

Контактная информация

Клюнин Алексей Олегович

aoklyunin@gmail.com

+7(921) 438-98-10

<https://github.com/aok-buran>

<https://habr.com/ru/users/aok-buran/>

Стек технологий:

Java, C/C++, C#, MATLAB, ROS, Android SDK, MySQL, Linux, Git, Python, Django, Docusaurus, Docker, Jenkins, QT, QNX

Образование:

- В 2012 году закончил ФМЛ № 239 в Санкт-Петербурге,
- 2012-2016 - бакалавриат кафедры СУиР университета ИТМО, тема диплома: «Силомоментное очувствление промышленного робота-манипулятора».
- 2016-2018 - магистратура кафедры СУиР, университета ИТМО, тема диплома: «Синтез алгоритмов управления мобильным роботом», диплом с отличием

Опыт работы

- 2013 – 2017 - инженер КИП и АСУ, ООО «НПО Поиск». Занимался проектированием, сборкой и наладкой автоматики. Также мною спроектирован и введен в эксплуатацию [станок по намотке баллонов](#). Суть работы станка в том, что необходимо синхронизировать движение рабочего инструмента с вращением баллона так, чтобы на баллоне получался заданный циклический рисунок. Это повышает надёжность и придает товарный вид изделию. Изделия задействованы в производственных цепочках ОПК РФ.
- 2015 – 2019 год инженер кафедры СУиИ университета ИТМО, сотрудник [Международного научного центра "Нелинейные и адаптивные системы управления"](#).
- 2018 – 2019 года инженер в компании TRA Robotics. Разработка программного обеспечения для управления робототехническими производственными линиями.
- 2019 – 2021 инженер-программист, ООО «НПО Поиск». Разработка аппаратно-программных комплексов неразрушающего контроля композитных изделий, модернизация и проектирование станков и измерительных систем.
- 2021 – 2022 проектирование и разработка фреймворка планирования движения для системы роботов-манипуляторов. [Обзорная статья на хабре](#). Данный фреймворк – моя личная разработка. С документацией фреймворка можно ознакомиться [здесь](#). Он написан на C++, собирается в deb-пакет с помощью Jenkins. Используемые технологии: C++, QT, OpenGL, Boost, Eigen
- С 2022 по настоящее время – разрабатываю различное программное обеспечение на заказ: от ERP систем до встраиваемого ПО, решаю прикладные математические задачи



Публикации

- [Быстрый поиск изоморфных подграфов \(habr.com\)](#)
- [Начало шестого технологического уклада \(habr.com\)](#)
- [Монотонная кубическая интерполяция \(habr.com\)](#)
- Интеграция робототехнических единиц в единую систему 2017 Сборник тезисов участников форума "Наука будущего-наука молодых" с. 309-310 А.О. Ключин, И.В. Петраневский
- Проблемы интеграции и взаимодействия роботов в производственную цепочку механической обработки деталей сложной геометрической формы 2017/2/21 А. О. Ключин, И. В. Петраневский, С. А. Колюбин, О. И. Борисов, В. С. Громов
- Идентификация сухого и вязкого трений в сочленениях робота-манипулятора И.В. Петраневский, А. О. Ключин, А.А. Пыркин - Навигация и управление движением, 2018
- Case study on human-free water heaters production for industry 4.0 Oleg I Borisov, Vladislav S Gromov, Sergey A Kolyubin, Anton A Pyrkin, Nikolay Y Dema, Vladimir I Salikhov, Igor V Petranevsky, Alexey O Klyunin, Sergey V Shavetov, Alexey A Bobtsov, 2018 IEEE Industrial Cyber-Physical Systems (ICPS) p. 369-374
- Силомоментное очувствление промышленного робота-манипулятора в задаче полировки 2017/2/21 И. В. Петраневский, А. О. Ключин, С. А. Колюбин, О. И. Борисов, В. С. Громов
- Идентификация статической модели робота-манипулятора А.О. Ключин, И.В. Петраневский, А.А. Пыркин - Навигация и управление движением, 2018
- Arc approximation algorithm of spatial movements for industrial robots Oleg I Borisov, Vladislav S Gromov, Anton A Pyrkin, Igor V Petranevsky, Alexey O Klyunin, Alexey A Bobtsov IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society 3429-3434
- Output robust control with anti-windup compensation for robotic boat Oleg I Borisov, Vladislav S Gromov, Anton A Pyrkin, Alexey A Bobtsov, Igor V Petranevsky, Alexey O Klyunin 2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR) p. 13-18
- Human-free robotic automation of industrial operations Oleg I Borisov, Vladislav S Gromov, Sergey A Kolyubin, Anton A Pyrkin, Alexey A Bobtsov, Vladimir I Salikhov, Alexey O Klyunin, Igor V Petranevsky IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society p.6867-6872

Патенты

- Программа для управления робототехнической системой посредством силомоментного датчика RU 2016619315
- Gb PCT/RU2018/000737 SENSOR-FREE FORCE/TORQUE SENSING IN AN ARTICULATED ELECTROMECHANICAL ACTUATOR-DRIVEN ROBOT
- EP3880412 (A1) - SENSOR-FREE FORCE/TORQUE SENSING IN AN ARTICULATED ELECTROMECHANICAL ACTUATOR-DRIVEN ROBOT
- US2022009097 SENSOR-FREE FORCE/TORQUE SENSING IN AN ARTICULATED ELECTROMECHANICAL ACTUATOR-DRIVEN ROBOT

Участие в конференциях и конкурсах

- XIX конференция молодых ученых «Навигация и управление движением» 2017 год
- Награды за лучший научно-исследовательский доклад Всероссийский конгресс молодых учёных в 2014 и 2016 годах
- Форум «Наука молодых, наука будущего» в 2016 и 2017 годах, второе место в конкурсе научных докладов в 2017 году.

О себе

- Занимал призовые места на региональных олимпиадах по математике, физике, химии и информатике
- Интересуюсь философией и нейрофизиологией
- Эксперт рабочей группы федерального проекта «Кадры для цифровой трансформации»

1. НПО Поиск

Мною реализован ряд проектов по автоматизации для [НПО Поиск](#). Эта организация занимается опытным и мелкосерийным производством композитных баллонов и агрегатов высокого давления. Самые значительные из моих проектов, реализованных для данной компании, описаны ниже.

1.1. Намотка баллонов

Чтобы снизить вес баллонов, поверх металлической оправки наматывается нить из стеклотекстолита, пропитанная эпоксидной смолой. За счёт такого упрочнения удаётся снизить толщину металлической оправки, а значит, и вес баллона в целом при равных предельных давлениях.

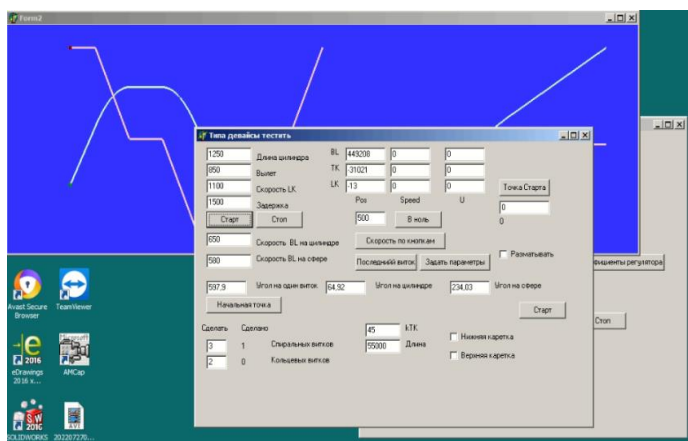


Видео работы намоточного станка после моей автоматизации можно посмотреть [здесь](#). Процесс намотки осуществляется следующим образом: на вращающийся баллон наматываются две нити, пропущенные через подвижные каретки.

Эти каретки движутся по определённому профилю скорости, смещая тем самым точку укладки нити. При помощи различных режимов движения можно добиться различных рисунков намотки.

Изначально станок функционировал за счёт срабатывания концевых датчиков. Когда металлические пластины, закреплённые на каретках, активировали тот или иной датчик, электрическое реле меняло направление скорости каретки на противоположное. Модуль скорости задавался с помощью потенциометра.

Программное автоматическое управление позволило синхронизировать движение кареток и баллона таким образом, чтобы каждая из координат станка обрабатывала заданную функцию положения от времени с приемлемой для технического процесса ошибкой. Подробнее о первой версии системы можно прочитать [здесь](#).



Программное обеспечение было написано на Delphi, т. к. не было серьёзных требований к скорости работы ПО, а эта среда позволяет очень быстро и удобно создавать формы. К тому же, в ней были встроенные инструменты для работы с COM-портом и сокетами.

Приложение состояло из двух частей: сервер и клиент. Сервер обрабатывал задания от клиента по положению и скорости в текущий момент времени, а клиент формировал эти задания в цикле по той или иной параметризованной траектории (некоторый аналог G-кодов).

Во второй версии программы управление приводами было переведено на цифровые каналы управления. Два маломощных частотных регулятора управлялись с помощью протокола Modbus (RS-232), а третий, движущий нижнюю быстроходную каретку, - с помощью внутреннего недокументированного протокола SEW. Т. к. программное обеспечение поставщика работало поверх RS-232, то с помощью UART-сниффера я распознал структуру пакетов для простых команд движения влево и вправо с заданной скоростью, затем подобрал алгоритм контрольной суммы.

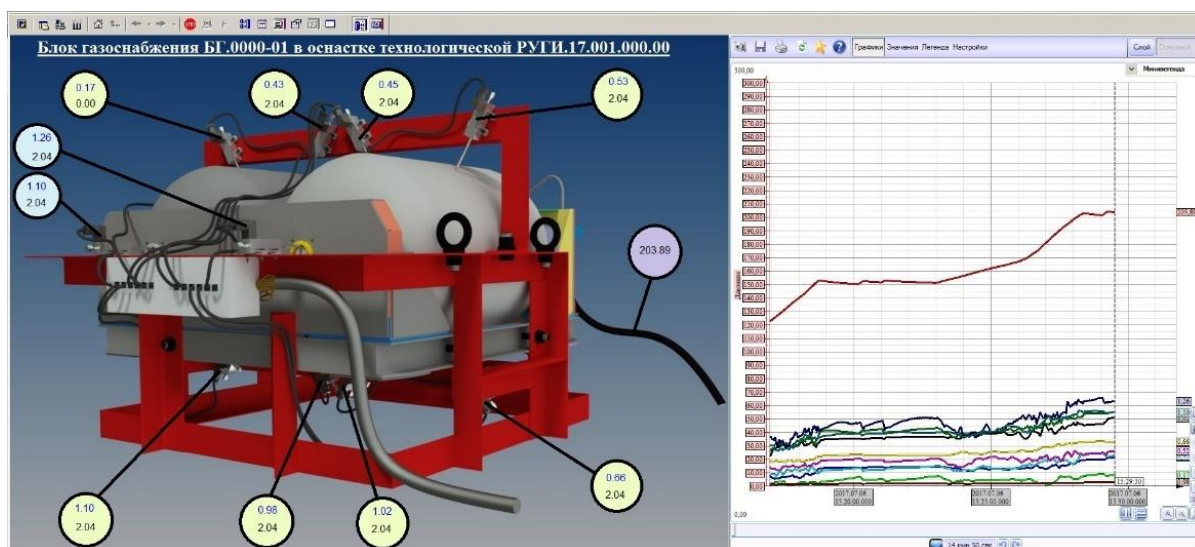
1.2. Измерение деформаций

У каждого баллона есть предельно допустимые деформации, зависящие от нагрузки.

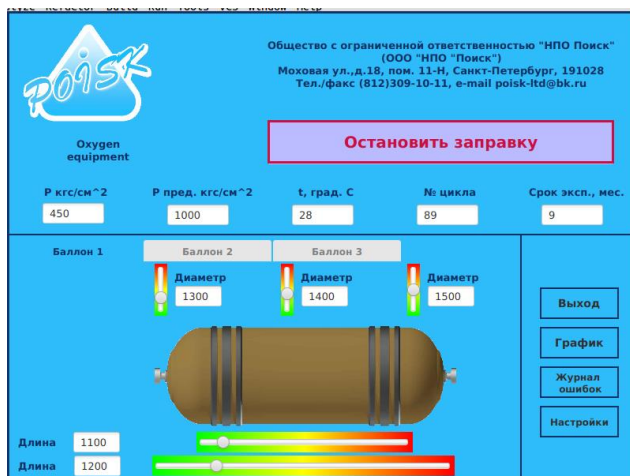
Сравнение реальных и теоретических смещений точек измерения на поверхности позволяет установить, пригоден баллон для дальнейшей эксплуатации или нет. Можно оснастить установку датчиками перемещения и с их помощью следить за состоянием баллона без демонтажа и транспортировки.

Для измерения перемещений точек поверхности баллонов мною были использованы резистивные датчики перемещения и промышленный контроллер. Также этот контроллер обрабатывал показания датчика давления. Из-за работы с высоким давлением испытательная зона находилась далеко от рабочего места оператора.

Испытания проводились посредством нагнетания высокого давления в баллонах. В качестве рабочего вещества используется жидкость из-за её низкой сжимаемости. Такой режим работы не поддерживается обычными промышленными контроллерами. В случае разрыва баллона контроллер может быть разрушен. Поэтому было решено вывести контроллер из испытательной зоны, а все датчики подключить с помощью ethernet-кабеля и patch-панелей.



Такое подключение добавляет паразитное сопротивление в резистивных датчиках. Чтобы его скомпенсировать, была проведена серия экспериментов с эталонными расстояниями в форме пластин для каждого датчика. После этого с помощью аппроксимации полиномом пятой степени в Matlab была установлена зависимость между измеренным значением на датчике и реальным. Компьютерная система диагностики была построена на основе MasterScada.



Далее была разработана система диагностики более габаритных баллонов. Проект руководства по эксплуатации (собран с помощью tex) можно скачать [здесь](#).

Также была разработана экспериментальная программа для связи с контроллером напрямую по COM-порту (протокол Modbus). В ней производился общий расчёт изменения геометрии баллонов. Это важно, т. к. во время нагрузки баллон может перемещаться и изгибаться.

1.3. Система газификации

Инженеры НПО "Поиск" разработали криогенную систему обеспечения газом КСОГ-1. Она предназначалась для перевода азота и метана из жидкой фазы в газообразную исключительно за счёт тепла, получаемого из окружающей среды. Электроэнергия была необходима только для питания датчиков.

Для исследования этой системы был проведён ряд экспериментов с газификацией безопасного азота, а после - метана. В ходе экспериментов измерялись температура в разных точках системы, давление внутри ёмкости, а также изменения веса при стравливании.

Эти данные фиксировались с помощью системы датчиков, промышленного контроллера и MasterScada. Для записи изменений веса отдельно была написана программа, работающая с цифровыми весами по СОМ-порту. После эксперимента показания веса синхронизировались с показаниями остальных датчиков по времени с помощью характерных точек.

По результатам экспериментов был составлен pdf-отчёт, который можно скачать [здесь](#). Также я смонтировал отчётное видео, с которым можно ознакомиться [здесь](#).

Для создания расширяющихся графиков из видео мной была написана вспомогательная программа. Исходные данные экспериментов сохранялись в формате csv. На их основе я порождал последовательность кадров, каждый из которых отображал всё большую и большую часть измерений.



2. Робототехника

Около пяти лет я работал младшим научным сотрудником на факультете СУиР университета ИТМО. Мною с коллегами по кафедре опубликован ряд статей как в российских, так и в зарубежных научных журналах. Их списки приведены на последних страницах портфолио.

В этом разделе будут описаны мои основные научные наработки, а также прикладные решения. В университете я в основном работал с промышленным роботом Kawasaki fs06n и мобильным роботом Kuka Youbot. На основе придуманного мною алгоритма было получено несколько патентов, в том числе зарубежных.

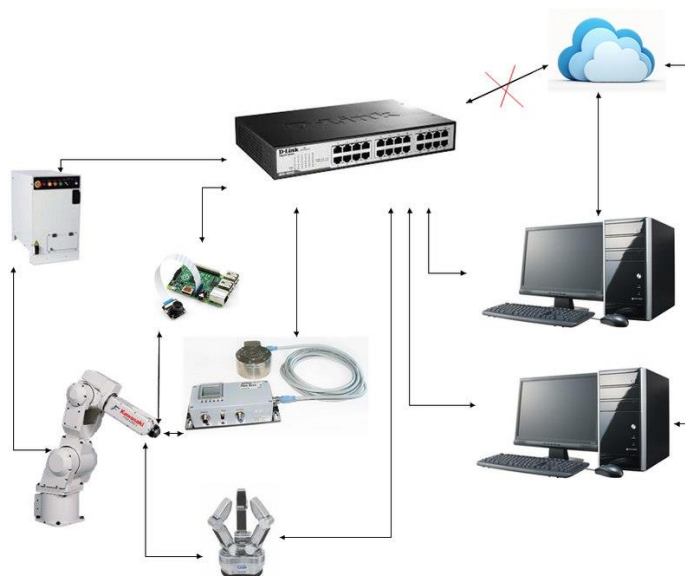
2.1. Силомоментное оцувствление

Несмотря на то, что звенья промышленных роботов позиционируются достаточно точно относительно друг друга, при обработке заготовок очень сложно достичь точного расположения рабочего инструмента относительно них. Помимо этого, часть технологических процессов требует воздействия с заданной силой или моментом.

В таких случаях на энд-эффектор робота устанавливается силомоментный датчик, в моём случае - ATI F/T IP60 Delta, а уже на него рабочий инструмент. Тогда, зная теоретические и реальные силу и момент, создаваемые рабочим инструментом, можно получить избыточные значения. Эти избыточные значения будут вызваны реакцией заготовки на воздействие инструментом.

Для демонстрации силомоментного очувствления я разработал программу, которая по сети считывала показания силомоментного датчика и определяла теоретическое воздействие рабочего инструмента. После этого она пересчитывала избыточные показания в абсолютную систему координат и формировала задание, куда сместить рабочий инструмент.

Получилось два режима работы. В первом можно было давить рукой на рабочий инструмент робота, и он смещался в соответствующую сторону. Такой режим называется Force Feedback. Демо-видео можно посмотреть [здесь](#).



Во втором режиме можно было поворачивать рабочий инструмент за счёт приложенных внешних моментов. Режим управления по моментам называется Torque Feedback. Демо-видео можно посмотреть [здесь](#). Подробнее об этой работе можно прочитать в моей [бакалаврской диссертации](#).

Чтобы построить такую систему, мне пришлось разобраться с внутренним языком Kawasaki, он довольно плохо документирован. В итоге выяснилось, что самый оптимальный способ управления – это контроль позиции. На контроллере в бесконечном цикле обрабатываются входящие команды по ethernet, пока не будет получена команда завершения работы. При этом пока не будет отработано предыдущее задание, отработка нового не начнётся. Структуру пакета пришлось составлять самостоятельно.

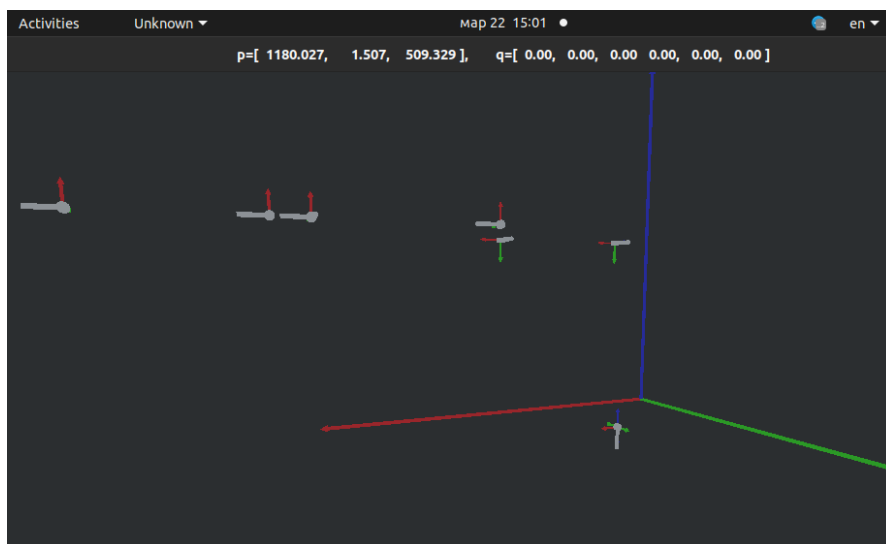
Строго говоря, такой робот вообще не был рассчитан под работу с обратной связью, поэтому быстрая работа системы происходила рывками, а плавный режим получался слишком медленным. Передвигать робота рукой, как на видео, без соответствующего алгоритма нельзя. Он поддерживает заданное положение и компенсирует все внешние воздействия.

Программа управления была написана на Java состояла из двух модулей: клиент и сервер. Сервер обобщал все показания системы и получал задание по перемещению от клиента. В качестве клиента я написал отдельно десктоп-программу для управления с обратной связью, отдельно Android-приложение, которое позволяло перемещать робота в заданное положение и поворачивать звенья в заданную конфигурацию. Также отображались показания силомоментного датчика. Для работы с датчиком AT1 в матлаб была написана библиотека. Демо-видео можно посмотреть [здесь](#).

2.2. Визуализация параметров Денавита-Хартенберга

Чтобы получить координаты рабочего инструмента робота-манипулятора в мировой системе координат, нужно составить матрицы перехода из каждого звена в следующее. Для этого каждое звено параметризуется четырьмя значениями: два отвечают за расстояния между звеньями, два – за повороты. Эти параметры называют параметрами Денавита-Хартенберга или ДХ-параметрами.

Для составления ДХ-параметров реального робота недостаточно составить кинематическую схему, требуется совместить нулевые точки (точки отсчёта углов) в звеньях робота. Точки отсчёта углов в каждом из сочленений робота не стандартизированы. Поэтому ДХ-параметры, составленные по кинематической схеме, могут давать положения рабочего инструмента, качественно отличные от реальных.



Для определения точек отсчёта и уточнения ДХ параметров была написана 3D программа визуализации параметров ДХ (C++ и OpenGL). Исходники можно скачать [здесь](#).

2.3. Обратная задача кинематики для мобильных роботов

У робота есть два способа определения пространства состояний:

1. положение с ориентацией рабочего инструмента
2. углы поворотов звеньев.

Первое называется операционным пространством, второе – конфигурационным.

Переход из конфигурационного пространства однозначен (прямая задача кинематики), а в обратную сторону (обратная задача кинематики) – нет. Из углов поворота звеньев робота мы однозначно можем получить положение и ориентацию рабочего инструмента. Решение обратной задачи достаточно трудоёмко.

Это вызвано сильной нелинейностью связи конфигурационного и операционного пространств. Обычно предлагается решать эту задачу геометрически либо через взятие частных производных.

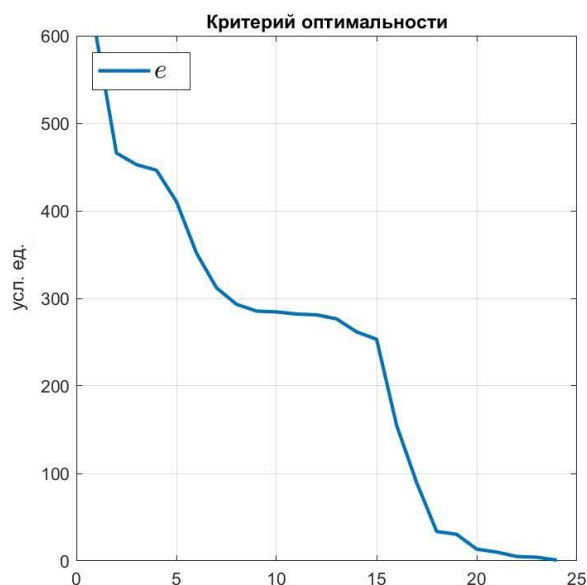
В своей [магистерской диссертации](#) я предложил новый способ решения обратной задачи кинематики. Обычно ориентация рабочего инструмента робота параметризуется углами Эйлера. Однако они определяются неоднозначно и имеют разрывы, если рассматривать их зависимость от углов поворота звеньев.

Я ввёл новую параметризацию через кватернионы, выраженные параметрами Родриго-Гамильтона. Полученные зависимости оказались непрерывными относительно конфигурационного пространства.

Это позволило составить функцию ошибки между текущим положением рабочего инструмента и желаемым. При этом функция ошибки в конечном счёте являлась скалярной функцией, зависящей только от углов поворота звеньев. Тогда стандартными методами для каждого робота можно составить



функцию такого рода, а потом классическими методами минимизировать значение, т. е. свести ошибку решения к нулю.



Для проверки разработанного алгоритма для робота Kuka Youbot было составлено несколько опорных точек, каждая из которых описывала произвольные достижимые положения рабочего инструмента и ориентацию. По каждой из точек с помощью минимизации были получены соответствующие углы поворота звеньев, после чего с помощью Gazebo и Ros был составлен python-скрипт, который последовательно отправлял виртуальному роботу задания по углам поворота звеньев и сравнивал полученные положение и ориентацию с заданными.

2.4. Бездатчиковое управление

Чтобы поддерживать заданную силу воздействия на рабочем инструменте робота, необязательно оснащать его силомоментным датчиком. Можно оценить динамическую модель робота, после чего сверить реальные моменты в сочленениях и теоретические. Этот избыток можно пересчитать в силу и момент на рабочем инструменте. Динамическая модель состоит из трёх компонент: первая зависит от ускорения - инерция, вторая от угла и скорости – различные трения, а третья – только от угла - гравитационная компонента.

Чтобы идентифицировать гравитационную компоненту, робот размещался в различных точках. Т.к. гравитационная компонента - это частная производная потенциальной энергии, то я выразил потенциальную энергию каждого из звеньев через матрицы преобразования и локальные радиус-векторы их центров тяжести. Звенья были идеализированно представлены, как невесомые стержни с прикреплённым точечным весом. При движении звена соответствующий точечный вес движется вместе с ним. За счёт такого допущения я составил явные зависимости потенциальных энергий каждого из звеньев от углов поворота звеньев.

После этого я продифференцировал суммарную потенциальную энергию системы по каждой из обобщённых координат (углов поворота звеньев) и получил зависимость гравитационной компоненты от них.

Робот Kuka Youbot был демонтирован с колёсной базы, после чего был проведён ряд экспериментов с фиксацией манипулятора в произвольных положениях. В каждом из



этих положений замерялась сила тока в каждом из сочленений. Она пропорционально связана с моментом в звене. Эксперименты выполнялись с помощью программ, написанных на Python+Ros.

Измеренные зависимости я разложил в виде матричного уравнения и применил метод наименьших квадратов. Сначала часть значений определялась с ошибкой, т. к. у составленной матричной функции число обусловленности приближалось к бесконечности. Это было связано с тем, что часть строк оказалась линейно зависимой (матрица неполного ранга), поэтому пришлось их найти и построить усечённую регрессионную модель.

Для идентификации компоненты трения был проведён ряд экспериментов с каждым из звеньев, кроме первого. Суть экспериментов заключалась в симметричном движении относительно вертикального положения из крайнего положения в крайнее. Получившиеся два участка различались только на гравитационную компоненту, причём взятую с разным знаком. Поэтому я с помощью Matlab достроил недостающие симметричные значения с каждой из сторон и потом попарно сложил их. С помощью аппроксимации были установлены оценки зависимости компоненты трения для каждого задействованного в эксперименте звена.

По составленной оценке гравитационной компоненты я разработал аналог программы управления по моментам роботом Kawasaki, но без использования силомоментного датчика. Реальные токи на каждом звене сверялись с оцененными, после чего их разница преобразовывалась в оценку внешнего воздействия. Демо-видео находится [здесь](#).

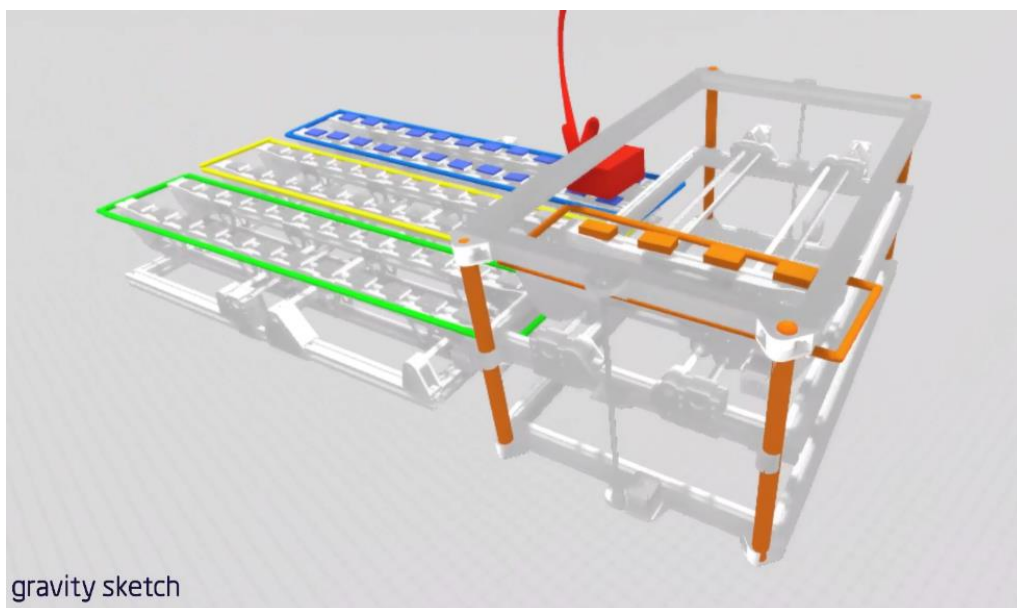
2.5. TRA Robotics

Данная компания занималась проектированием полностью автоматизированной фабрики по производству электрогрузовиков. Работая в ней, весь код я писал на C++.

В TRA я разработал класс C++ для снятия показания с силомоментного датчика DynPick и прописал для этого датчика интеграцию в корпоративный BlackBoard. Мой код поднимал топик и раз в заданное время писал в него считанные значения сил и моментов.

Для разработки цифровых двойников робототехнических ячеек я написал модуль UnrealEngine, который по заданным конфигурациям формировал с помощью бисплайнов непрерывные траектории.

Для проекта автономной тележки я разрабатывал кинематическую модель, проводил моделирование системы и тестировал простейшие алгоритмы управления.



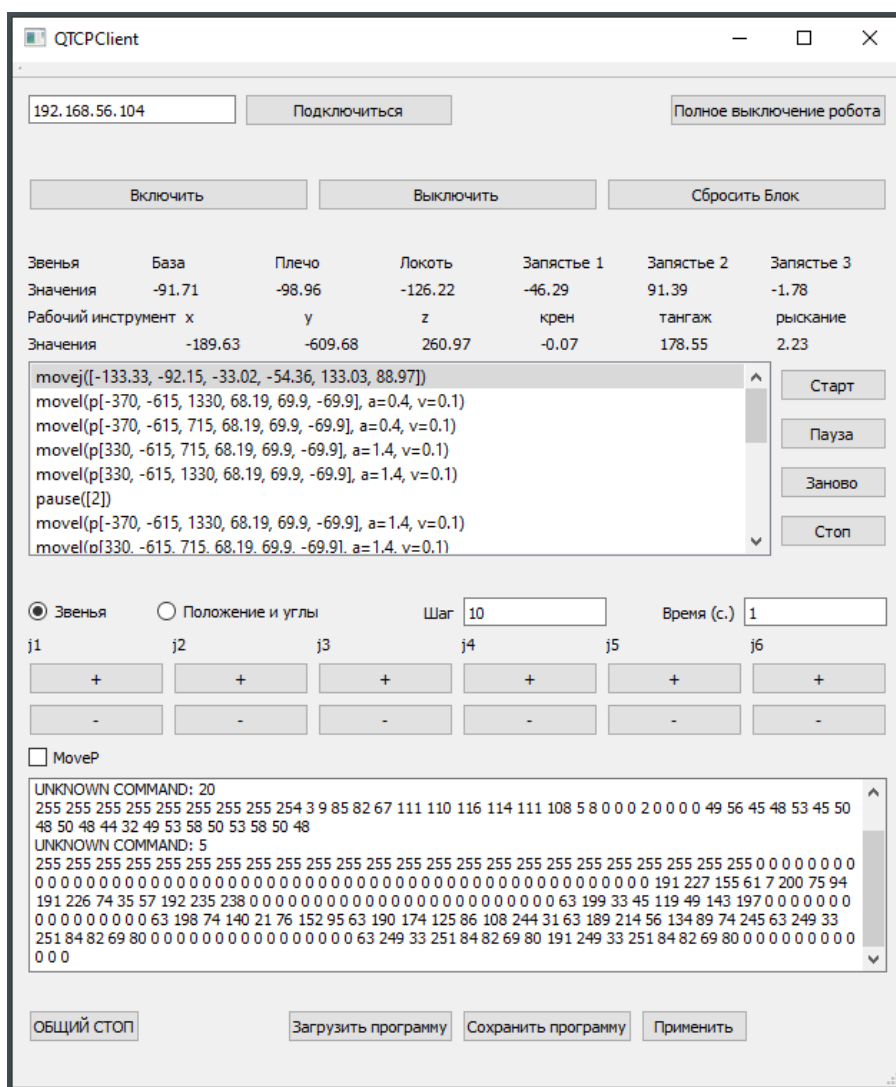
Работая в TRA Robotics, я разработал концепт нового типа складов. Он экономил место хранения в замкнутом объёме. Идея такого склада основана на принципе ханойской башни. Система хранения состоит из направляющих механизмов (зелёный, жёлтый и синий) и лифта(оранжевый).

Каждый предмет хранения помещался на специальный поддон с магнитами, а на направляющих были расположены движущиеся ленты, тоже с магнитами. В месте перехода с направляющих к лифту были помещены специальные отсекатели, которые не позволяли грузу уходить вниз. За счёт них груз заталкивался на лифт во время вращения лент.

С демонстрационным видео можно ознакомиться по [ссылке](#).

2.6. G-коды UR-10

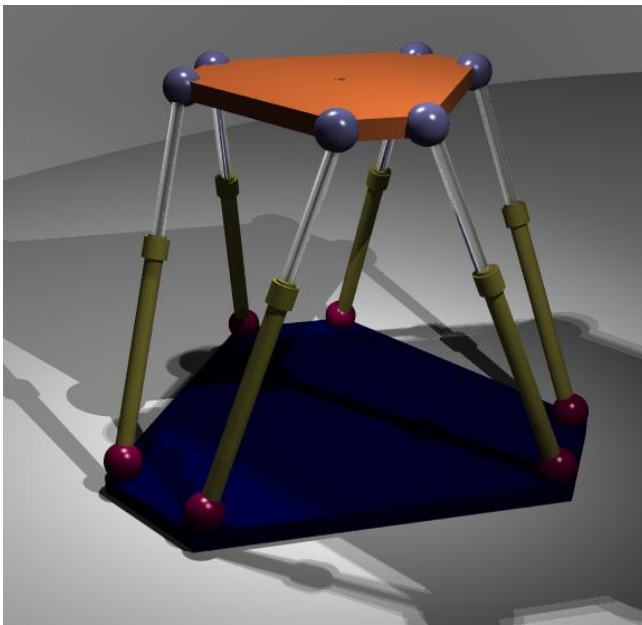
Для автоматизации сварочных работ по заказу Завода «Контакт» мною была написана программа для ОС Windows (QT, C++). Сварка осуществлялась с помощью робота UR-10. Этот робот может получать внешние команды управления по сети.



Разработанная программа позволяет управлять движением робота с помощью аналога G-кодов без использования пульта управления и встроенного языка, сохранять программы в текстовые файлы, и загружать их.

С инструкцией Вы можете ознакомиться по [ссылке](#).

2.7. Гексапод



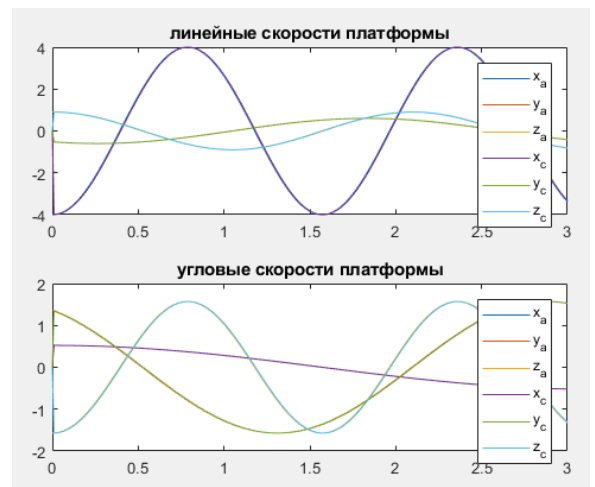
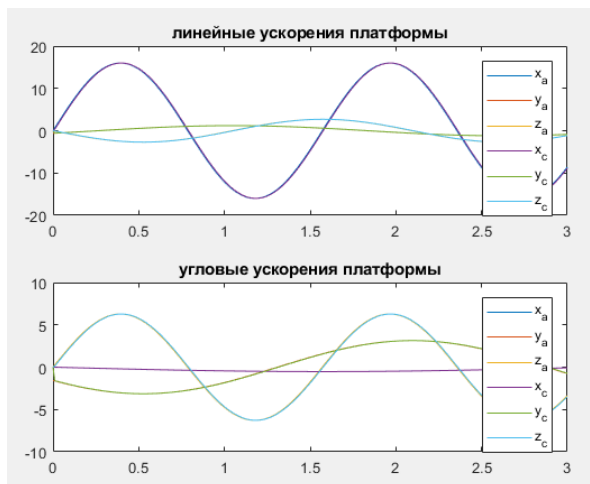
Для компании [Си Проект](#) я разработал модель гексапода – платформы Гью-Стюарта. Данная установка нужна для компенсации качки корабля.

В качестве имитации качки были выбраны три параметризованные синусоиды. По ним задавались углы ориентации платформы, а также угловые скорости и ускорения.

Необходимо было восстановить по ним положения, углы и скорости штоков.

Скорости и ускорения потребовалось свести в единый вектор угловой скорости. Для этого пришлось строить матрицы поворота вдоль каждой из осей и вычислять единый угол для композиции вращений.

Моделирование было выполнено с помощью Matlab.

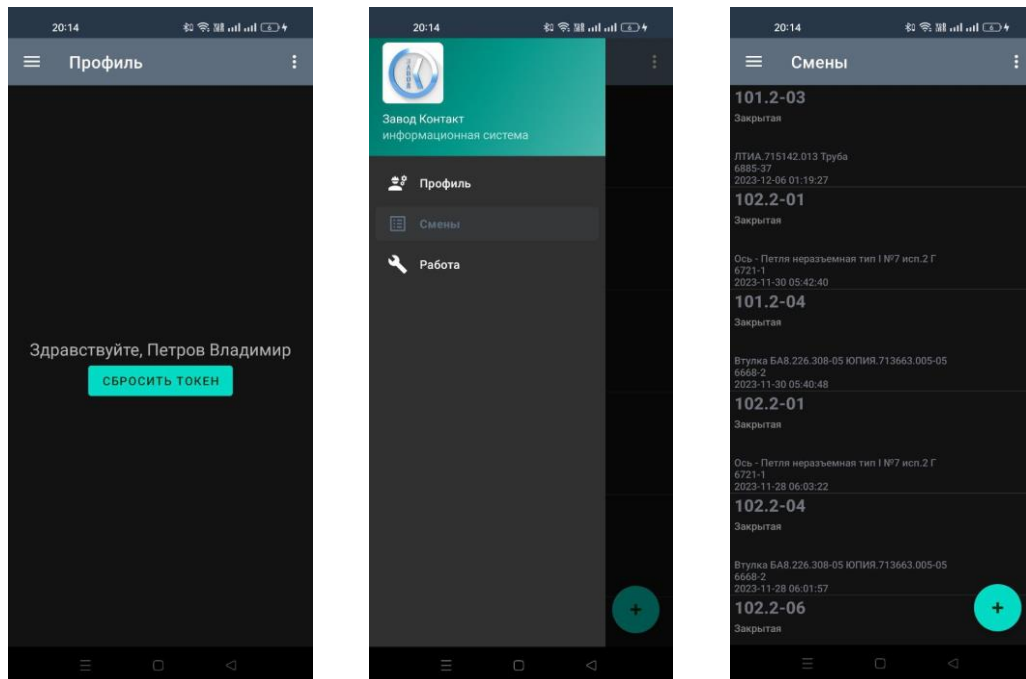


3. ERP Системы

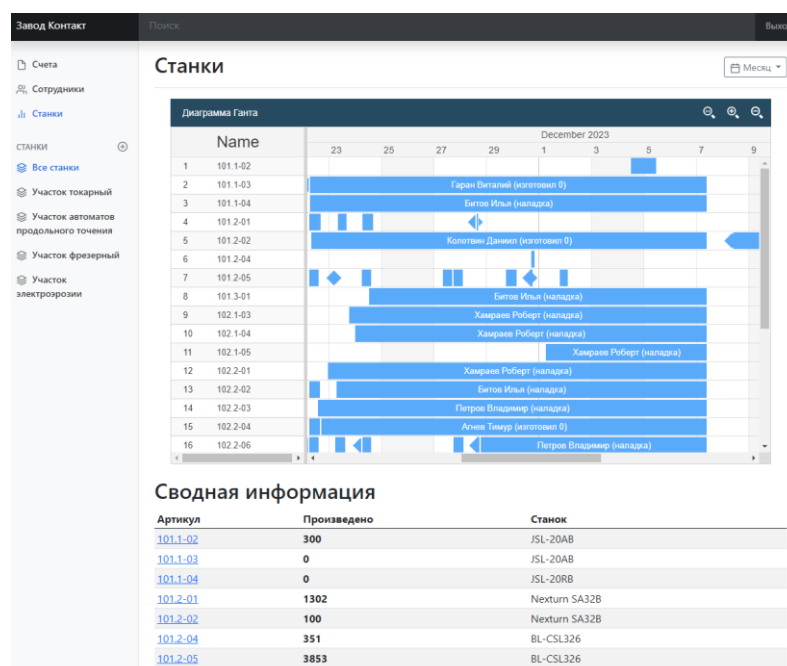
При росте объёмов следить за процессами на производстве становится всё сложнее. Эту проблему решают системы диспетчеризации – ERP системы.

3.1. Завод контакт

Для повышения эффективности производства на заводе «Контакт» я разработал систему диспетчеризации токарного цеха. Данное [предприятие](#) специализируется на металлообработке. Для операторов и наладчиков токарного цеха я написал android-приложение.



Менеджеры использовали веб-сайт. Вся информация хранится на веб-сервере на Django+Python. Для визуализации использую ibm-gantt (диаграммы Ганта) chart.js (графики) и bootstrap

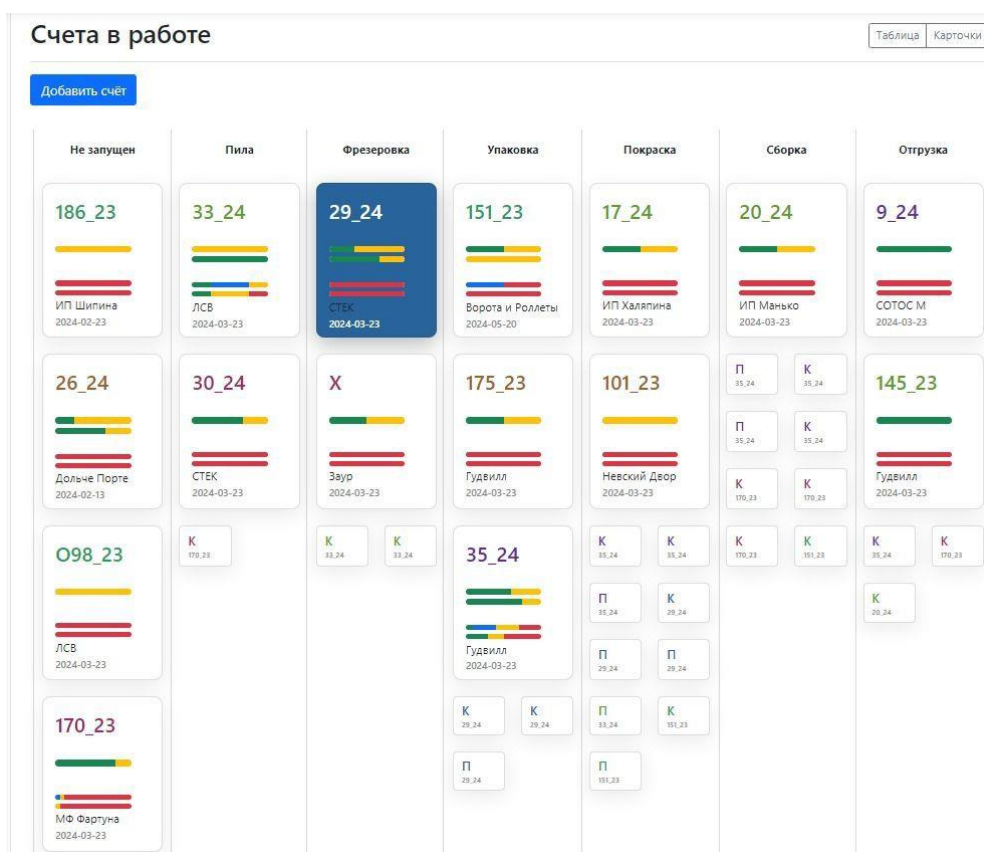


3.2. Компания «Форма»

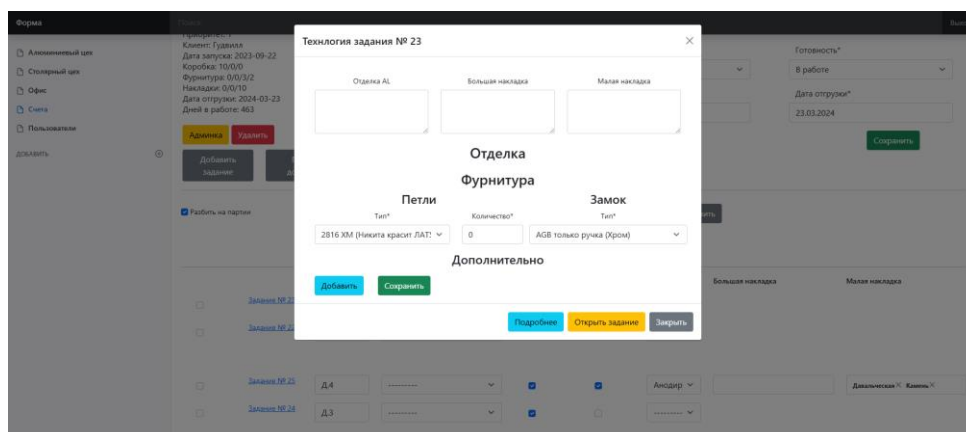
Данная компания занимается производством дверей. Производство разделено на три цеха: алюминиевый, столярный и сборочный. Сами двери состоят из двух основных элементов: «коробки» - фиксированная часть двери, «полотна» - подвижная створка.

Две части обрабатываются отдельно и должны поставляться в разное время. Окраска выполняется сторонними исполнителями. Перед сборкой нужно заказать фурнитуру или определить, что она есть на складе.

Каждый заказ состоял из нескольких дверей, часть из них имело только коробки, часть – только полотна. Чтобы следить за состояниями заказа я написал страницы с карточками. Большие карточки отображали заказы, а малые – полотна или коробки в том или ином заказе.



Каждая карточка находилась на самом раннем невыполненном этапе работы над одним из изделий в соответствующем заказе. Процент обработанных полотен и коробок отображался с помощью многоцветных progress-баров.



Каждую позицию в заказе технолог указывал, как именно должна быть обработана дверь на каждом из участков. Исполнители видели только задачи в своих цехах.

Исполнители отмечали, полотно или коробку какого заказа они обработали. Администратор мог изменять статус готовности любого объекта, а также видел, какой работник сколько работы выполнил.

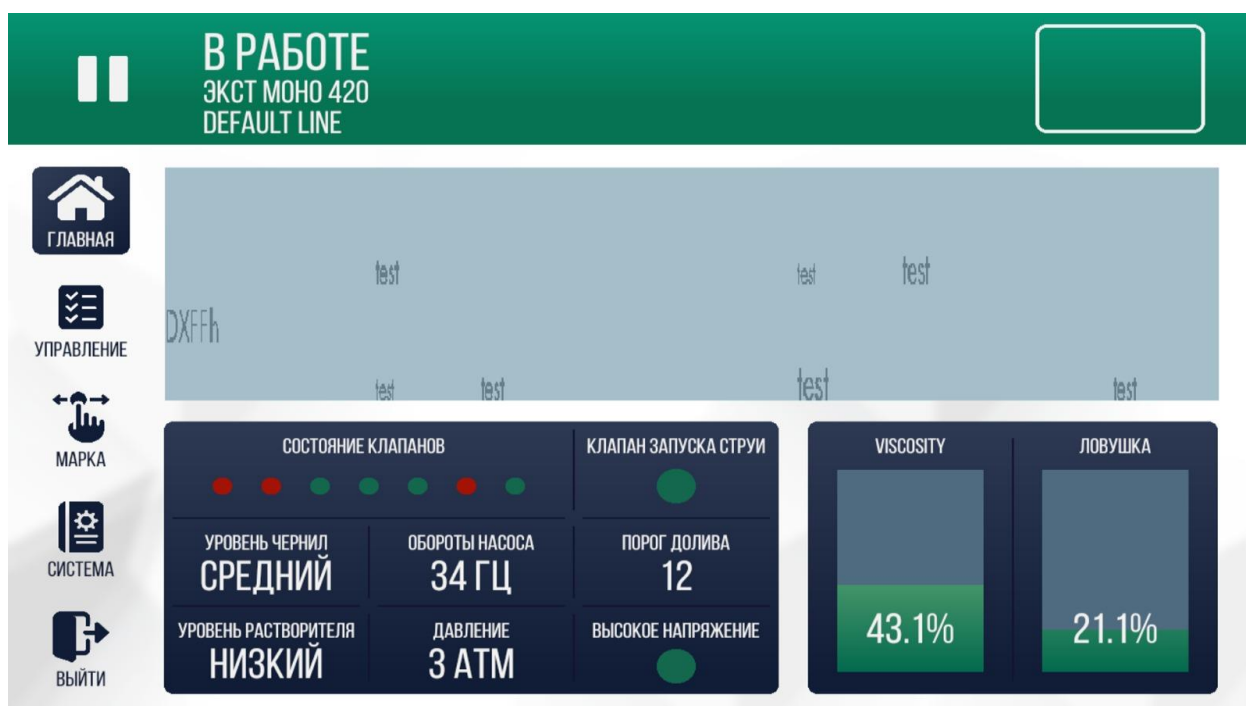
Остатки продукции отображались в разделе «Склад», раздел офиса был организован в той же логике, что цеха. Но их работой была закупка той или иной продукции, необходимой для изготовления дверей.

4. Встраиваемое ПО

В производстве интерактивных устройств часто вместо полноценного компьютера используются одноплатные компьютеры, такие как Raspberry Pi или обычные микропроцессоры.

4.1. Институт ЭКСТ

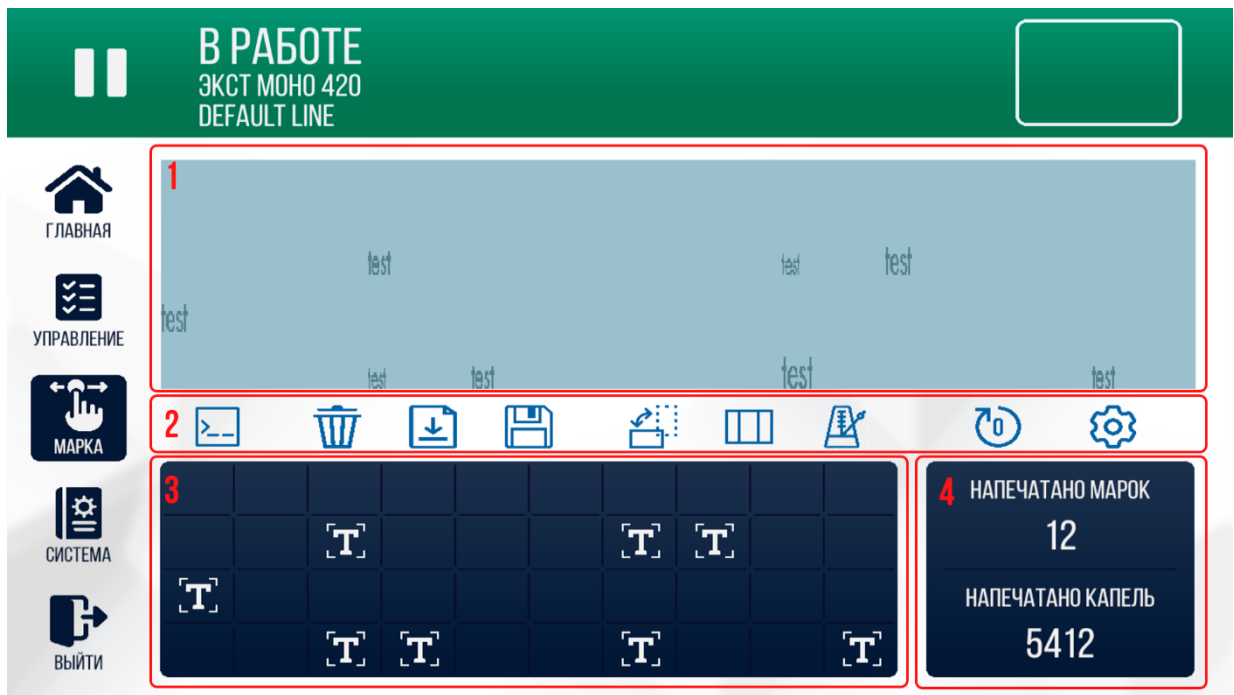
Для [института электро-каплеструйных технологий](#) я разработал программу управления принтером с графическим интерфейсом под одноплатный компьютер HelperBox T507 с помощью Qt Embedded (C++).



Электрокаплеструйный принтер работает за счёт отклонения капель чернил с помощью электромагнитного поля. Капли вылетают из сопла в сторону поверхности, поле отклоняет их в вертикальной оси.

Грубо говоря, отрисовка выполняется по одному «пикселю» (капле) вертикальными колонками. Когда на текущую колонку нанесён рисунок, сопло смещается на одну колонку и закрашивается следующая колонка согласно эскизу.

Верхний уровень разбивал все «пиксели» эскиза на блоки равного размера. Эти блоки называются марками. Видов марок было довольно много: простой текст, счётчик, текущее время или дата и т.д. Марки настраиваются в окне редактора марки:



С демо-видео редактора марок можно ознакомиться по [ссылке](#).

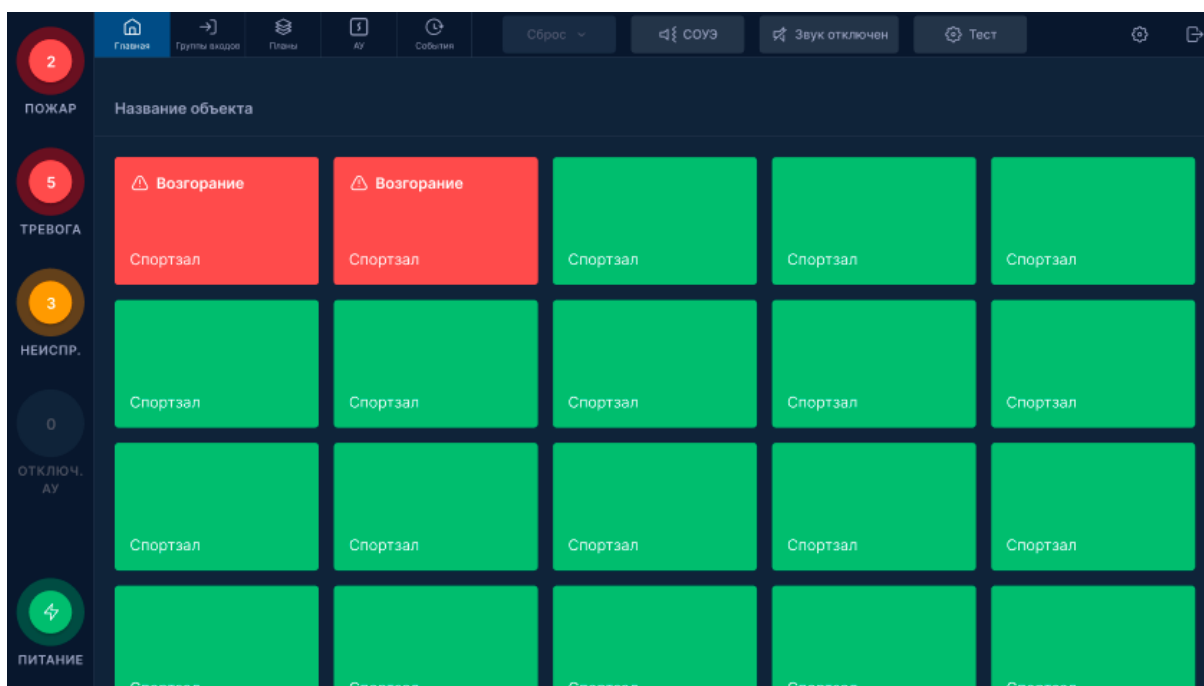
4.2. Зевс

Для компании [ООО «Зевс»](#) я разработал программу графического управления системой пожаротушения. Разработка велась под одноплатный компьютер Raspberry Pi с помощью QT (C++).

Программа отображала поэтажные планы помещения, каждый этаж был разбит на секции. Если в секции случалась авария, то она окрашивалась в красный цвет, все сообщения о тревогах выводились в правом углу.



Главный экран отображал плитки секций. Все секции с авариями окрашивались в красный. По каждой из секций можно было кликнуть и перейти на соответствующий план.



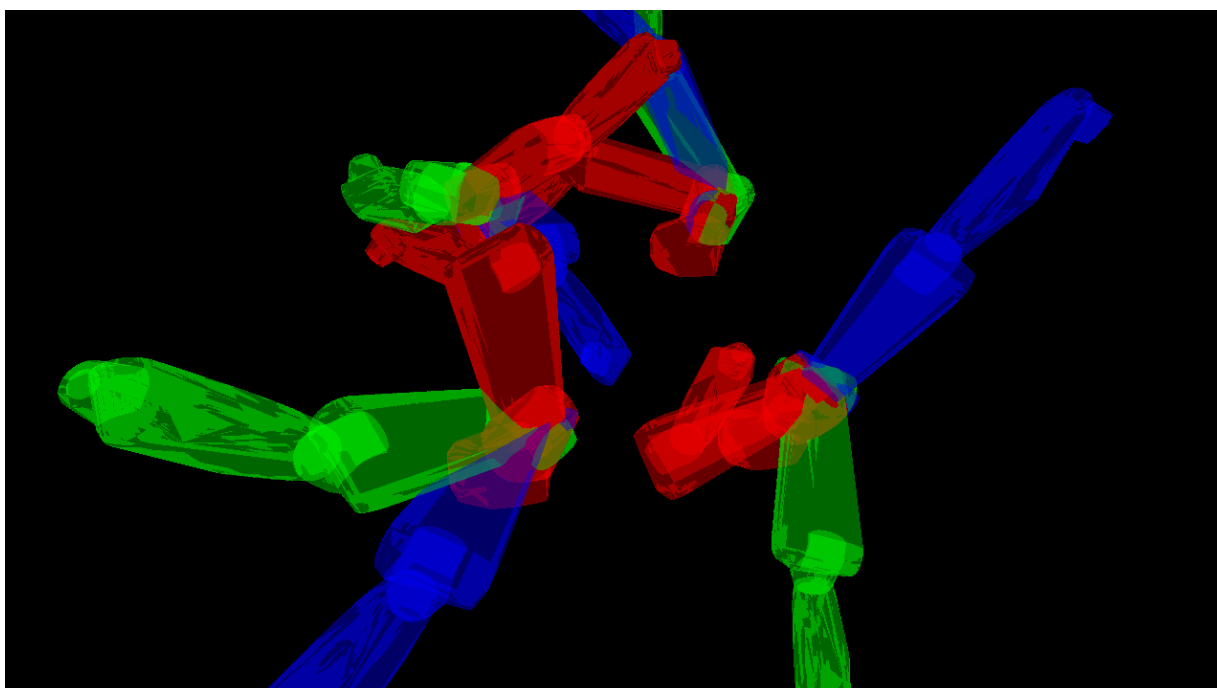
Непосредственное взаимодействие с системой датчиков было реализовано с помощью сервера на одноплатном компьютере Raspberry Pi. Моя программа обращалась к серверу с помощью API-запросов.

Планы этажей задавались с помощью конфигурационного файла.

5. OpenSource проекты

5.1. Buran Motion Planning Framework

Данный фреймворк – моя личная разработка, написан на C++, собирается в deb-пакет с помощью Jenkins. С обзорной статьёй можно ознакомиться [на хабре](#).



На текущий момент подавляющее большинство средств планирования движения работает по одному и тому же принципу: вся сцена описывается как один робот, после чего выполняется планирование на сетке.

У такого подхода есть две основных проблемы:

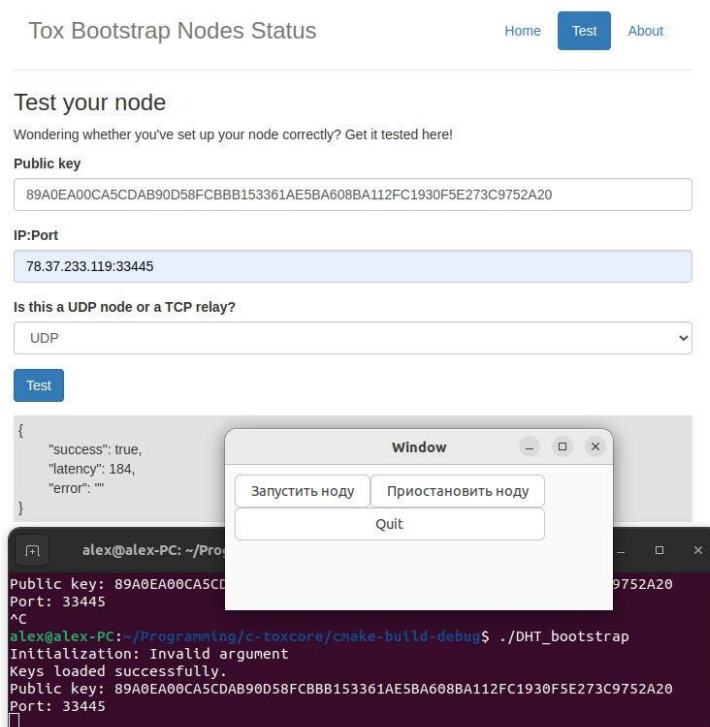
- 1) планирование на сетке гарантирует допустимость только состояний в её узлах, промежуточные никак не оцениваются и не проверяются.
- 2) для сцены из нескольких роботов размерность пространства планирования получается слишком большой (алгоритмическая сложность планирования растёт как показательная функция).

Данный фреймворк решает обе озвученные проблемы. С документацией фреймворка можно ознакомиться [здесь](#).

5.2. Графическая оболочка ноды P2P месседжера

P2P месседжеры не требуют центрального сервера, вместо серверов разворачиваются программы-ноды, каждая из которых может осуществлять функции сервера.

Целью работы была разработка на чистом C управляемой ноды, которую можно было бы приостанавливать снова запускать с помощью графического окна. Полный перезапуск консольного приложения заказчика не устраивал.

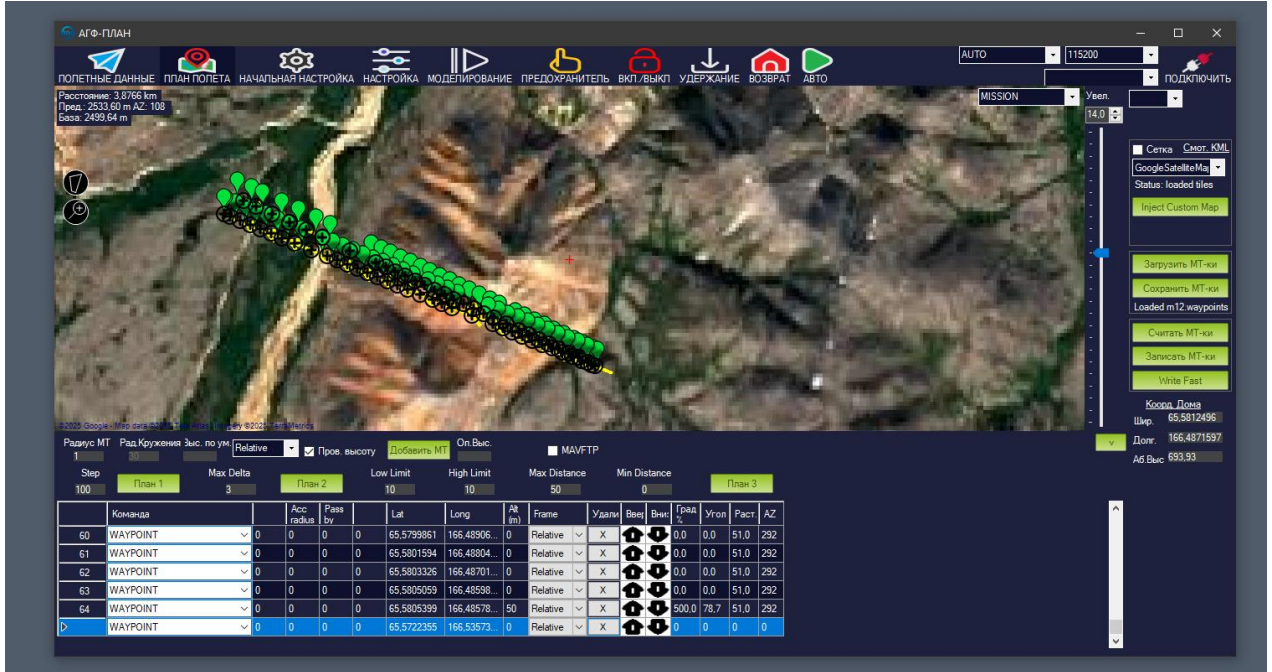


За основу заказчиком было выбрано openSource-решение [Tox](#). Мною оно было модернизировано под использование графической оболочки, оригинальное решение было сервисом и не предполагало какого-либо управления работой нодой.

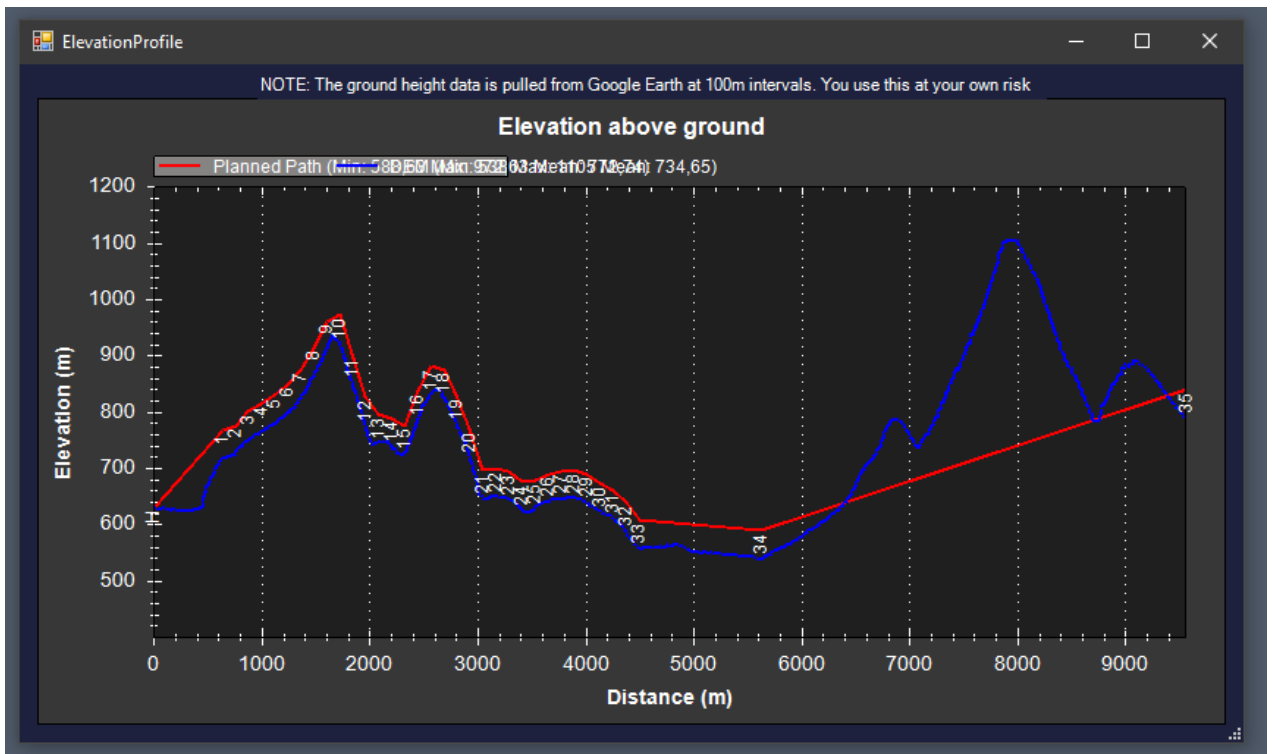
Основная сложность заключалась в малопопулярном инструменте сборки Autotools, а также в реализации графики с помощью библиотеки GTK (требование заказчика).

5.3. Планировщик полётных заданий Mission Planner

Для компании Аэрогеофизика я доработал программное обеспечение [MissionPlanner](#). Эта программа, написанная на C#, позволяет построить маршрут для беспилотника по заданным точкам. Каждая точка характеризуется высотой над уровнем моря, широтой и долготой. Далее программа загружается в сам беспилотник, и он отработывает заданный маршрут за счёт GPS-датчика и датчика высоты.

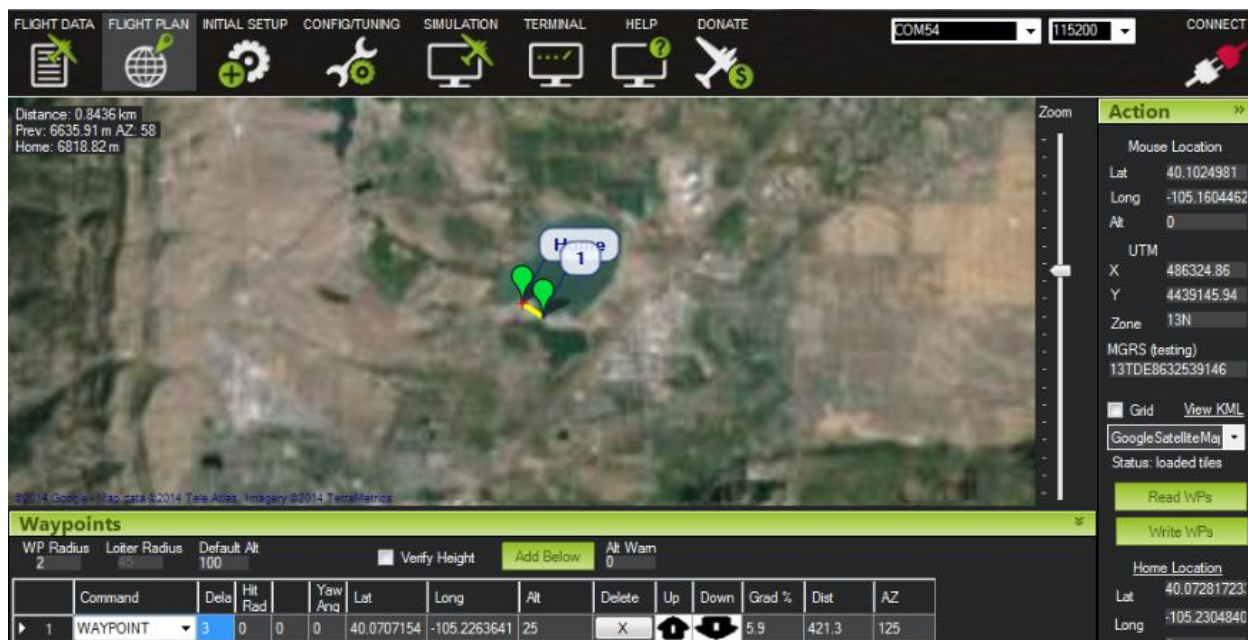


Основная проблема готового решения заключалась в том, что не было возможности автоматически расставлять точки с заданным расстоянием. Если опорные точки беспилотника расположены слишком близко друг к другу, тогда он не успевает разогнаться до крейсерской скорости. Если точки будут расположены слишком далеко друг от друга, тогда беспилотник врежется в рельеф.



Моё решение позволяло автоматизировать расстановку точек. Оператор ставил только опорные, все промежуточные точки софт выставлял автоматически. Причём он ставил точки не с равным шагом, а только когда очередная промежуточная точка на отрезках исходных опорных по высоте выходила за заданные границы. Тогда добавлялась новая точка, которая «выправляла» траекторию.

Также была немного видоизменена тема в сторону, исходный вид программы на скрине ниже



6. Обратная разработка

6.1. Лекала для плоттера

Для центра детейлинга автомобилей требовалось разработать программу, которая бы эмулировала режущий плоттер Graphtec FC8600 и сохраняла чертежи в формате DXF. На плоттере вырезались лекала. Существует два режима работы плоттеров: HP-GL и GP-GL. Первый – универсальный, второй используется только для плоттеров Graphtec.

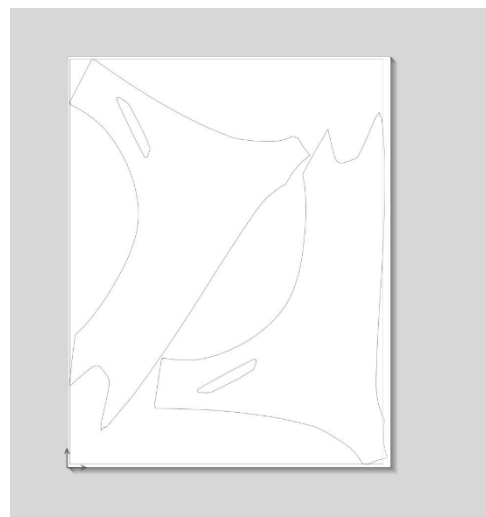
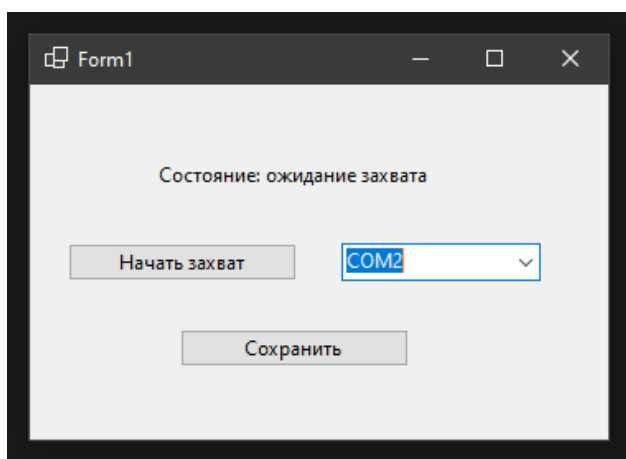
Проблема при работе с плоттером заключалась в том, что при обрезке лекал возникали ошибки, но посмотреть и исправить чертёж не представлялось возможным. Современные программы были полностью закрытыми и позволяли выполнять подготовку лекал только своими-неудобными средствами, либо импортировать готовый чертёж. Подготовленные программой лекала нельзя было отредактировать в распространённых и функциональных графических редакторах на подобии CorelDraw.



Изначальным заданием заказчика была разработка usb-эмулятора, который бы принимал команды от оригинальной программы плоттера и составлял чертёж. Однако, такие работы были очень объёмными и не соответствовали бюджету заказчика. Поэтому я предложил использовать для коммуникации COM-порт. В отличие от USB его реализация гораздо проще. Это просто последовательный интерфейс передачи информации. К тому же работа с COM-портом не обязательно требует написания драйверов.

Есть готовое программное решение виртуальных пар COM-портов. Два виртуальных порта замыкаются друг на друга, что позволяет подключить программу плоттера к программе составления чертежей так, как будет и первая, и вторая, подключены к COM-портам. На самом деле, они взаимодействуют друг с другом.

В ходе sniffing пакетов, выяснилось, что программа отправляет HP-GL команды в текстовом виде, иногда добавляя свои спец-символы. Дальше оставалось только написать программу, которая «слушает» COM-порт и интерпретирует полученные команды сначала в графические примитивы, а потом – в DXF.



6.2. Робот-ровер



Для компании «АКМ», подрядчика петербургского метростроя, я составил описание системы управления роботом-ровером, а также техническое задание на разработку платы управления и нашёл исполнителя. С видео работы робота можно ознакомиться по [ссылке](#).

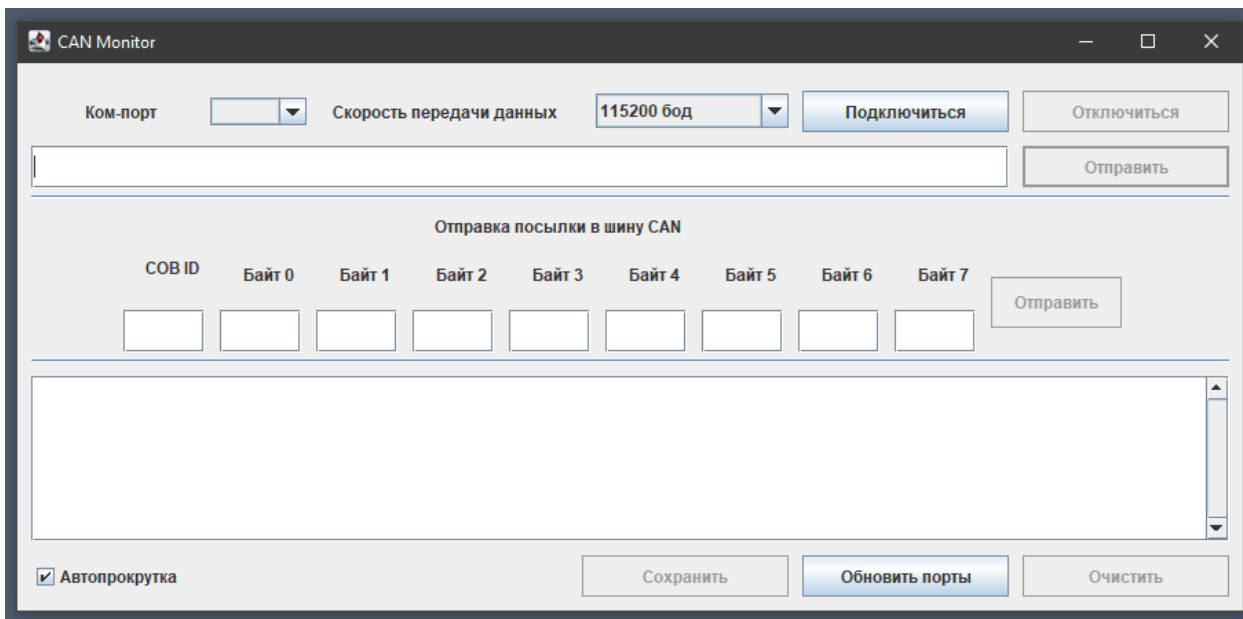
Робот предназначен для демонтажа железо-бетонных конструкций. Также его используют для разбивания кембрийских глин. Метро прокладывают зачастую как раз на уровне их залегания. Эти глины очень прочные, чтобы их извлечь, нужно их сначала разрушить: либо ручным отбойным молотком, либо таким роботом.



Для управления роботом использовался контроллер IFM CR0232. В связи с санкциями, стоимость приобретения контроллера стала превышать проектирование и изготовление своего собственного.

Робот построен на пропорциональных гидравлических клапанах, которые управляют цилиндрами.

Для составления комплекса команд я написал сниффер, уже под управлением компьютера, т.к. шину CAN можно было далеко протянуть, а непосредственное управление не требовало создания скриншотов. Команды накапливались и раскладывались с помощью адаптера (ESP32 и MCP2515) и программы-клиента, написанной на Java.



Когда пакеты появлялись в сети, они добавлялись в виде списка:

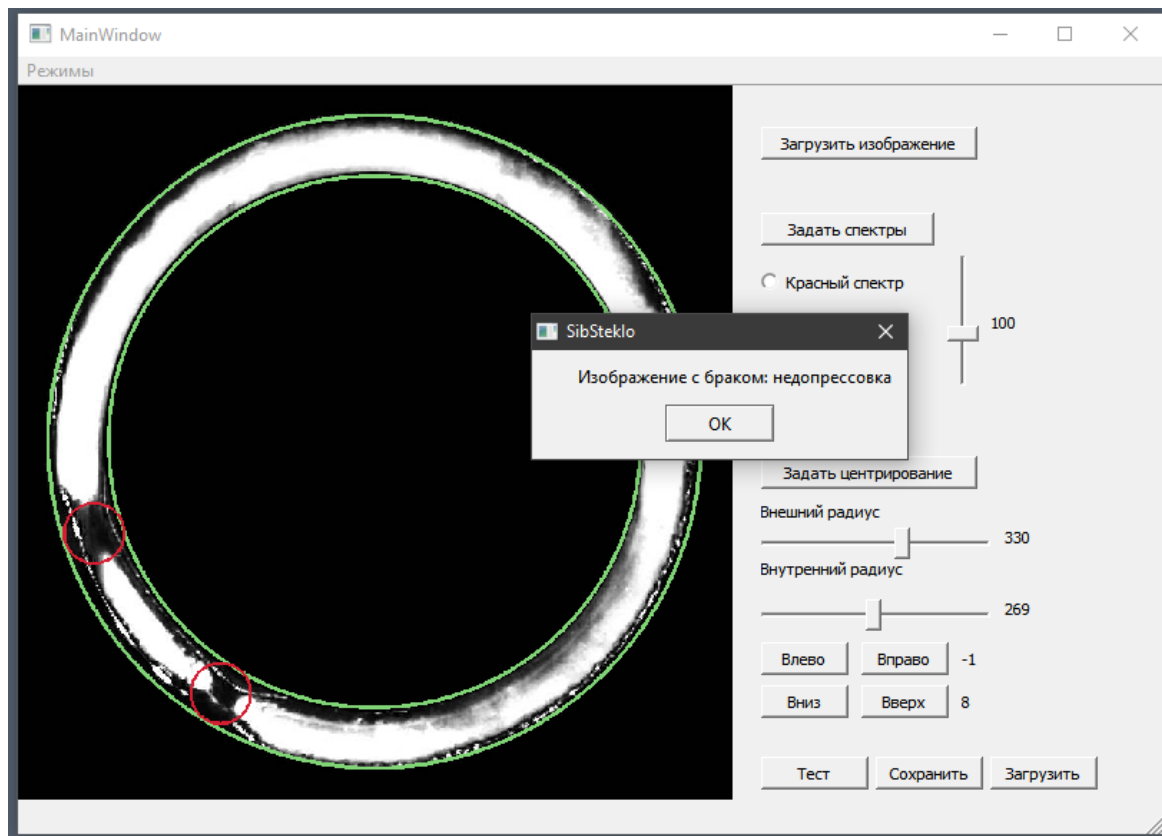
COB ID	Последнее время	Размер	Байты послыл...
246	14:43:50(8)	2	160 20
	14:43:57(12)	3	160 20 68
	14:43:58(5)	0	
15	14:43:51(3)	3	160 20 68
	14:43:51(1)	0	

После того, как полный комплект документации был готов, я нашёл исполнителя из Самары, который уже проектировал контроллеры с токовым ШИМ. Схемотехника этих каналов была самой сложной, остальные дополнения были не критичными.

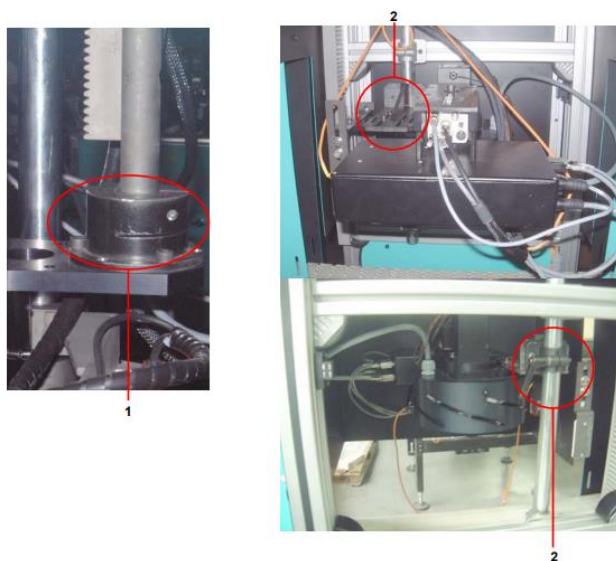
7. Прикладное программное обеспечение

7.1. Система детектирования брака стеклотары

Для компании «СибСтекло» необходимо было разработать программное обеспечение для детектирования брака в стеклотаре.



Принцип работы системы был следующий: по образцовому изображению настраиваются спектры так, чтобы наилучшим образом было различимо изображение, потом у каждой бутылки фотографируется горлышко. Фотография передаётся программе верхнего уровня



Интеграцию с камерами заказчик взял на себя, от меня требовалось разработать программу верхнего уровня на Qt и C++, которая бы позволяла по выбранной фотографии определить вид брака.

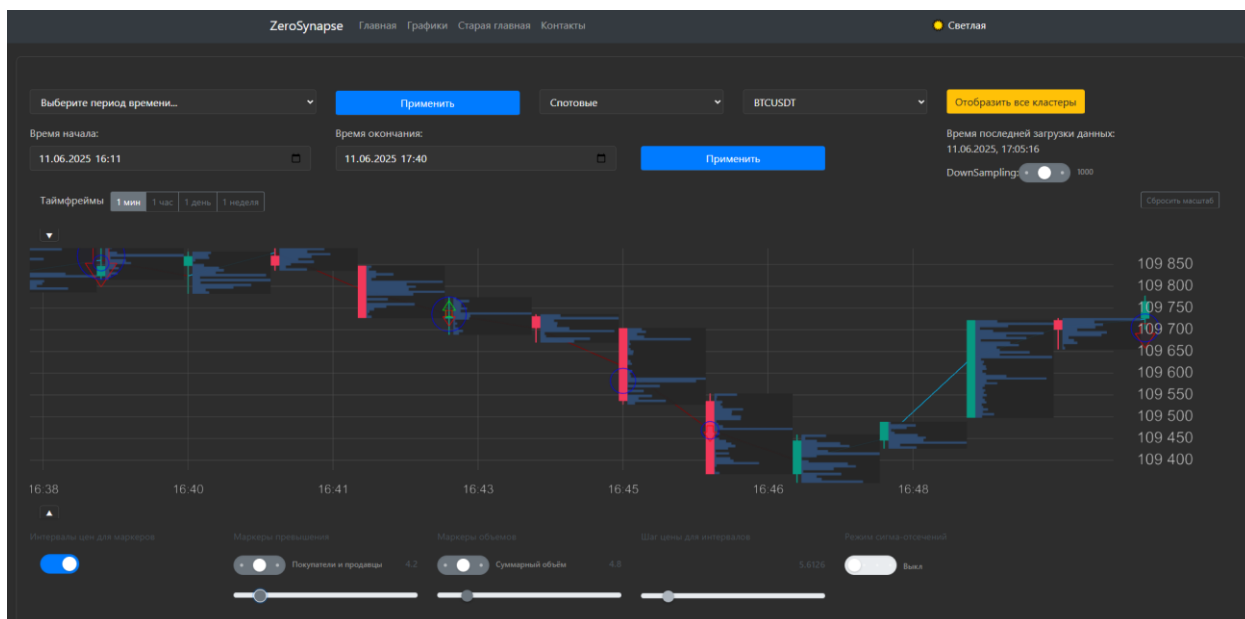
За счёт настройки двух окружностей: внешней и внутренней устанавливались границы зоны, которая должна была быть «светлой», остальные области должны были быть «тёмными».

Брака было три вида: пререпрессовка – соответствовала утоньшению горлышка, недопрессовка – появлялись утоньшения в зоне детектирования, нить в ободке – вне

зоны детектирования появлялась «светлая» зона. Для надёжности решения я реализовал его на обычных алгоритмах нечёткой проверки пикселей, без использования нейросетей. Также этот подход позволил снизить минимально необходимый датасет.

7.2. Аналитическая система торгов криптовалют

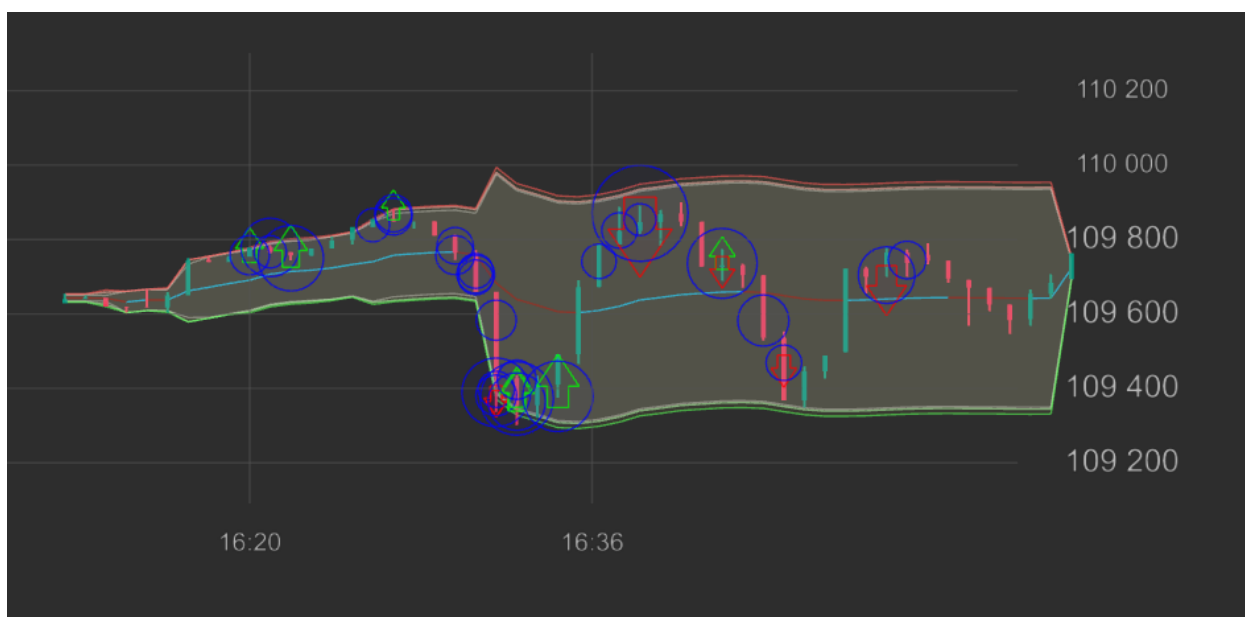
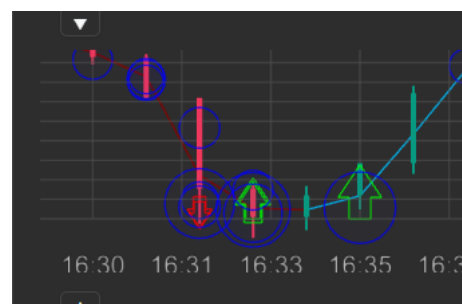
По заказу частного лица я разработал веб-систему для анализа бирж криптовалют



Основная идея платформы заключалась в том, что биржи криптовалют не регулируются, поэтому крупные интервенции определяют реально движение рынков. Эти крупные интервенции отображались в двух режимах: круги – это уровни цен, объёмы которых превышают заданный.

Стрелочки показывают превышения объёмов покупателей и продавцов по отдельности. Продавцы в методологии бирж – это те, кто обменивают криптовалюты на USDT, покупатели – кто обменивает USDT на криптовалюты.

Также для анализа система рассчитывала графики средневзвешенной и стандартных 70, 80 и 90 процентных сигма-отклонений. Это – зоны, в которые попадает соответственно 70, 80 и 90% всех сделок по объёму.

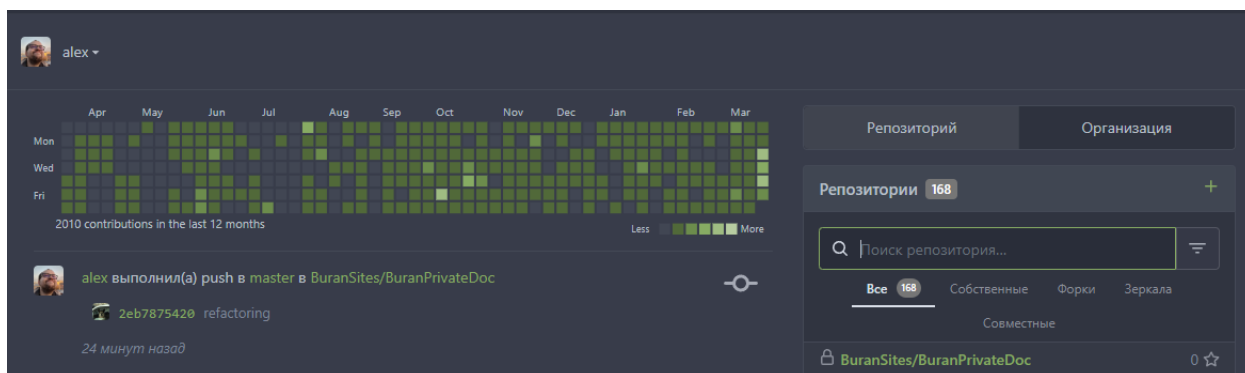


8. Дополнительная информация

8.1. Сетевое администрирование

У меня дома стоит сервер с гипервизором. На одной из виртуальных машин поднят обратный прокси nginx, он обрабатывает запросы к домену buran.center через внешний DNS-сервер.

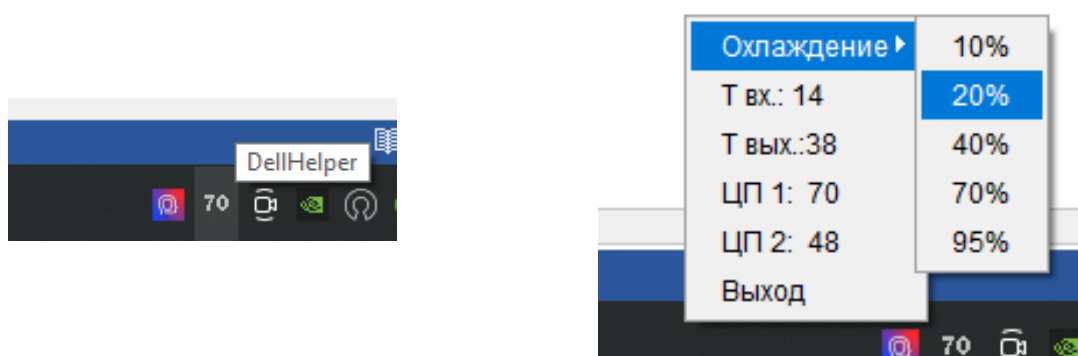
Свои сайты я развернул на нескольких виртуальных машинах. Все редактируемые мной сайты оборачиваются в докер-контейнер с помощью сервера Jenkins. Он, в свою очередь, в автоматическом режиме следит за изменениями в Gitea (git сервер с открытым исходным кодом).



Чтобы распределять запросы к этим сайтам на этой же виртуальной машине поднят Traefic, обёрнутый в докер-контейнер.

Также я развернул свой [maven сервер](#). Зависимости для него публикуются автоматически с помощью Jenkins. На отдельных машинах я развернул базу данных Postgerss и сервер облачного хранения Nextcloud.

Т.к. при вычислительных экспериментах сервер сильно греется, то для управления охлаждением сервера я написал небольшую программу в трей. Для управления использовался внутренний консольный инструмент DELL ipmitool. Программа в трее по таймеру запускает консольную команду запроса температуры и сохраняет ответ во внутренних переменных.



Также данная программа позволяет задавать мощность работы системы охлаждения в процентах.

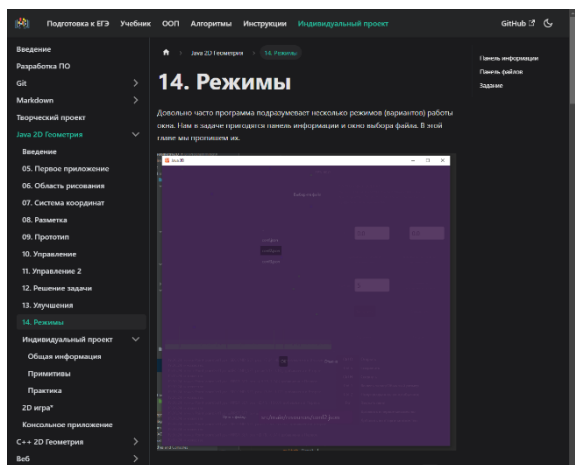
На одной из виртуальных машин у меня развёрнуто облако файлового хранилища Nextcloud.

8.2. Преподавательская деятельность

Несколько лет я вёл курсы программирования Android в рамках проекта [Samsung IT School](#).

Также я преподавал информатику в старших классах ФМЛ №239. Для этой работы я написал [сайт-учебник](#) с помощью фреймворка Docusaurus. Задания выполнялись на Java. Для проверки решений учащихся я использовал свой сервер Moodle.

Для агрегации результатов автоматических проверок заданий и перевода их в оценки я написал сайт с помощью фреймворков Django, Selenium и Bootstrap. Этот сайт также анализировал исходные коды учащихся на предмет списывания.



The screenshot shows a gradebook interface for the course "Информатика Проект Java АОК 10 класс 2022-2023". The table lists students and their scores across various topics. The columns represent different topics, and the rows represent individual students. The average score for each student is listed on the right side of the table.

Студент	01. Матрицы	02. Классы	03. Конструкторы	04. Нисходящая рекурсия	05. Страница	06. Абстракция	07. Передача параметров	08. Область видимости	09. Область видимости	10. Область видимости	11. Управление потоками	12. Управление потоками	13. Управление потоками	14. Режимы	Среднее
Абрамов Константин	5														0.3
Андреева Лейла	5	3	1	6	1		T	T	T	T	T	T	T	T	0.3
Андреев Григорий	0														0.0
Андреев Павел	4	1	1	6	0	0	0			T	T	T	T	T	0.3
Антонов Максим	4	3	1	2						T	T	T			0.3
Белтов Максим	5	2	1	4	1	7	T	T	T	T	T	T	T	T	0.6
Белкасов Абдулла	5	3	1	4	1	7	T	T	T	T	T	T	T	T	0.6
Билево Лев	5	3	1												0.5
Богданов Александр	3	1	4												0.4
Болгов Александр	5	3	1	6	1										0.9
Волкова Алина	5	3	1												0.5