Санкт–Петербургский государственный университет

Отчет по производственной практике “Разработка ПО управления системой неразрушающего контроля”

Уровень образования: **магистратура**
Направление: **02.04.03 “Математическое обеспечение и администрирование информационных систем ”**
Основная образовательная программа: **ВМ.5665.2019 “Математическое обеспечение и администрирование информационных систем”**

Студент 2 курса, группы 23.М04-мм:
Сатановский А. Д.

Научный руководитель:
Луцив Д.В.

г. Санкт – Петербург
2025

Содержание

1 Введение

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.

Область неразрушающего контроля достаточно большая. Направление неразрушающего контроля при помощи радиографии развивается на базе НИПК Электрон.

В рамках работы планируется реализовать и протестировать:

- Систему позиционирования – ПО, обеспечивающее перемещение аппаратного комплекса
- Систему безопасности – ПО, обеспечивающее безопасность аппаратного комплекса
- Пульт управления системой позиционирования – ПО, обеспечивающее возможность взаимодействия аппаратного пульта управления с системой позиционирования 11

2 Постановка цели и задач

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 для разработки
- Реализация архитектуры, логики
- Ручное и автоматизированне тестирование с использованием Gitlab CI/CD
- Пакетирование

3 Функциональные требования

Функциональные требования, предъявляющиеся системе:

- 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. Формат ошибки унифицирован
- Система формирует статус и передает подключенным клиентам – статус контроллеров, текущие позиции, статус выполнения команды

4 Нефункциональные требования

Данные нефункциональные требования должны быть выполнены при разработке:

- Отказоустойчивость. Каждый модуль располагается на Raspberry Pi [1]
- Надежность. Модуль должен вести журнал событий
- Гибкость. Система настраивается через конфигурационные файлы

- Соответствие стандартам предприятия. Взаимодействие между клиент –сервером должна происходить по TCP в формате Gcode Marlin [2]
- При подключении более одного клиента, каждый запрос должен обрабатываться в порядке очереди
- Автозапуск. Контроль запуска осуществляется при помощи supervisor

5 Архитектура и разработка

5.1 Аппаратно-программная конфигурация

Работа серверной части происходит на устройстве Raspberry Pi4, обладающем следующими характеристиками:

- Процессор: Cortex-A72 (ARM v8)
- ОЗУ: 8gb
- ОС: Ubuntu 22.04
- SSD: 64gb

5.2 Выбор библиотек

Разработка ведется на языке C++. Зафиксированы следующие библиотеки, удовлетворяющие вышеперечисленным требованиям:

- Для возможности конфигурации используется классическая библиотека **json**, позволяющая настраивать систему через файлы, что позволяет не компилировать программу повторно [3]
- Для работы с форматом **Gcode** используется форк **gpr** [4]. Форк доработан, тесты включены в общее тестирование
- Работу с сетью обеспечивает библиотека **clsocket** [5]
- Модуль логгирования –библиотека, основанная на фреймворке **QT** [6; 7]. Данная библиотека переиспользуется из других компонентов системы неразрушающего контроля
- Фреймворк Unit-тестирования **GTest/GMock** [8]

5.3 Протоколы

Для общения составлены протоколы Gcode команд и протокол обмена данными контроллера.

Пример команды из протокола обмена данными контроллера:

Команда	Данные master	Адрес, байт	Код функции	Данные, байт	Значения данных	CRC, байт	Данные slave	Адрес, байт	Код функции	Данные, байт	Значения данных	CRC, байт
Задать адрес устройства	Адрес	1	10 (0x0A)	3	1 байт: 0x00; 2 байт: 0x01; 3 байт: адрес от 0x01 до 0xF7	2	Установленный адрес	1	10 (0x0A)	3	1 байт: 0x00; 2 байт: 0x01; 3 байт: адрес от 0x01 до 0xF7	2

Рис. 1: Пример команды установки адреса контроллера

Пример команды из Gcode протокола:

Старт вращения со скоростью	**G0 F[Value] [Ось] [Value]** **F[Value]** – скорость перемещение mm/s. Устанавливается как номинальная скорость для всех осей, перечисленных в строке или до следующей F в этой строке. **[Ось]:** A, B, C, X, Y, Z Value: mm ***Usage:*** ***Command:*** *G0 F100 X20.1 – для оси X* ***Answer:*** *G0 F100 X20.1 – Подтверждение выполнения команды* **F[Value]** – скорость вращения град/с. Устанавливается как номинальная скорость для всех осей, перечисленных в строке или до следующей F в этой строке. Знак для оси определяет направление движения **[Ось]:** A, B, C, X, Y, Z Value: degree ***Usage:*** ***Command:*** *G0 F10 X20.1 – вокруг оси X* ***Answer:*** *G0 F10 X20.1 – Подтверждение выполнения команды*

Рис. 2: Пример Gcode команды линейного перемещения

5.4 Общая архитектура взаимодействия

Структурно архитектуру серверной части можно представить следующим образом:

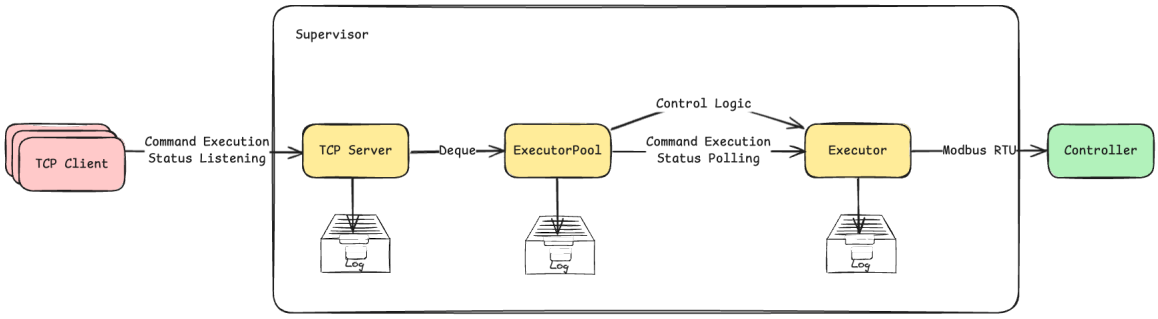


Рис. 3: **Архитектура взаимодействия серверной части**

Программно реализуются следующие модули:

- Модуль Server. Отвечает за обработку подключений клиентов по сети. Получает сообщения и передает их в модуль ExecutorPoll. Передает статус системы.
- Модуль ExecutorPoll. Валидирует сообщения (в соответствии с заданными regex), при помощи Executor переводит сообщение из формата Gcode в байт-пакет. Периодически выполняет запрос статуса контроллера (Status Polling). Выполняет дополнительную логику по контролю системы, например, удержание точки интереса.
- Модуль Executor. Инкапулирует логику перевода Gcode в байт-пакет. Реализует интерфейс взаимодействия с контроллерами –это может быть Serial Port RS485, Ethernet и другие. Пересылает байт-пакет

контроллерам и получает ответ (если предусмотрено). Проверяет наличие ошибки в соответствии с протоколом Modbus RTU. Выполняет проверку контрольной суммы CRC32.

В общем и целом, логика работы такой архитектуры сводится к следующим стадиям:

- Передать статус подключенному клиенту с заданным периодом
- Получить Gcode команды от клиента, выполнить валидацию в соответствии с форматом
- Выполнить парсинг команды, перевести в байт-пакет
- Передать байт-пакет контроллеру, получить ответ (если предусмотрено), обработать ответ. Если есть ошибка –обработать ошибку.
- Обратно сформировать Gcode из полученного байт-пакета
- Передать сформированный Gcode обратно клиенту в качестве ответа

При этом в системе позиционирования, безопасности и в пульте управления стадия валидации, парсинга, перевода в байт-пакет и формирования ответного Gcode может быть своя и зависит от набора Gcode команд в протоколе для конкретной системы.

Status Polling также может быть свой для вышеперечисленных систем. Таким образом, изменению подвержен только класс ExecutorPoll, где реализуется основная логика системы с специфичными протоколами.

Также архитектурно заложена возможность подменить протокол для общения с контроллерами. Для этого достаточно изменить "обертку реализующую перевод Gcode в байт-пакет.

Интерфейс подключения к контроллеру также может быть произвольным. Для этого достаточно добавить поддержку нужного интерфейса в внутренний класс, инкапсулирующий работу с интерфейсом.

Серверная часть запускается при помощи менеджера процессов [9], что позволяет гибко управлять системой и считывать логи. Используется UI часть из "коробки"для запуска и остановки сервера.

5.5 Пример выполнения Gcode команды линейного перемещения для системы позиционирования

Пусть необходимо выполнить относительное перемещение системы позиционирования по оси X на 50.5 мм, Y на 25.3 мм, Z на 22.4 со скоростью 100 мм/мин. На уровне TCP команда в формате Gcode выглядит как G1 F100 X50.5 Y-25.3 Z22.4, где $G1$ –команда линейного перемещения.

На уровне Modbus RTU команда транслируется в следующие байт пакеты:

- установить скорость 100 мм/мин для оси X
- установить скорость 100 мм/мин для оси Y
- установить скорость 100 мм/мин для оси Z
- относительное перемещение оси X на 50.5 мм
- относительное перемещение оси Y на -25.3 мм
- относительное перемещение оси Z на 22.4 мм

При возникновении ошибок на уровне Modbus RTU посылается команда на экстренную остановку. Экстренную остановку также можно произвести программно Gcode командой $M112$.

6 Формат ошибок

В серверах унифицирован формат ошибок, он состоит в следующем:

$$Error : [ErrorCode]"Description"$$

[ErrorCode] –(uint32_t) (4 байта в десятичном виде)

[byte 2], [byte 3] – Уникальное описание ошибки. Расшифровка передается разработчиком сервера. Расшифровка должна содержать информацию, необходимую и достаточную для понимания почему команда не была выполнена.

[byte 0] – Локализация ошибки:

- 1 – Сервер. Ошибка аппаратной работы сервера. (Пример: отсутствует соединение с ОУ, ошибки в аппаратной части обмена, CRC не совпало, количество байт не совпало).
- 2 – Оконечное устройство (ОУ). Оконечное устройство корректно отвечает, но не может выполнить команду из-за аппаратной ошибки такой как: ошибка драйвера, ошибка датчика, отсутствие калибровки, авария, выход в запрещенную зону.
- 3 – Не корректные данные, ОУ не может работать с данными параметрами. (Пример: параметры скорости/координат движения противоречат ограничениям из config-файла).
- 4 – Не корректный формат G-code команды. G-code составлен не корректно: команда не заканчивается символом конца строки, сообщение содержит больше одной команды "(G0 F10 X100 G28 M112").

[byte 1] – Приоритет ошибки:

- 0 – Информация, в этом случае необходимо добавлять информационное сообщение в виде текста.
- 1 – устранимая ошибка, команда не может быть выполнена, но эту ошибку можно устранить (например, такой код ошибки может произойти кода при включении системы безопасности аварийная блокировка не была снята или ОУ находится в процессе инициализации после включения).
- 2 – Предупреждение, такой код не приводит к потере работоспособности, но команда не может быть выполнена.
- 3 – Неустранимая ошибка, продолжение работы с ОУ невозможно.

Пример:

```
Input:
G0 F10 X100
Output:
G0 F10 X100 Error: 66306 ‘‘Permissible excess current value’’
где --66306 = 0x00010302 -> [[byte0 = 2] [byte1 = 3]] - универсальный код ошибки устройства,
где 2 - Ошибка в ОУ, 3 - Неустранимая ошибка.
([ErrorCode]) >> 16 = (0x00010302 >> 16) = 0x0001 = 1 - уникальный код ошибки.
```

7 Алгоритм удержания точки интереса для системы позиционирования

Точка интереса – это точка стола, которая находится в центре области видимости детектора на момент начала перемещения.

Для системы позиционирования комплекса AXI определены такие команды как G0 –линейное перемещение в точку, G6 старт вращения со скоростью для осей X, Y, Z, (стол) A(детектор), C(дуга).

Wx_0 и Wy_0 – это точка центра системы координат относительно которой вращается дуга и детектор. Данная точка считается центром мировых координат;

Tx_0 и Ty_0 – это точка центра системы координат стола. Для осей X и Y в системе позиционирования это точка с координатами 0, 0; Левее и ниже данной точки стол перемещаться не может;

PS_x и PS_y – это координаты центра стола системы позиционирования. Данные координаты соответствуют координатам текущего положения X и Y в системе позиционирования.

7.1 Функциональные требования

Разработан следующий алгоритм для системы позиционирования с следующими функциональными требованиями:

1. Включение функциональности удержания точки интереса, должно быть настраиваемым через файл конфигурации сервера.
2. Если функциональность удержания точки интереса включена в конфигурационном файле, то при изменении (перемещении) системы позиционирования с использованием команд G0 и/или G6 система должна выполнять коррекцию положения стола в горизонтальной плоскости
3. Корректировка положения стола при обработке команды G0. При получении команды G0 в которой нет изменения координат по осям X и/или Y, но есть изменения для любой другой оси (Z, A или C). Система позиционирования должна рассчитать требуемое перемещение стола по осям X и Y и выполнить их вместе с основным перемещением системы позиционирования.

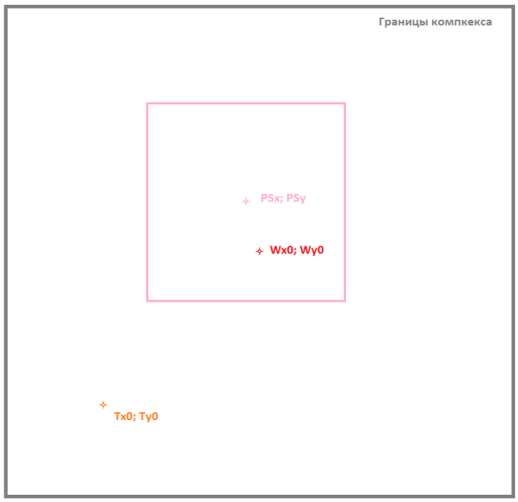


Рис. 4: **Системы координат комплекса AXI.**Ось Z для всех систем координат совпадает. И направлена снизу вверх.

4. Корректировка положения стола при обработке команды G6 в режиме следования стола за точкой интереса. При выполнении перемещения по осям Z, A и/или C необходимо с определённой периодичностью (периодичность должна настраиваться в файле конфигурации) выполнять перемещение стола (оси X и Y) таким образом, чтобы система позиционирования обеспечивала следование точки интереса за текущими перемещениями системы позиционирования.

   После остановки перемещения выполняемого по команде G6, необходимо обеспечить финальное позиционирование системы (перемещение стола по осям X и Y) таким образом, чтобы точка фокуса оказалась в центре области видимости детектора.

5. При завершении перемещения по команде G6, ответ о завершении выполнения команды должен отправляться только после завершения финальной корректировки положения стола по осям X b Y.

6. В ответе о завершении выполнения команд G0 и G6 (при успешном завершении) должны содержаться только те оси, которые были указаны в команде G0 и/или G6 при её получении сервером.

7. В случае получения команды G0 и/или G6, в которой указано перемещение хотя бы по одной из осей X и/или Y – корректировка положения стола для удержания точки интереса выполняться не должна.

8. Допускается реализация всей функциональности описанной в данных требования в виде отдельного микро сервиса, но с соблюдением всех изложенных в требованиях требований

9. Реализация всех требований, должна иметь набор тестов, покрывающих все требования для перемещения как по отдельным осям, так и по нескольким осям одновременно. Все тесты должны проверять корректность координат удерживаемой точки интереса после завершения перемещения.

8 Алгоритм контроля положения системы позиционирования в границах цифровых пределов

8.1 Функциональные требования

1. Включение функциональности контроля положения Системы позиционирования в границах цифровых пределов, должно быть настраиваемым через файл конфигурации сервера. Функциональность включена, если в файле конфигурации имеется ключ «ENABLE_CCDC» со значением true (или 1, –конкретная реализация типа значения выбирается разработчиком). Если данный флаг не указан или указан со значением, false (или 0), то функциональность – не используется.

2. Если функциональность контроля положения Системы позиционирования в границах цифровых пределов – включена, то при наступлении любого из событий: Получение команды M76 (Стоп вращения); Завершении выполнения команд G0 или G62 (Выполнение команды завершено, но подтверждение её не отправлено). Должен запускаться алгоритм контроля (проверки), что все оси находятся в границах, ограниченных цифровыми пределами для них.

3. Если функциональность контроля положения Системы позиционирования в границах цифровых пределов – включена, то при запуске сервера должен выполняться запрос значений минимального и максимального цифрового предела для каждой оси из контроллера. Если в контроллере не настроены цифровые пределы или полученные значения являются некорректными (например, максимальный предел меньше или равен минимальному) – сервер должен добавить соответствующую запись в файл журнала и завершать свою работы.

4. Если функциональность контроля положения Системы позиционирования в границах цифровых пределов – не включена, то при работе сервера никаких дополнительных проверок не должно выполняться.

9 Алгоритм повышения точности позиционирования

9.1 Функциональные требования

1. Включение функциональности повышения точности Системы позиционирования, должно быть настраиваемым через файл конфигурации сервера. Функциональность включена, если в файле конфигурации имеется ключ «ENABLE_INCREASED_POSITIONING_ACCURACY» со значением true (или 1, конкретная реализация типа значения выбирается разработчиком). Если данный флаг не указан или указан со значением, false (или 0), то функциональность – не используется.

2. Если функциональность повышения точности Системы позиционирования включена, то из конфигурационного файла для каждой из осей должна выполняться попытка загрузки значения опциональных ключей – «POSITIONING_ACCURACY» и «CORRECTION_SPEED». Если указанные выше ключи есть для конкретной оси Системы позиционирования, то их значения должны быть прочитаны и использоваться для проверки необходимости корректировки и установки скорости корректировки. Если указанных выше ключей нет в файле конфигурации, то по умолчанию должны использоваться следующие значения:

   POSITIONING_ACCURACY равно 0.01

   CORRECTION_SPEED равно:

   2 мм./сек. Для линейных осей (X, Y b Z);

   1 град./сек. Для осей вращения (A, B и C).

3. Если функциональность повышения точности позиционирования Системы позиционирования – включена, то при завершении выполнения команды G0 должен запускаться алгоритм проверки необходимости корректировки положения для каждой из осей и собственно корректировки, если это необходимо.

4. Алгоритм проверки необходимости корректировки для оси должен работать так:

   Текущее положение по оси сравнивается с ожидаемых положение (значением координаты, переданной в команде G0);

   Если разность превышает указанный порог (POSITIONING_ACCURACY), то запускается алгоритм корректировки положения по данной оси;

   Если разность НЕ превышает заданный порог, то корректировка не требуется;

5. Алгоритм корректировки для оси должен запускаться только один раз (после команды G0). Алгоритм должен выполнить повторное перемещение в ожидаемую позицию (значением координаты, переданной в команде G0), но с заданием при этом минимальной скорости (указанной в файле конфигурации или инициализированной значением по умолчанию). После выполнения корректировки, скорость перемещения для данной оси – должна быть восстановлена, чтобы при обработке следующей команды G0 в которой скорость не указывается, система перемещалась с ожидаемой (быстрой) скоростью

10 Файлы конфигурации

Ниже приведен пример конфигурации для системы позиционирования master_config.json

```
{
  "master": {
    "control_console": {
      "host": "",
      "port": 4002,
      "usage": "on"
```

```
    },
    "server": {
      "log_path": "/var/log/server",
      "host": "",
      "port": 4000,
      "status_port": 4001,
      "status_pooling_ms": 1000
    },
    "executor": {
      "log_path": "/var/log/server",
      "modbus-interface": {
        "port_devices": "/dev/ttyS0",
        "baud_rate": 9600,
        "parity": "N",
        "usage": "on"
      }
    },
    "executor_pool": {
      "log_path": "/var/log/server",
      "status_pooling_ms": 100
    },
    "enable_ccdc": "off",
    "enable_increased_positioning_accuracy": "off",
    "model_math_usage": "off",
    "model_math_tuning_ms": 100,
    "model_params": {
      "m_tableCenter": {
        "x": 254.688,
        "y": 271.377,
        "z": 10.0
      },
      "m_tableSize": {
        "w": 450.0,
        "h": 450.0
      },
      "m_arcRadius": 606,
      "m_fdd": 600.0,
      "m_detectorSize": {
        "w": 140.0,
        "h": 116.5
      },
      "m_height_above_table_mm": 8
    },
    "server_version": "v0.0.8",
    "modbus_protocol": "v18",
    "gcode_protocol": "v1.11"
  }
}
```

Далее приведен фрагмент для конфигурации работы с контроллерами. В системе Gcode контроллер обозначается при помощи тега (tag) и имени оси (X, Y, Z, A, B, C) T0.json:

```
{
    "slaves": {
        "A": {
            "brake_condition": "off",
            "conf_el": {
                "cyc_rev_rel_enc": 4096,
                "freq_poll_ms": 1,
                "inversion_absolute_enc": "yes",
                "inversion_increment_enc": "no",
```

```
                "inversion_pot": "no",
                "presence_absolute_enc": "no",
                "presence_brake": "no",
                "presence_end_sensors": "no",
                "presence_increment_enc": "no",
                "presence_pot": "yes"
            },
            "conf_motor": {
                "engine_inversion": "yes",
                "microstep_mode": "off",
                "mm_rev_frac_part": 1,
                "mm_rev_whole_part": 0,
                "mode_work_stepper": "micro_step",
                "pairs_bldc": 1,
                "type_motor": "bldc"
            },
            "correction_speed": 1,
            "displacement": "absolute",
            "home_position": 0.0,
            "home_position_status": 0,
            "limit": {
                "boost_speed_bldc": 0,
                "enable_position_limit": "yes",
                "enable_safe_zone": "no",
                "max_current": 0,
                "max_extremal_boost": 6000,
                "max_pos": 57,
                "max_speed_bldc": 3000,
                "min_current": 1,
                "min_pos": 1,
                "minimum_motor_shaft_frequency": 0,
                "speed_reduction_mm": 0,
                "speed_reduction_pers_max": 0
            },
            "main_init": {
                "baud_rate": 9600,
                "firmware_version": "none",
                "parity": "none",
                "slave_address": 10,
                "usage": "yes"
            },
            "position_accuracy": null,
            "speed_linear_default": 10,
            "speed_linear_move": 0.0
        }
    },
    "tag": "T0"
}
```


11 Состав пакета

Было принято решение что устанавливается при помощи bash скрипта install_run.sh и пакета server*.tar.gz.

Пример установочного скрипта запуска сервера с помощью supervisor install_run.sh, принимает путь к server*.tar.gz:

```
#!/bin/bash

# install & run
# ---------
# ARGUMENT $1 = prefix --path to server*.tar.gz
# run server with supervisorctl
```

```
set -ex

if [ -z $1 ] ; then
  echo "First parameter needed!" && exit 1;
fi

prefix=$1
echo $prefix
cd $prefix

# server stage:unpack
echo "server stage:unpack"
if [ -d "/builds/server" ]; then
  OLD_DIR_TMP=$(mktemp -d "${TMPDIR:-/builds/}server.XXXXXXXXXXX")
  echo "copying old server to ${OLD_DIR_TMP}"
  cp -r /builds/server/* ${OLD_DIR_TMP}
  rm -rf /builds/server/
fi
mkdir -p /builds/server/
tar -xzvf server*tar.gz -C /builds/server/ --strip-components=1
cd /builds/server/
# shellcheck disable=SC2010
cd "`ls | grep server*tar.gz | head -n1`"

echo `pwd`/lib/* >> /etc/ld.so.conf.d/libc.conf
cp `pwd`/lib/* /usr/local/lib
ldconfig

service supervisor stop
service supervisor start

# allow ports
echo "allow ports"
update-alternatives --list iptables
update-alternatives --set iptables /usr/sbin/iptables-legacy
ufw allow 4000
ufw allow 4001
ufw allow 4002

# mkdir log folder
mkdir -p /var/log/server
mkdir -p /var/log/gpio
touch /var/log/server/out.log /var/log/server/err.log /var/log/gpio/out.log /var/log/gpio/err.log

# copy supervisor server config and reread/update service
echo "copy supervisor server config and reread/update service"
cp server.conf /etc/supervisor/conf.d
cp gpio.conf /etc/supervisor/conf.d
supervisorctl reread
supervisorctl update

# server access rights
echo "server access rights"
chmod +x main_run.sh files/*/run.sh

# server stage:run
# echo "server stage:run"
supervisorctl restart server || true
```

server*.tar.gz пакет сервера содержит:

- bin/server – бинарный исполняемый файл сервера
- files/AXI(NDT) – файлы конфигурации для стендов
- files/master_config.json – файл конфигурации мастер сервера
- lib/* – библиотеки *.a
- run.sh – скрипт запуска. Принимает в качестве аргументов путь к bin/server и /files/AXI(NDT)
- server.conf – конфигурация сервера для запуска с помощью supervisor
- gpio.conf – конфигурация gpio порта для запуска с помощью supervisor
- gpio.sh – скрипт запуска для gpio. Устанавливает логический 0 на GPIO
- requirements.sh – скрипт для установки необходимых пакетов и доступов(требует подключение к интернету)
- set-network.sh – скрипт для конфигурации сети, принимает путь к network.yaml
- network.yaml – конфигурация сети, содержащая новый статический ip адресс, маску и gateway

12 Автоматизация

GitLab CI/CD (непрерывная интеграция/непрерывная доставка/развертывание) — это функция GitLab, которая позволяет применять методы DevOps для автоматической сборки, тестирования и развертывания кода. Это часть пакета GitLab, которая помогает автоматизировать процесс разработки программного обеспечения, делая его более быстрым и надежным.

Непрерывная интеграция (CI): включает автоматическое тестирование и интеграцию изменений в общий репозиторий несколько раз в день. Она гарантирует, что новые изменения не нарушат существующую функциональность приложения.

Непрерывная доставка (CD) гарантирует, что изменения кода автоматически готовятся к выпуску в производство. С помощью простого ручного утверждения разработчики могут развертывать свой код в производственных средах.

Непрерывное развертывание: расширяет непрерывную доставку для дальнейшей автоматизации процесса развертывания, позволяя развертывать изменения непосредственно в производстве без ручного утверждения.

Конвейеры GitLab CI/CD определяются в файле ‘.gitlab-ci.yml‘, который хранится в корне репозитория. Этот файл определяет этапы, задания и скрипты, которые необходимо выполнить. Runner, отдельный компонент GitLab, выбирает эти задания и выполняет их в назначенной среде.

Были подняты раннеры на архитектурах x86_64 и arm64 на сборочной Raspberry Pi и автоматизированы следующие стадии:

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

В общем, пайплайны выглядят следующим образом:

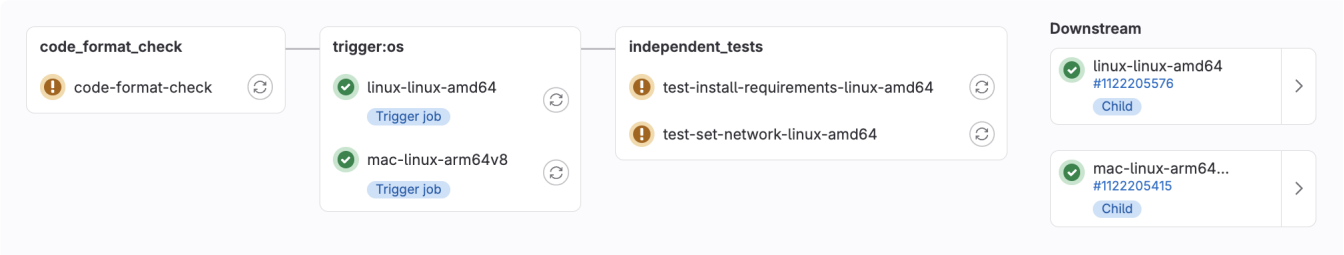


Рис. 5: **Пример Gitlab CI/CD пайплайна.** Сборка происходит в дочерних пайплайнах для архитектур **x86_64** и **arm64**.

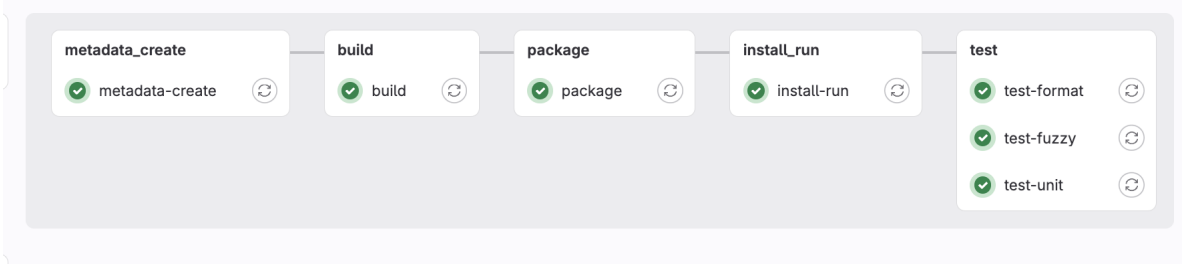


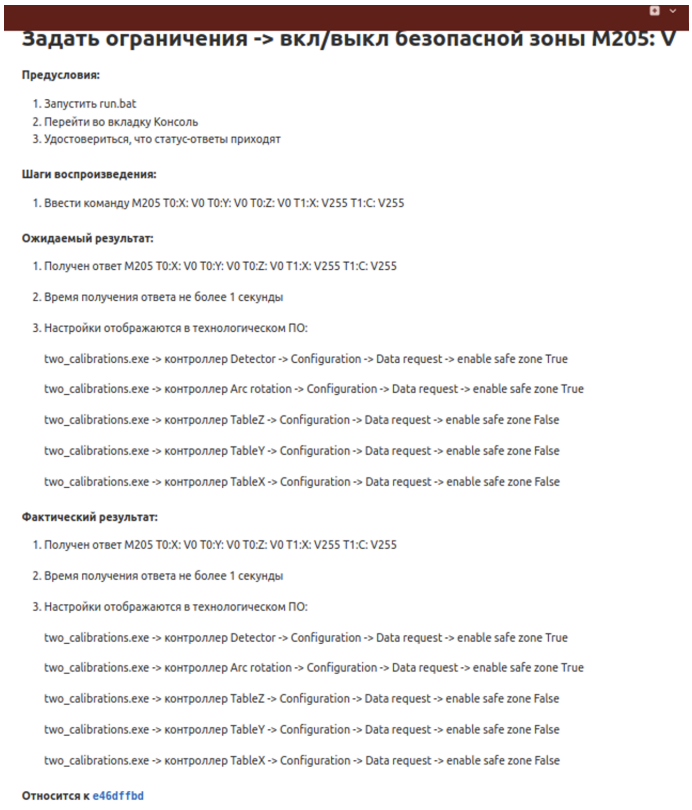
Рис. 6: **Пример дочернего пайплайна.**

13 Тестирование

13.1 Ручное тестирование

Ручное тестирование –это процесс поиска "багов"при помощи сил человека. Тестировщик имитирует реальные действия пользователя и старается охватить максимум функций системы и найти ошибки.

Для тестирования созданы тест-кейсы в локальном GitLab 7.



Задать ограничения -> вкл/выкл безопасной зоны M205: V

Предусловия:

1. Запустить run.bat
2. Перейти во вкладку Консоль
3. Удостовериться, что статус-ответы приходят

Шаги воспроизведения:

1. Ввести команду M205 T0:X: V0 T0:Y: V0 T0:Z: V0 T1:X: V255 T1:C: V255

Ожидаемый результат:

1. Получен ответ M205 T0:X: V0 T0:Y: V0 T0:Z: V0 T1:X: V255 T1:C: V255
2. Время получения ответа не более 1 секунды
3. Настройки отображаются в технологическом ПО:

two_calibrations.exe -> контроллер Detector -> Configuration -> Data request -> enable safe zone True

two_calibrations.exe -> контроллер Arc rotation -> Configuration -> Data request -> enable safe zone True

two_calibrations.exe -> контроллер TableZ -> Configuration -> Data request -> enable safe zone False

two_calibrations.exe -> контроллер TableY -> Configuration -> Data request -> enable safe zone False

two_calibrations.exe -> контроллер TableX -> Configuration -> Data request -> enable safe zone False

Фактический результат:

1. Получен ответ M205 T0:X: V0 T0:Y: V0 T0:Z: V0 T1:X: V255 T1:C: V255
2. Время получения ответа не более 1 секунды
3. Настройки отображаются в технологическом ПО:

two_calibrations.exe -> контроллер Detector -> Configuration -> Data request -> enable safe zone True

two_calibrations.exe -> контроллер Arc rotation -> Configuration -> Data request -> enable safe zone True

two_calibrations.exe -> контроллер TableZ -> Configuration -> Data request -> enable safe zone False

two_calibrations.exe -> контроллер TableY -> Configuration -> Data request -> enable safe zone False

two_calibrations.exe -> контроллер TableX -> Configuration -> Data request -> enable safe zone False

Относится к e46dffbd

Рис. 7: **Пример тест-кейса для ручного тестирования.**

13.2 Unit-тестирование

Unit-тестирование –это разновидность тестирования в программной разработке, которое заключается в проверке работоспособности отдельных функциональных модулей, процессов или частей кода приложения.

Unit-тестирование позволяет избежать ошибок или быстро исправить их при обновлении или дополнении ПО новыми компонентами, не тратя время на проверку программного обеспечения целиком.

Во всех проектах применяется фреймворк GoogleTest [8]. Можно сказать, что он является стандартом индустрии и имеет хорошо развитое сообщество.

В проекте присутствуют сдедующие тесты:

- test-algorithms –тесты самописных алгоритмов
- test-config-update –тесты на обновление конфигурации
- test-errros –тесты на формат ошибок
- test-executor-pool –тесты на класс ExecutorPool
- test-gcode-conv –тесты на создание байт-пакетов из Gcode
- test-gpr –тесты на парсинг Gcode
- test-regex –тесты на сопоставление Gcode в regex. Для каждого ПО regex свои, согласно протоколу общения в Gcode формате
- test-serial –тесты сериал порта, вычисление контрольной суммы CRC32

Тесты постоянно разрабатываются. Сейчас покрытие тестами оценивается в 40%.

13.3 Format-тестирование

В проекте написан скрипт для тестирование входящих сообщений от клиента. Сервер сопоставляет входящие сообщения с набором допустимых regex и присылает ответ.

Скрипт сравнивает ответ сервера с эталонными значениями.

Например, при сообщении с неправильным форматом ответ от сервера будет содержать ошибку

13.4 Fuzzy-тестирование

Fuzzy-тестирование –техника тестирования программного обеспечения, заключающаяся в передаче приложению на вход неправильных, неожиданных или случайных данных.

В проекте реализован скрипт для такого тестирования. На осовании regex для проверки формата сообщения, скрипт генерирует входные данные и пересылает их серверу. Ожидается, что сервер обработает входные данные с ошибкой на формат сообщения. Соединение при этом не рвется.

14 Результаты

За время разработки были выполнены следующие этапы:

- Реализована система позиционирования. Она постоянно тестируется. Для ручного тестирования описаны основные сценарии. Сцеарии занесены в локальный Gitlab
- Написан веб-пульт для ручного тестирования 10. Он умеет отправлять команды и получать статус сервера. Тестировщику позволяет бесшовно общаться с модулями. Для переключения между модулями достаточно указать нужный IP адрес.
- Реализован сервер пульта управления, пройдено ручное тестирование. Пульт подключался напрямую к системе позиционирования.
- Реализован сервер системы безопасности, пройдено ручное тестирование.
- Для сборки, пакетирования и тестирования развернут gitlab CI/CD [10]. На выходе получаем автоматически оттестированный пакет, который можно развернуть на стенде для дальнейшего ручного тестирования 5, 6.

- Проведен рефакторинг системы позиционирования в модуле Server. Создан базовый класс, куда вынесена общая функциональность, связанная с передачей и получением сообщений, поддержкой подключенных клиентов.
- Реализованы алгоритмы управления: удержания точки интереса, контроля положения системы позиционирования в границах цифровых пределов, повышения точности позиционирования
- Поднят отдельный регистри для Docker образов

15 Дальнейшая работа

В дальнейшем планируется:

- Проработать использования брокера сообщений для перехода от старой архитектуры 8 к новой 9
- Провести рефакторинг системы позиционирования. Собрать интерфейсную библиотеку для переиспользования в других серверах.
- Дальнейшая интеграция с ПО АРМ Оператора. ПО является главным управляющим элементом для разрабатываемых систем.
- Тестирование под санитайзерами. Сделать покрытие дополнительными unit тестами. Увеличить процент покрытия
- Провести замеры производительности
- Сделать Deploy на стенды для автоматического обновления
- Поднять Registry для хранения пакетов

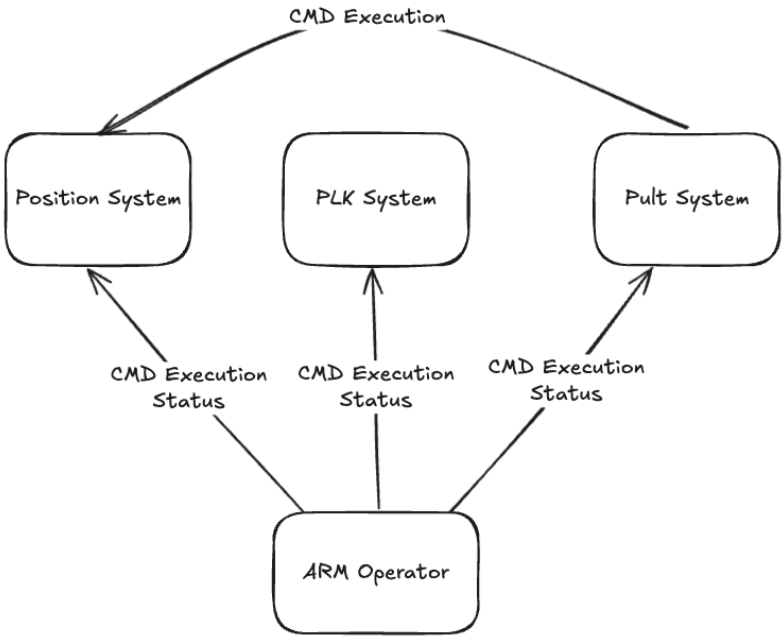


Рис. 8: Взаимодействие компонент комплекса в старой архитектуре

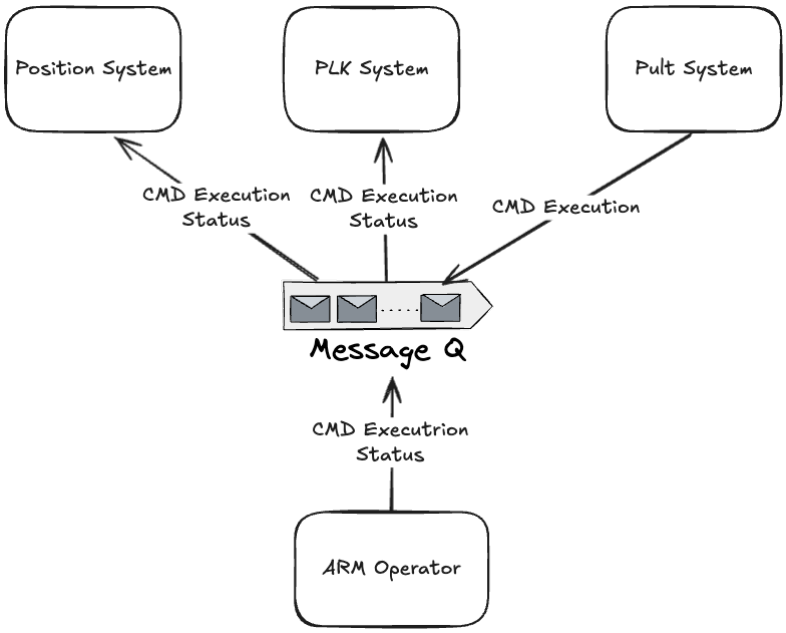


Рис. 9: Взаимодействие компонент комплекса в новой архитектуре

Список литературы

1. Raspberry Pi4. — Accessed: 2025-03-04. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.
2. Gcode Marlin Documentation. — Accessed: 2025-03-04. https://marlinfw.org/meta/gcode/.
3. Json library. — Accessed: 2025-03-04. https://github.com/nlohmann/json.
4. Gcode parsing library. — Accessed: 2025-03-04. https://github.com/childhoodisend/gpr.
5. Socket library. — Accessed: 2025-03-04. https://github.com/DFHack/clsocket.
6. Logger library. — Accessed: 2025-03-04. https://gitlab.com/childhoodisend/qt-logger.
7. Logger library. — Accessed: 2025-03-04. https://doc.qt.io/qt-5/.
8. GoogleTest Framework. — Accessed: 2025-03-04. https://github.com/google/googletest.
9. Supervisor. — Accessed: 2025-03-04. https://supervisord.org/.
10. Gitlab CI/CD documentation. — Accessed: 2025-03-04. https://docs.gitlab.com/ee/ci/.

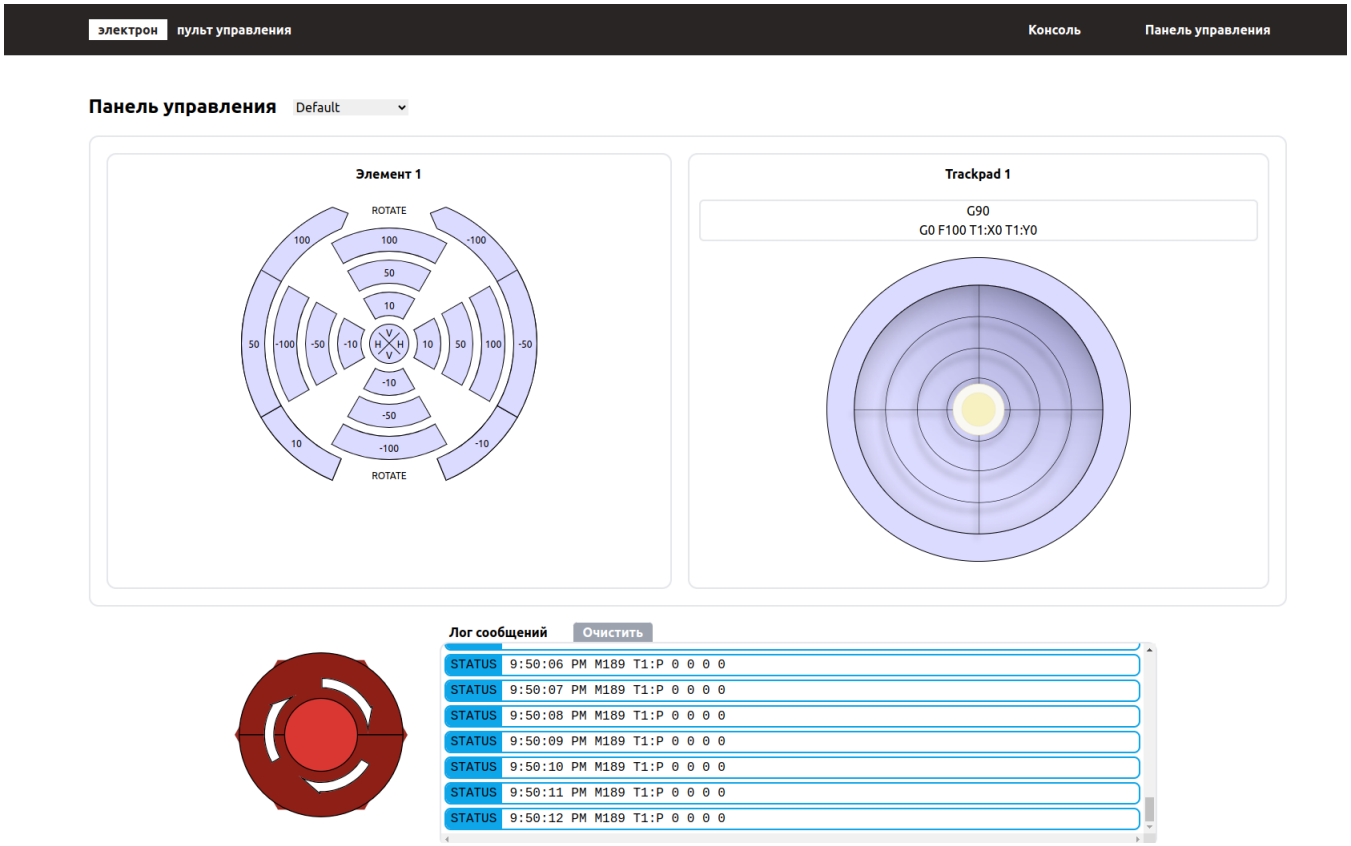


Рис. 10: **Стендовый пульт управления.** Эмулирует аппаратную часть. Способен слать команды и получать статус от системы позиционирования, безопасности, пульта управления.

Рис. 11: **Аппаратный пульт управления системой позиционирования.**