Разработка ПО управления системой неразрушающего контроля

Сатановский А.Д.

СПбГУ, Кафедра системного программирования

Научный руководитель: Луцив Д. В.

9 января 2024 г.

План презентации

1. Введение в неразрушающий контроль: определение, цели и области применения
2. Цель и задачи
3. Требования, архитектура, модули системы
4. 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.

Цель и задачи

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
- Создание архитектуры системы безопасности
- Создание архитектуры пульта управления системой позиционирования
- Выбор протоколов взаимодействия между клиентом - сервером - контроллером
- Выбор библиотек
- Реализация и тестирование

Требования

Функциональные требования

- При включении питания модуль должен автоматически запускаться
- При включении должна быть led индикация
- Модуль принимает команду от клиента, исполняет, формирует ответ и передает его клиенту
- Если возникает ошибка, клиент должен получить ошибку в ответ
- Формирует статус и передает клиенту

Требования

Нефункциональные требования

- Отказоустойчивость. Каждый модуль располагается на Raspberry Pi
- Безопасность. При подключении более одного клиента, каждый запрос должен обрабатываться в порядке очереди. Статус должен отправляться подключенным клиентам
- Надежность. Модуль должен вести журнал событий
- Гибкость. Система конфигурируется через файлы
- Взаимодействие между клиент - сервером должна происходить по TCP в формате Gcode Marlin

Разработка и архитектура

Выбор протоколов взаимодействия:

- Клиент - Сервер TCP + Gcode Marlin (Ethernet)
- Сервер - контроллер modbus rtu (Serial Port RS485, Ethernet, Moxa UPORT 1150 и др)

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, относительное перемещение оси X на 50.5 мм, Y на -25.3 мм, Z на 22.4 мм

Разработка и архитектура

Raspberry Pi4 + Ubuntu 22.04, arm64. Разработка ведется на C++14

Используемые библиотеки, удовлетворяющие требованиям:

- https://github.com/nlohmann/json – для файлов конфигурации
- https://github.com/childhoodisend/gpr – парсер Gcode
- https://github.com/DFHack/clsocket – TCP ip sockets
- https://gitlab.com/childhoodisend/qt-logger – логгирование

Архитектура

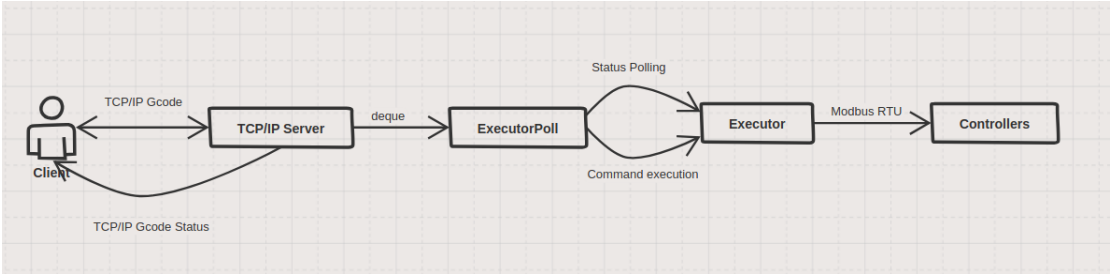
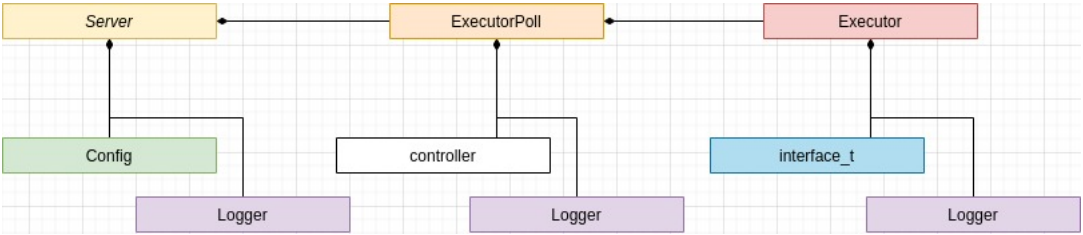


Диаграмма классов



Интерфейс подключения к контроллеру может быть произвольным. Для этого достаточно добавить поддержку нужного интерфейса в класс *interface_t*.

Результаты и дальнейшая работа

- Разработка ведется 2 года. Реализована система позиционирования, находится в тестировании
- Для тестирования написан веб-пульт. Он умеет отправлять команды и получать статус. Тестировщику позволяет бесшовно общаться с модулями
- Для сборки, пакетирования и тестирования развернут gitlab CI/CD
- Нужно разработать систему безопасности и пульт управления
- Нужно провести итерацию рефакторинга системы позиционирования
- Тестировать под санитайзерами, сделать покрытие доп тестами