси 4 в отрицательном / положительном направлении
A5 − / A5 +	Перемещение оси 5 в отрицательном / положительном направлении
A6 − / A6 +	Перемещение оси 6 в отрицательном / положительном направлении
A7 − / A7 +	Перемещение оси 7 в отрицательном / положительном направлении

При выборе режима декартового управления те же кнопки управляют перемещением TCP по осям X / Y / Z и ориентацией A / B / C.

Управление скоростью (Override)

Кнопка	Описание
0	Уменьшение скорости ручного перемещения
100	Увеличение скорости ручного перемещения

Значение отображается в процентах от максимальной скорости. В режиме T1 скорость TCP аппаратно ограничена до 250 мм/с.

Пользовательские кнопки

Четыре белые круглые кнопки в нижней части левой панели. Их назначение задаётся программно в Sunrise-проекте через API. По умолчанию не назначены.

РЕЖИМЫ РАБОТЫ

Режим	Описание
T1	Ручное управление, скорость ограничена до 250 мм/с по TCP. Требуется удержание enabling device.
T2	Ручное управление, нормальная скорость. Требуется удержание enabling device.
AUT	Автоматический режим. Кнопки осей недоступны, управление осуществляется кнопками Старт / Стоп.

Дополнительное меню

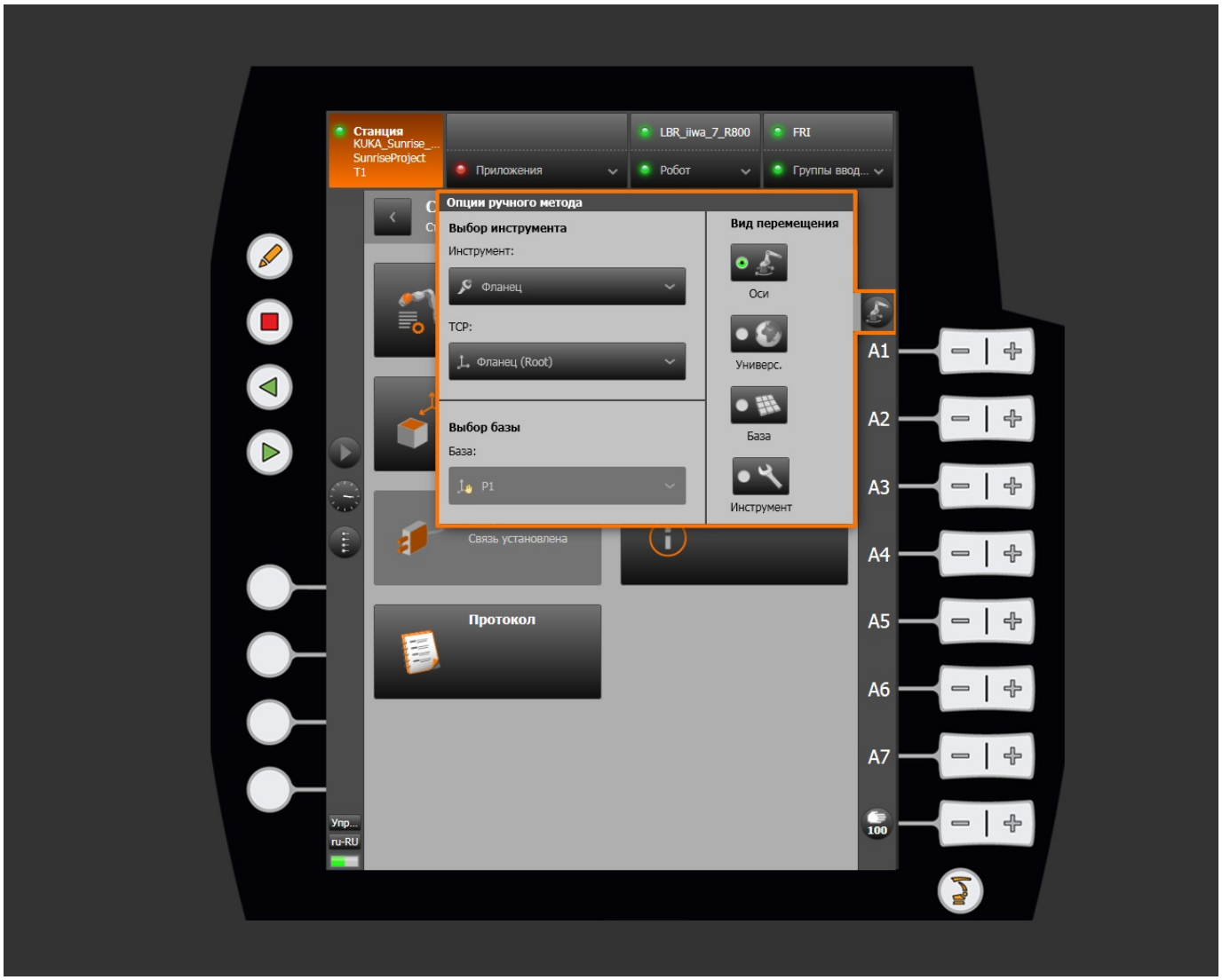
Описание дополнительных параметров управления приведено в разделе Дополнительное меню.

Дополнительное меню

Помимо основных функциональных блоков раздела Станция, smartHMI предоставляет доступ к дополнительным параметрам управления роботом через боковую панель smartPAD. Панели открываются поверх текущего вида и не требуют перехода в отдельный раздел интерфейса.

МЕТОД УПРАВЛЕНИЯ РОБОТОМ

Панель **Опции ручного метода** позволяет настроить параметры ручного перемещения: активный инструмент, точку управления (TCP), базовую систему координат и вид перемещения.



Выбор инструмента и TCP

Параметр	Описание
Инструмент	Активный инструмент, закреплённый на фланце. По умолчанию: `Фланец`
TCP	Точка управления инструментом. По умолчанию: `Фланец (Root)`

Выбор базы

Базовая система координат, относительно которой выполняется ручное перемещение. Выбирается из списка фреймов, определённых в проекте (например, `P1`).

Вид перемещения

Определяет систему координат, в которой работают кнопки осей A1–A7:

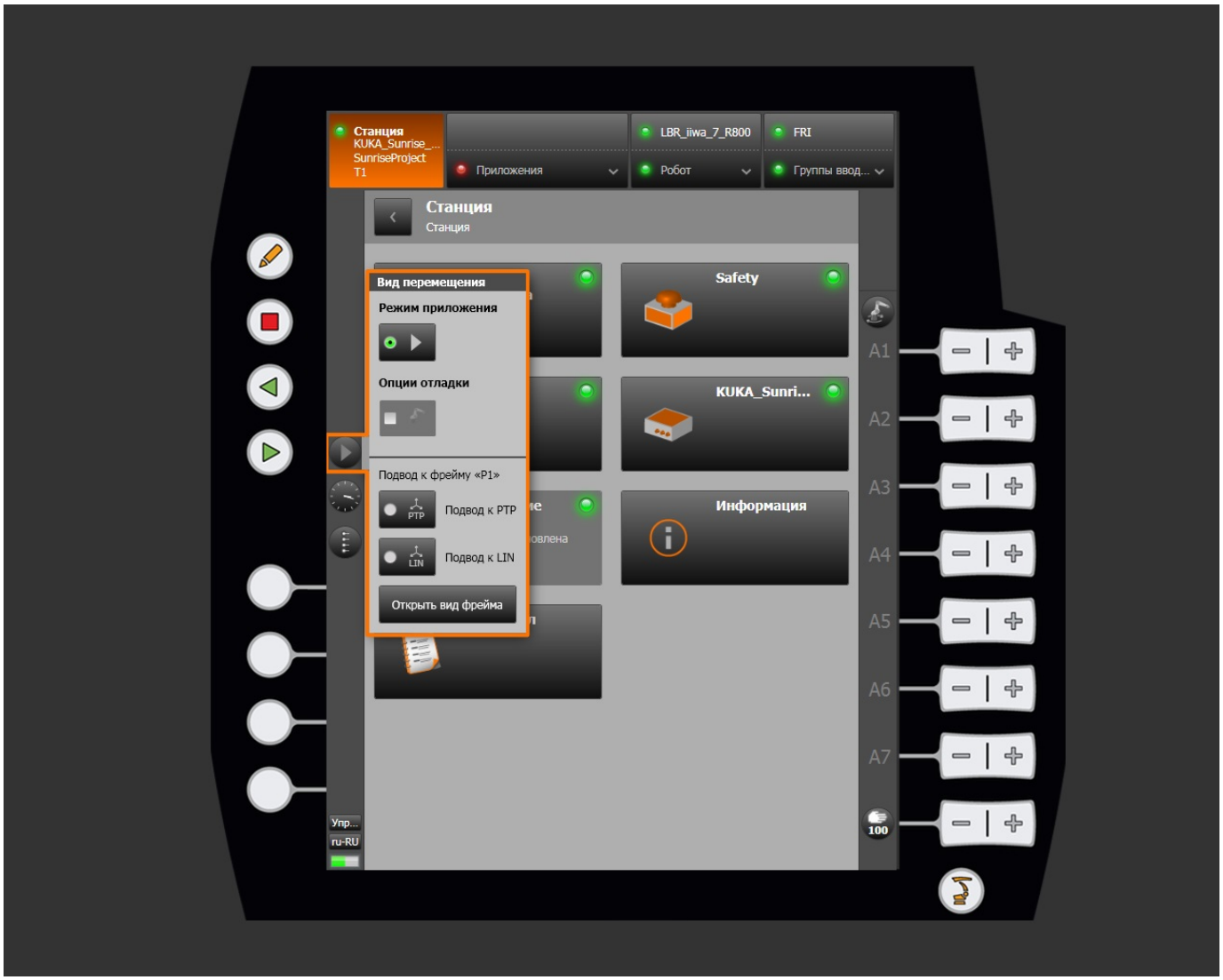
Режим	Описание
Оси	Поосевое управление. Каждая кнопка перемещает соответствующую ось независимо.
Универс.	Перемещение в универсальной (мировой) системе координат.
База	Перемещение в выбранной базовой системе координат.
Инструмент	Перемещение в системе координат активного инструмента (TCP).

СКОРОСТЬ УПРАВЛЕНИЯ

Панель **Скорость** управляет процентным ограничением скорости ручного перемещения и выполнения программы.

ВИД ПЕРЕМЕЩЕНИЯ

Панель **Вид перемещения** управляет режимом работы кнопки **Старт** и параметрами подвода к фреймам.



Режим приложения

Режим	Описание
Старт – непрерывный	Кнопка Старт запускает приложение в непрерывном режиме (по умолчанию).
Пошаговое выполнение	Кнопка Старт выполняет программу по одному шагу. Используется при отладке.

Подвод к фрейму

Тип	Описание
Подвод к PTP	Движение по кратчайшей траектории в пространстве суставов (Point-to-Point).
Подвод к LIN	Движение по прямолинейной траектории TCP в декартовом пространстве (Linear).

Кнопка **Открыть вид фрейма** выполняет переход в раздел Фрейм.

ЧАСЫ

Нажатие на иконку часов отображает текущее системное время и дату контроллера.

Note

Системное время синхронизировано с часами контроллера KUKA Sunrise Cabinet. Изменение времени выполняется в настройках операционной системы контроллера.

ГРУППА ПОЛЬЗОВАТЕЛЕЙ

Диалог **Войти в систему** обеспечивает смену активной группы пользователей и соответствующего уровня доступа к функциям HMI.

Уровень доступа определяет перечень доступных операций: редактирование фреймов, управление Safety, изменение конфигурации проекта и другие.

СМЕНА ЯЗЫКА

Диалог **Выбор языка** позволяет изменить язык интерфейса smartHMI.

Изменение вступает в силу немедленно без перезагрузки системы. Текущий язык интерфейса отображается в нижнем левом углу smartHMI (например, `ru-RU`).

Приложения

Раздел **Приложения** предоставляет доступ к выбору и управлению управляющими программами, разработанными в SunriseWorkbench и развёрнутыми на контроллере. Раздел открывается через кнопку **Приложения** в строке навигации smartHMI.

СПИСОК ПРИЛОЖЕНИЙ

Страница выбора приложений разделена на два столбца:

Столбец	Описание
Приложения робота	Управляющие программы, запускаемые вручную оператором.
Фоновые приложения	Программы, выполняющиеся автоматически (`backgroundTask`).

Каждая запись в списке содержит:

- **Индикатор состояния** — цветная точка слева от наименования.
- **Наименование приложения** — имя класса Java.
- **Пакет** — пространство имён или категория (например, `[application]`, `[ros]`, `[demo]`).
- **Чекбокс** — для выбора или деактивации приложения.

Кнопка **Сбросить выбранное приложение робота** (иконка руки) снимает выбор активного приложения без его остановки. Фоновые приложения, находящиеся в состоянии выполнения, отображаются с зелёной точкой и кнопкой **Стоп**.

ВЫБОР И АКТИВАЦИЯ ПРИЛОЖЕНИЯ

Для выбора приложения нажмите на его наименование в списке. Выбранное приложение выделяется оранжевым цветом, в чекбоксе появляется галочка (✓), а наименование отображается в строке навигации smartHMI. Система автоматически открывает страницу **Контроль приложений** с текущим состоянием программы и журналом её выполнения.

СОСТОЯНИЯ ПРИЛОЖЕНИЯ

Активизировано

Серый круглый индикатор — приложение выбрано и загружено в контроллер, однако ещё не запущено.

Выполнение

Зелёный индикатор воспроизведения — программа активно выполняется. В журнале в реальном времени отображаются события, определённые программистом. Если программа предусматривает взаимодействие с оператором, поверх журнала появляется диалог с кнопками выбора.

Перемещение приостановлено

Жёлтый индикатор паузы – выполнение прервано. Для возобновления продолжите выполнение программы, по тому же принципу как и запускали приложение.

Ошибка

Красный индикатор — в ходе выполнения возникло необработанное исключение. Строка состояния отображает код ошибки. Исправление логических ошибок выполняется в SunriseWorkbench.

ДЕАКТИВАЦИЯ ПРИЛОЖЕНИЯ

Для снятия выбора с приложения перейдите в раздел **Приложения**, найдите активное приложение (выделено оранжевым, чекбокс ✓) и нажмите на чекбокс – приложение будет деактивировано.


Станция
KUKA_Sunrise_...
SunriseProject
T1
TeachKuka
LBR_iiwa_7_R800
FRI
Приложения
Робот
Группы ввод...
Сбросить выбранное приложение робота
Приложения робота
Фоновые приложения
RobotApplication [demo]
RotationalImpedan... [toTry]
ServerFriRos2 [ros]
ShaskiTCPOpen [application]
SmartServoLINInt... [com.kuka.connectivit...
SmartServoLINSim... [com.kuka.connectivit...
SmartServoSample... [com.kuka.connectivit...
SmartServoSample... [com.kuka.connectivit...
SmartServoSample... [com.kuka.connectivit...
tcpTest [RNF_grant]
TeachKuka [application]
TestTestTest [RNF_grant]
TransformationPro... [com.kuka.connectivit...
TranslationalImpe... [toTry]
TransportPosition [utils]
Упр...
ru-RU
A1
A2
A3
A4
A5
A6
A7
100

Меню робота

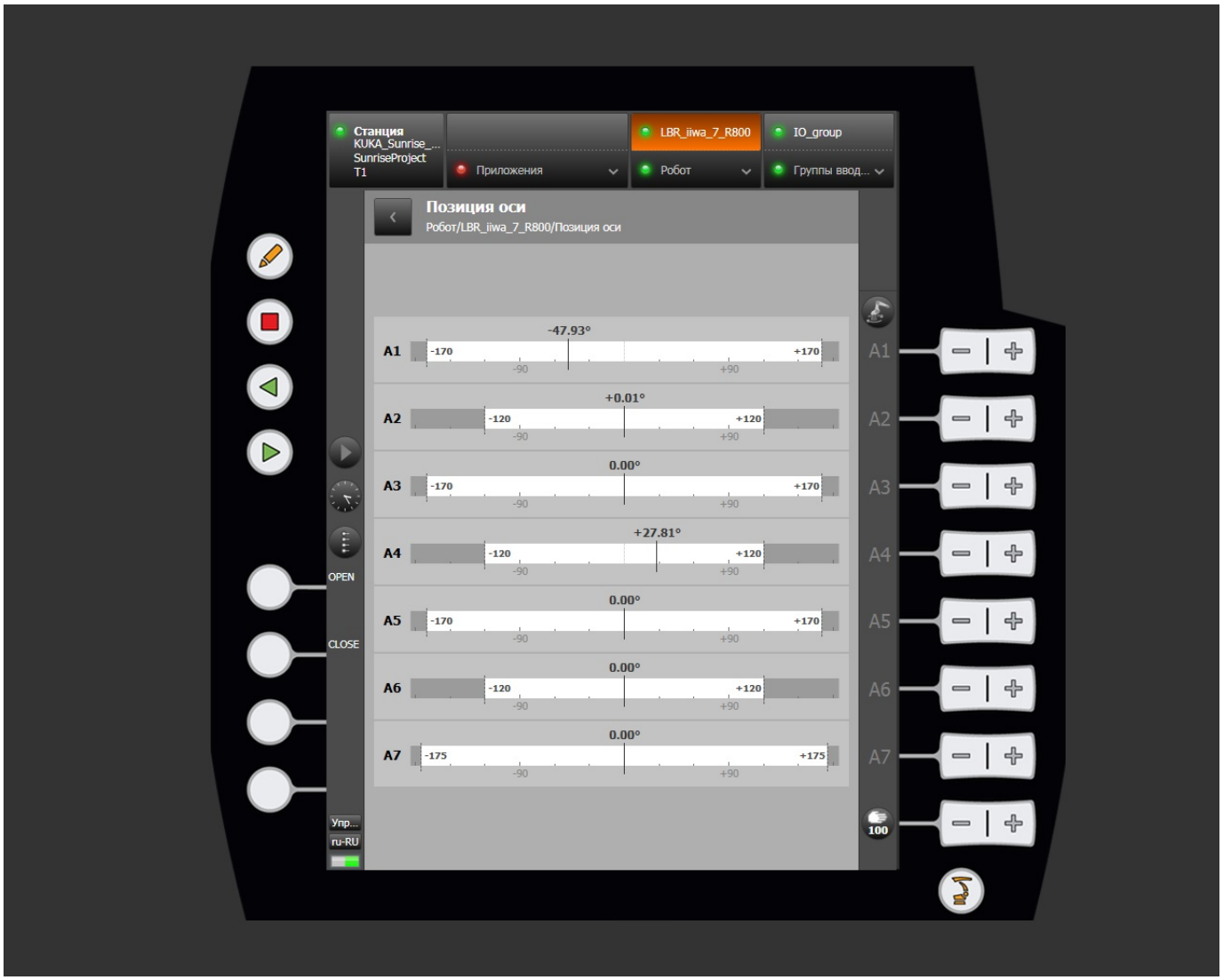
Раздел **Робот** является одним из основных разделов интерфейса KUKA smartHMI. Он предоставляет оператору доступ к диагностической информации о состоянии робота, функциям юстировки, калибровки инструментов и баз, а также к параметрам нагрузки.

В таблице ниже приведены основные пункты меню и их назначение.

Пункт	Описание
Позиция оси	Отображает текущую позицию каждой оси робота в градусах
Положение в прямоугольной системе координат	Отображает текущую позицию инструмента в декартовой системе координат
Моменты оси	Отображает текущие моменты на каждой оси робота
Юстировка	Предоставляет доступ к функциям юстировки осей и обучения смещений инструмента
Данные нагрузки	Позволяет ввести или откалибровать параметры нагрузки на фланце
Разрешение на перемещение	Отображает активность сигнала разрешения ручного перемещения
Протокол	Отображает журнал событий и ошибок (идентичен пункту в разделе Станция)
Состояние устройства	Отображает текущее состояние устройства с помощью цветового индикатора
Калибровка	Предоставляет доступ к функциям калибровки инструментов и баз

ПОЗИЦИЯ ОСИ

В данном разделе отображается текущая угловая позиция каждой из 7 осей робота в градусах. Значения обновляются в реальном времени. Для каждой оси также указаны программные ограничения в виде минимального и максимального допустимых значений.



ПОЛОЖЕНИЕ В ПРЯМОУГОЛЬНОЙ СИСТЕМЕ КООРДИНАТ

В данном разделе отображается положение точки управления инструментом (TCP) в декартовой системе координат относительно выбранной базы. Доступны следующие параметры:

- **X, Y, Z** — линейные координаты TCP (мм)
- **A, B, C** — углы ориентации (°)

> **Соответствие углов осям**
>
> Угол **A** соответствует вращению вокруг оси Z, угол **B** — вокруг оси Y, угол **C** — вокруг оси X (соглашение ZYX).

Также отображается информация о текущем контексте расчёта:

- Используемый инструмент (Tool)
- Активный TCP
- Используемая база (Base)

Параметры отображения можно изменить в разделе **Опции ручного метода** (см. Дополнительное меню), выбрав другой инструмент, TCP или базу. После выбора данные в этом разделе будут пересчитаны относительно новых параметров.

МОМЕНТЫ ОСИ

В данном разделе отображаются текущие крутящие моменты на каждой из 7 осей робота в ньютон-метрах (Нм). Значения обновляются в реальном времени. Эта информация позволяет оператору:

- контролировать нагрузку на каждую ось;
- диагностировать возможные механические проблемы;
- определять характерные моменты для последующей настройки управляющей программы.

ЮСТИРОВКА

Юстировка — это процедура привязки механического положения робота к его программной модели. Без корректной юстировки программные координатные данные не будут соответствовать фактическому положению осей.

Главное меню юстировки предоставляет доступ к следующим функциям: обновлению данных юстировки, разъюстировке отдельных осей, а также обучению смещений инструмента (Tool offset).

Для сохранения новых значений юстировки после выполнения процедуры используется функция **Обновить данные юстировки**. После её активации контроллер записывает текущее механическое положение осей как эталонное.

Функция **Разъюстировка** снимает данные юстировки с выбранной оси. После разъюстировки ось считается не откалиброванной и допускает перемещение за пределы программных ограничений.

Когда выполнять разъюстировку?

Разъюстировку следует выполнять, если одна из осей робота упёрлась в программное ограничение и не может продолжать движение. После разъюстировки и повторного выведения оси из ограничения необходимо обязательно выполнить повторную юстировку для восстановления корректной работы робота.

Функция **Обучение смещению инструмента** позволяет задать поправку к нулевому положению оси без полной переюстировки. Она применяется при незначительных смещениях механической системы.

Важно

Чтобв активировать функцию обучения смещения инструмента, необходимо выбрать инструмент, для которого будет выполняться обучение.

ДАННЫЕ НАГРУЗКИ

Корректное задание параметров нагрузки необходимо для точного планирования движений, предотвращения перегрузки осей и правильной работы функций ограничения мощности и усилия (Power and Force Limiting, PFL).

Главное меню данных нагрузки отображает список доступных слотов инструментов. Для каждого инструмента можно задать или откалибровать параметры нагрузки.

В разделе **Определение данных нагрузки** выполняется ввод или автоматическое определение параметров нагрузки: массы, центра масс и тензора инерции.

Процедура **Калибровки массы инструмента** позволяет контроллеру автоматически измерить массу закреплённого инструмента путём выполнения тестовых движений. Следуйте инструкциям на экране для завершения процедуры.

По завершении процедуры контроллер отображает измеренные значения параметров нагрузки. Рекомендуется проверить результаты и убедиться в их соответствии фактическим характеристикам инструмента.

После подтверждения обновлённые данные нагрузки сохраняются в контроллере и вступают в силу немедленно.

РАЗРЕШЕНИЕ НА ПЕРЕМЕЩЕНИЕ

Данный пункт отображает активность сигнала разрешения ручного перемещения (Move Enable). Сигнал активируется при нажатии на кнопку разрешения на корпусе манипулятора. Активное состояние визуально отображается на smartHMI: цвет индикаторов осей изменяется с серого на белый.

> **Ограничение режима**
>
> Функция разрешения перемещения недоступна в **автоматическом режиме** (Auto). Сигнал активен только в режимах ручного управления (T1, T2).

ПРОТОКОЛ

Данный раздел полностью эквивалентен пункту **Протокол**, описанному в разделе Станция. Отображается журнал событий, предупреждений и ошибок контроллера, что позволяет оператору отслеживать хронологию событий и диагностировать возникшие неисправности.

СОСТОЯНИЕ УСТРОЙСТВА

В данном разделе отображается текущее состояние устройства в виде цветового индикатора:

Цвет	Состояние
Зелёный	Устройство функционирует нормально
Жёлтый	Активно предупреждение или потенциальная проблема, требующая внимания
Красный	Обнаружена критическая ошибка или неисправность

КАЛИБРОВКА

Раздел **Калибровка** обеспечивает доступ к процедурам определения геометрических параметров инструментов и баз, которые используются контроллером для расчёта декартовых координат.

Главное меню содержит две основные категории калибровки: калибровка базы (Base) и калибровка инструмента (Tool).

Калибровка базы позволяет задать положение рабочей системы координат относительно мировой системы координат (World). Используется для привязки программы к конкретному местоположению заготовки или оборудования в рабочей ячейке.

Калибровка инструмента определяет положение TCP (Tool Center Point) и ориентацию инструмента относительно фланца робота. Для каждого инструмента доступно несколько методов калибровки.

Выбор метода калибровки инструмента определяет порядок выполнения процедуры. Наиболее распространённый метод — **XYZ 4-Point** — предполагает подвод TCP к одной контрольной точке из четырёх различных ориентаций.


Станция
KUKA_Sunrise_...
SunriseProject
T1
Приложения
LBR_iiwa_7_R800
Робот
IO_group
Группы ввод...
калибровка инструмента;
Робот/LBR_iiwa_7_R800/Калибровка/калибровка инструмента;
Инструмент
patron
TCP
TCP
трансформация
[мм]
X 0.00 Y 0.00 Z 90.00
[°]
A 0.00 B 0.00 C 0.00
Погрешность вычислений
Метод
XYZ, 4 точки
Точка измерения
Преобразование (в мм)
Точка измерения 1
Определение точки измерения
Точка измерения 2
Определение точки измерения
Точка измерения 3
Определение точки измерения
Точка измерения 4
Определение точки измерения
Определить параметры инструмента
Записать точку измерения
OPEN
CLOSE
Упр...
ru-RU
A1
A2
A3
A4
A5
A6
A7
100

Меню IO Group

Раздел **Группы вводов/выводов** предоставляет доступ к мониторингу и ручному управлению цифровыми сигналами ввода/вывода, сконфигурированными в проекте Sunrise. Раздел открывается через строку навигации smartHMI.

ДОСТУПНЫЕ ГРУППЫ

При нажатии на кнопку **Группы вводов/выводов** в строке навигации открывается список доступных групп. В данном проекте определены следующие группы:

Группа	Описание
FRI	Группа сигналов интерфейса FRI (Fast Robot Interface)
IO_group	Пользовательская группа цифровых вводов/выводов

ПРОСМОТР СИГНАЛОВ

После выбора группы открывается страница со списком всех сигналов.

Структура таблицы сигналов

Столбец	Описание
Вход / Выход	Иконка направления сигнала
Имя	Наименование сигнала (например, `In_1`, `Out_16`)
Тип	Тип сигнала (для данной группы — цифровой Boolean)
Значение	Текущее состояние сигнала (`0` / `1`)

УПРАВЛЕНИЕ ВЫХОДНЫМИ СИГНАЛАМИ

Для сигналов с направлением **Выход** в нижней панели отображаются кнопки принудительной установки значения:

Кнопка	Действие
True	Установить значение выходного сигнала в `1` (активен)
False	Установить значение выходного сигнала в `0` (неактивен)

Note

Сигналы с направлением **Вход** доступны только для чтения.

3. Решение проблем

3.1 Ошибка конфигурации

При запуске управляющих программ может возникнуть ошибка, указывающая на то, что инструмент или фрейм не найден. Причиной является аварийное отключение контроллера, при котором конфигурация теряется.

3.1.1 Признаки проблемы

Проблема проявляется при запуске программ **TeachKuka** или **LBRserver**. Диагностические признаки в главном меню smartHMI:

- Жёлтые предупреждающие индикаторы в разделе **Данные процесса**;
- Жёлтые предупреждающие индикаторы в разделе **Фреймы**.


Станция
KUKA_Sunrise_...
SunriseProject
T1
LBR_iiwa_7_R800
FRI
Приложения
Робот
Группы ввод...
Данные процесса
Станция/Данные процесса
Не удалось загрузить данные процесса. Для получения более детальной информации см. файл журнала.
A1
A2
A3
A4
A5
A6
A7
100
Упр...
ru-RU


Станция
KUKA_Sunrise_...
SunriseProject
T1
Приложения
LBR_iiwa_7_R800
Робот
FRI
Группы ввод...
Фреймы
Станция/Фреймы
Не удалось загрузить фреймы. Для получения более детальной информации см. файл журнала.
A1
A2
A3
A4
A5
A6
A7
100
Упр...
ru-RU

3.1.2 Порядок устранения

Выполните следующую последовательность действий:

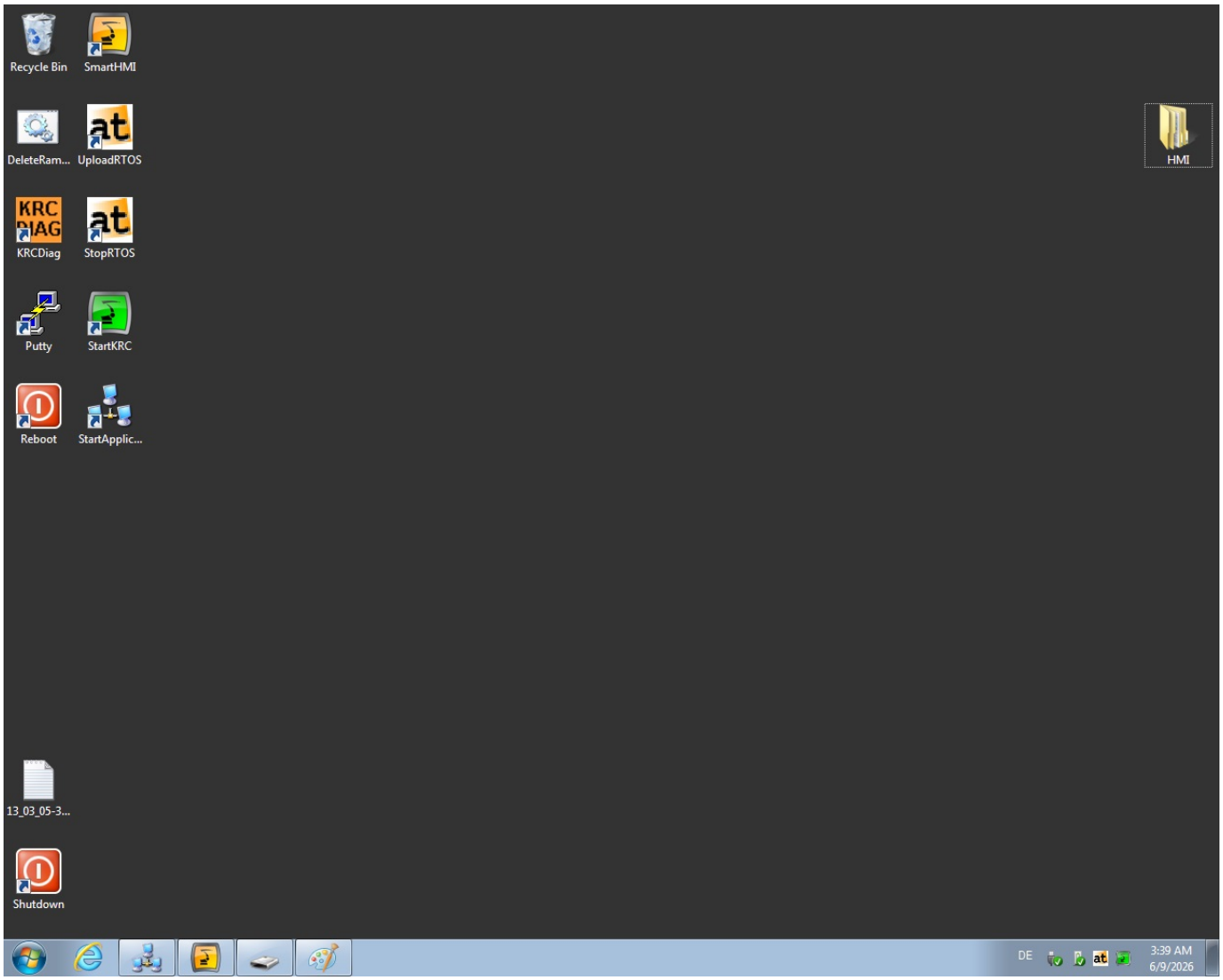
1. Подключите внешний монитор к контроллеру робота.
2. Перезагрузите контроллер.
3. Выполните вход в систему с учётными данными:
   - Логин: `KukaUser`
   - Пароль: `68kuka1secpw59`

> **Раскладка клавиатуры**
>
> По умолчанию на контроллере установлена немецкая раскладка клавиатуры (`DE`). Учитывайте это при вводе пароля.

4. Скопируйте полный проект на USB-накопитель.
5. Откройте проводник файлов сочетанием клавиш `⊞ Win` + `E`.
6. Перейдите по пути:

```
C:\KRC\Projects
```

7. Замените все файлы в данной директории версиями с USB-накопителя.
8. Запустите программу перезагрузки системы с рабочего стола контроллера.



После перезагрузки конфигурация будет восстановлена, и управляющие программы запустятся корректно.