

Разработка ПО управления системой неразрушающего контроля

Сатановский А.Д.

СПбГУ, Кафедра системного программирования

Научный руководитель: Луцив Д. В.

14 июня 2024 г.

- ❶ Введение в неразрушающий контроль: определение, цели и области применения
- ❷ Цель и задачи
- ❸ Требования, архитектура, модули системы
- ❹ Результаты и дальнейшая работа

Введение в неразрушающий контроль: определение, цели и области применения

Неразрушающий контроль – контроль надёжности основных рабочих свойств и параметров объекта или отдельных его элементов/узлов, не требующий выведения объекта из работы либо его демонтажа.

Целью использования неразрушающего контроля в промышленности является надёжное выявление опасных дефектов изделий.

Область применения: промышленность

Направление неразрушающего контроля при помощи радиографии развивается на базе [НИПК Электрон](#).

Целью работы является создание и реализация архитектуры ПО управления системой неразрушающего контроля. А именно, модулями системы позиционирования, системы безопасности, пульта управления системой позиционирования.

Задачи

- Создание архитектуры системы позиционирования, системы безопасности, пульта управления
- Выбор протоколов, библиотек
- Реализация, тестирование, рефакторинг
- Автоматизация сборки, тестирования, пакетирования

Функциональные требования

- При включении питания модуль должен автоматически запускаться и сигнализировать о готовности при помощи led индикации
- Модуль принимает команду от клиента, исполняет, формирует ответ и передает его клиенту.
- Если возникает ошибка, клиент должен получить ошибку в ответ
- Формирует статус системы и передает подключенным клиентам

Нефункциональные требования

- Изолированность. Каждый модуль располагается на Raspberry Pi
- Надежность. Модуль должен вести журнал событий
- Гибкость. Система настраивается через конфигурационные файлы
- Соответствие стандартам предприятия. Взаимодействие между клиент - сервером должно происходить по TCP в формате Gcode Marlin
- При подключении более одного клиента, каждый запрос должен обрабатываться в порядке очереди.

Выбор протоколов взаимодействия:

- Клиент - Сервер TCP + Gcode Marlin (Ethernet)
- Сервер - контроллер Modbus RTU (Serial Port RS485, Ethernet)

Пример Gcode команды для системы позиционирования на уровне TCP: **G1 F100 X50.5 Y-25.3 Z22.4**

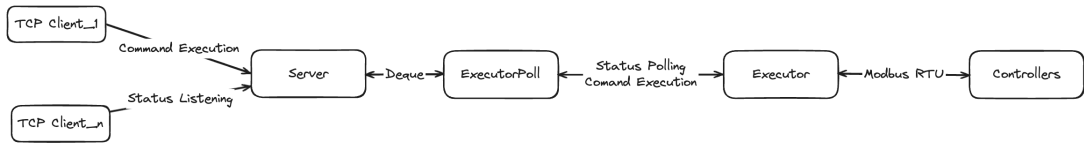
На уровне modbus rtu транслируется в байт пакеты (установка скорости 100 мм/мин для осей X, Y, Z, относительное перемещение оси X на 50.5 мм, Y на -25.3 мм, Z на 22.4 мм)

Raspberry Pi4 + Ubuntu 22.04. Разработка ведется на C++

Используемые библиотеки:

- <https://github.com/nlohmann/json> – для файлов конфигурации
- <https://github.com/childhoodisend/gpr> – парсер gcode
- <https://github.com/DFHack/clsocket> – sockets
- <https://gitlab.com/childhoodisend/qt-logger> – логгирование
- <https://github.com/google/googletest> unit-тестирование
- + переиспользуемые локальные библиотеки

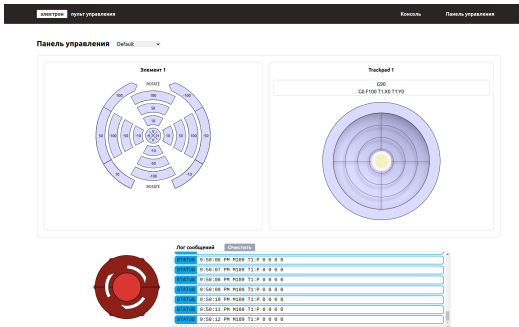
Архитектура взаимодействия



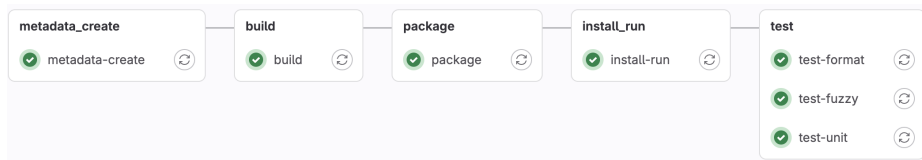
Пульт управления



(а) Аппаратный пульт управления системой позиционирования



(б) Программный пульт управления



(a) Пайплайн сборки, пакетирования, тестирования

- Создание версии пакета
- Сборка
- Пакетирование
- Проверка установки и запуска
- Тестирование на формат сообщений
- Нагрузочное тестирование
- Unit-тестирование

Результаты и дальнейшая работа

Результаты:

- Реализована система позиционирования. Находится в тестировании
- Написан веб-пульт для ручного тестирования. Он умеет отправлять команды и получать статус сервера. Тестировщику позволяет бесшовно общаться с модулями
- Реализован сервер пульта управления, пройдено ручное тестирование
- Развернут gitlab CI/CD
- Проведен рефакторинг системы позиционирования в модуле Server

Дальнейшая работа:

- Разработать алгоритм удержания точки интереса
- Разработать систему безопасности
- Провести дальнейший рефакторинг системы позиционирования
- Научиться тестировать под санитайзерами. Сделать покрытие дополнительными unit тестами. Увеличить процент покрытия
- Сделать стадию Deploy и поднять Registry