МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра Компьютерных Технологий

Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Программная инженерия и компьютерные науки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Родченко Анны Сергеевны

Тема работы:

Исследование подходов к разработке виртуальных лабораторных стендов в среде CoDeSys

«К защите допущена»
Зав. КафКТ,
д. т. н., доцент
Зюбин В. Е/……..
«31» мая 2022г.

Руководитель ВКР
д. т. н., доцент,
Зав. КафКТ
Зюбин В. Е. /……..
«31» мая 2022г.

Соруководитель ВКР
к. т. н., доцент,
преподаватель КафКТ
Розов А. С. /……..
«31» мая 2022г.

Новосибирск, 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра Компьютерных Технологий
(название кафедры)

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль): Программная инженерия и компьютерные науки

УТВЕРЖДАЮ

Зав. кафедрой Зюбин В. Е.
(фамилия, И., О.)

…….
(подпись)

«24» февраля 2022г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студентке Родченко Анне Сергеевне, группы 18208
(фамилия, имя, отчество, номер группы)

Тема Исследование подходов к разработке виртуальных лабораторных стендов в среде CoDeSys
(полное название темы выпускной квалификационной работы)

утверждена распоряжением проректора по учебной работе от 29 октября 2021г № 0029

скорректирована распоряжением проректора по учебной работе от 24 февраля 2022г № 0297

Срок сдачи студентом готовой работы май 2022 г.

Исходные данные (или цель работы): Исследование подходов к разработке виртуальных лабораторных стендов в среде CoDeSys

Структурные части работы: Анализ подходов к организации практических занятий по программированию ПЛК, формирование списка требований к ВЛС, Разработка архитектуры ВЛС, определение методики создания ВЛС, экспериментальная отработка подхода на модельной задаче.

Руководитель ВКР
Заведующий кафедрой,
д. т. н., доцент
Зюбин В. Е/……..
«24» февраля 2022г.

Задание принял к исполнению
Родченко А. С./……..
«24» февраля 2022г.

Соруководитель ВКР
ст. преподаватель,
преподаватель КафКТ
Розов А. С. /……..
«24» февраля 2022г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В нашу жизнь постоянно проникают различные киберфизические системы. Современный человек редко осознает, что половина его ежедневных действий (наподобие езды в лифте, использования стиральной машины или бесконтактного санитайзера) могла бы быть попросту невозможна без должного развития программируемых контроллеров. Но куда более важную роль играют системы автоматизации в производственной промышленности, без которых вся отрасль, если и не встанет, то точно замедляется в десятки раз.

Столь важная часть современного общества требует подготовки специализированных кадров, разбирающихся в основах работы киберфизических систем и, в частности, программируемых контроллеров. К сожалению, из-за ряда проблем с организацией практических занятий (для получения необходимых навыков) на реальных объектах управления, часто подготовка специалистов сводится к получению только лишь теоретических знаний. Решением данной ситуации является введение в практикум виртуальных лабораторных стендов (ВЛС) взамен реальных объектов.

Вопросы управления промышленными объектами посредством программируемых логических контроллеров рассматриваются стандартом IEC 61131. Один из недавних результатов ИАиЭ СО РАН является процесс-ориентированный язык poST, предназначенный для спецификации управляющих программ в рамках концепции IEC 61131-3.

Язык poST изучается в курсе НГУ “Процесс-ориентированное программирование”. Ранее лабораторные работы для данного курса проводились на базе концепции виртуального объекта управления и реализованы в среде LabVIEW. В качестве языка программирования использовался Си-подобный язык Reflex, не используемый при программировании ПЛК. В связи с появлением языка poST и переориентацией исследований на IEC 61131-3 концепцию, более привлекательной выглядит

организация лабораторных работ на базе одного из существующих IDE IEC 61131-3.

Цель работы – Исследование подходов к разработке виртуальных лабораторных стендов в среде CODESYS.

В рамках поставленной цели определены следующие **задачи**:

1. Анализ подходов к организации практических занятий по программированию ПЛК
2. Формирование списка требований к разрабатываемому подходу
3. Разработка архитектуры ВЛС
4. Определение методики и методов создания ВЛС
5. Определение модельной задачи и экспериментальная отработка подхода на ней

Новизна работы представляет собой рассмотрение способа организации виртуальных лабораторных стендов для на языке poST в среде CODESYS.

Практическая ценность данной работы состоит в сокращении трудоемкости организации учебного процесса и формирование навыков работы с языками IEC 61131-3 и процесс-ориентированным языком poST.

Структура работы. Работа состоит из введения, 2 глав и

заключения. В первой главе рассматриваются вопросы организации практических занятий с использованием лабораторных стендов, обзор существующих средств для создания ВЛС и возможности по визуализации в среде CODESYS. Во второй главе выделены требования, описана архитектура разрабатываемого стенда и подход к его реализации с описанием проделанной работы. В заключении сформулированы основные выводы произведенной работы.

ОСНОВНАЯ ЧАСТЬ

1. Обзор и анализ

Вне зависимости от объекта изучения, наиболее успешным процесс познания становится тогда, когда его теоретическая составляющая находит свое применение на практике. Для наилучшего усвоения материала и на сегодняшний день остается актуальным принцип наглядности, который способен получить в условиях современных информационных технологий новый виток своего развития.

Во многих дисциплинах неотъемлемой частью обучающей программы являются занятия с использованием лабораторных стендов, которые позволяют проводить работы близкие к реальной производственной деятельности. Это придает процессу обучения особый стиль и вызывает больший интерес у студентов. В последнее время все чаще всплывает понятие «виртуальный стенд». Под виртуальным стендом понимается учебно-практический (лабораторный) стенд, который способствует укреплению теоретических знаний студентов, приобретению ими необходимых навыков по определенному направлению посредством компьютерных программ и технологий.

В отличии от обычных материальных лабораторных стендов, виртуальные стенды имеют ряд существенных преимуществ:

- Уменьшение затрат времени на проведение лабораторных работ, за счет использования компьютерных эффектов;
- Финансовая экономия ресурсов, так как виртуальные стенды обходятся значительно дешевле;
- Простота модификации стенда (в том числе в индивидуальном порядке) в случае необходимости;
- Финансовая экономия ресурсов – нет необходимости в закупке сырьевых или других материалов, так как они также заменяются виртуальными аналогами;

- Отсутствие дополнительных расходов в случае выхода оборудования из строя – достаточно перезапустить/переустановить виртуальный стенд;
- Наличие возможности демонстрации последствий неверного следовая инструкциям, несоблюдения техники безопасности или других опасных ситуаций.

1.1. Подходы к организации практических занятий

В настоящее время можно выделить четыре основных подхода к организации практических занятий с использованием лабораторных стендов. Проведение оных занятий возможно при помощи:

1. реального объекта;
2. физического имитатора;
3. программно-аппаратного имитатора (HIL);
4. цифрового двойника.

В первом случае для организации занятий предполагается наличие реального объекта, на котором практиканты будут отрабатывать свои навыки. Значительным преимуществом данного подхода является непосредственное взаимодействие студента с объектом изучения. Однако данный вариант организации занятий сопряжен с рядом рисков, таких как выход оборудования или создание потенциально опасных ситуаций. Также организация даже одного стенда может обойтись в значительную сумму, обеспечить доступ каждого студента к своему экземпляру оборудования порой становиться просто невозможно, что в свою очередь приводит, во-первых, к ограничению времени доступа к стенду каждого из студентов, а во-вторых, к необходимости тщательного слежения за работоспособностью установки (во избежание прекращения лабораторных занятий вовсе).

Проведение практических занятий с использованием физического имитатора подразумевает создание упрощенной и/или масштабированной модели реально существующего объекта. В этом случае организация стенда может предоставлять студентам возможность более подконтрольного изменения

состояний системы для имитации отказов или демонстрации критических ситуаций. Однако финансовые затраты на такого подхода все еще могут быть высоки в случае, например, содержания большого числа аналогичных стендов, их поломок или закупки сырьевых материалов (при необходимости).

Hardware-in-the-Loop (HIL) - это специализированные стенды, на которых в реальном времени работает цифровая модель объекта управления. К ним подключается испытываемый блок управления (контроллер), который посылает стенду реальные сигналы для для изменения состояния модели объекта. Этот вариант способен предотвратить проблемы связанные с выходом реального оборудования из строя, тем самым сводя финансовые затраты к закупке программируемых контроллеров, что является значительной выгодой. Благодаря такому подходу, объект управления становится прост для внедрения изменений, моделирования критических ситуации или имитации отказов.

Как и предыдущий, вариант с использование цифрового двойника предоставляет виртуальную модель реального объекта управления. Однако, взамен загрузки управляющего кода на реальный контроллер это происходит на виртуальном блоке управления. Такой подход минимизирует финансовые затраты на содержание лабораторных стендов, предоставляет возможность свободной модификации объекта управления, простого восстановления исходного состояния системы (за счет переустановки виртуального стенда), а также возможность предоставления персонального доступа для каждого из студентов.

Был проведен анализ данных подходов по основным аспектам, влияющим на склонность к выбору их в качестве релевантного в вопросе способа организации практических занятий (прил. 1). Результаты данного исследования подтверждают актуальность и выгоду использования виртуальных лабораторных стендов.

1.2. Типовой состав виртуального стенда

Для создания любого лабораторного стенда в первую очередь необходима организация объекта управления. В случае ВЛС в качестве такого объекта выступает цифровой двойник (виртуальная модель реального объекта). Управление физическими стендами, как правило, осуществляется при помощи кнопок, джойстиков, пультов и т. д., а отслеживание состояния системы происходит через различные индикаторы и датчики. Следовательно, цифровой двойник также должен обладать виртуальными аналогами данных интерфейсов.

Преимуществом использования ВЛС является возможность введения графических дополнений в виде математических моделей, графиков, диагностических панелей или, при необходимости, отельных компонентов объекта управления, что позволяет в полной мере оценивать состояние и работоспособность системы.

Для формирования необходимых умений, в комплекте с виртуальным стендом может прилагаться ряд типовых ситуационных задач, на которых предполагается отработка навыков. Также могут включаться тесты, позволяющие оценить верность решений или работоспособность системы.

Для ВЛС, демонстрирующих сложные киберфизические системы является хорошей практикой создание интерактивных и мультимедийных пособий, прилагающихся в к стенду или находящихся в открытом доступе. В таком случае, если у кого-то из студентов возникают затруднения, часть их вопросов может быть уже рассмотрена заранее, что позволяет быстро ориентироваться в проблемных ситуациях.

В отличии от реальных физических объектов управления, каждый студент имеет возможность изменить параметры лабораторного стенда в индивидуальном порядке. В тоже время, преподаватель способен вмешаться в работу в любой момент для остановки, перезапуска стенда или просмотра его параметров.

1.3. Обзор сред для разработки ВЛС

В качестве базы под виртуальные лабораторные стенды часто выбирают такие программные обеспечения как SimInTech, LabVIEW, logi.CAD или CODESYS. Рассмотрим каждое из них в отдельности.

1.3.1 SimInTech

SimInTech («Simulation in technik») - программа, разработанная российской компании ООО «3В Сервис», которая предназначена для моделирования систем автоматического управления. В настоящий день SimInTech выступает как отечественная альтернатива таким программным продуктам, как SimuLink (MathWorks), VisSim (Visual Solution), LabView (National Instruments) и т.д.

В качестве достоинств данного программного средства можно выделить следующие факторы:

- возможность интеграции с различным ПО
- наличие инструментов для создания интерфейсов управления
- моделирование в режиме «реального времени»
- как бонус, наличие полной русификации

SimInTech хорошо подходит для моделирования и детального анализа различных физических процессов, например, теплогидравлики, механических взаимодействий, электрических приводов и пр. Но основными направлениями для использования SimInTech являются создание моделей объектов управлений, проектирование и разработка алгоритмов для управления этими объектами, далее их отладка и генерация исходного кода для программируемых контроллеров на одном из поддерживаемых языков (Си, ST).

SimInTech обладает возможностью подключения ряда вспомогательных библиотек для моделирования типовых блоков в различных отраслях, а также для создания дополнительных интерфейсов, проведения аналитики, верификации и много другого.

Значительным недостатком данного средства является отсутствие его свободного распространения. Скачать данное ПО возможно только по предварительной заявке. Стоит также принять во внимание тот факт, что при доступе к ознакомительной версии SimInTech существует ряд ограничений:

- максимальное количество блоков в модели - 250
- максимальное количество функций - 50
- не поддерживается генерация кода СИ
- возможны другие ограничения в зависимости от версии ПО

1.3.2 LabVIEW

LabVIEW - это платформа системного проектирования и среда разработки, которая была нацелена на создание возможностей для разработки всех форм систем. Она была разработана компанией National Instruments в качестве рабочей среды для управления контрольно-измерительными приборами. Однако ее применение распространилось на всю область проектирования и эксплуатации систем.

LabVIEW используется в системах сбора и обработки данных, а также для управления техническими объектами и технологическими процессами. Идеологически LabVIEW очень близка к SCADA-системам, но в отличие от них в большей степени ориентирована на решение задач не столько в области АСУ ТП, сколько в области АСНИ.

Программа LabVIEW называется и является виртуальным прибором (англ. Virtual Instrument) и состоит из двух частей:

- блочной диаграммы, описывающей логику работы виртуального прибора;
- лицевой панели, описывающей внешний интерфейс виртуального прибора.

Виртуальные приборы могут использоваться в качестве составных частей для построения других виртуальных приборов.

Лицевая панель виртуального прибора содержит средства ввода-вывода: кнопки, переключатели, светодиоды, верньеры, шкалы, информационные табло

и т. п. Они используются человеком для управления виртуальным прибором, а также другими виртуальными приборами для обмена данными.

Блочная диаграмма содержит функциональные узлы, являющиеся источниками, приёмниками и средствами обработки данных. Также компонентами блочной диаграммы являются терминалы («задние контакты» объектов лицевой панели) и управляющие структуры (являющиеся аналогами таких элементов текстовых языков программирования, как условный оператор «IF», операторы цикла «FOR» и «WHILE» и т. п.). Функциональные узлы и терминалы объединены в единую схему линиями связей.

LabVIEW поддерживает огромный спектр оборудования различных производителей и имеет в своём составе (либо позволяет добавлять к базовому пакету) многочисленные библиотеки компонентов:

- для подключения внешнего оборудования по наиболее распространённым интерфейсам и протоколам (RS-232, GPIB-488, TCP/IP и пр.);
- для удалённого управления ходом эксперимента;
- для управления роботами и системами машинного зрения;
- для генерации и цифровой обработки сигналов;
- для применения разнообразных математических методов обработки данных;
- для визуализации данных и результатов их обработки (включая 3D-модели);
- для моделирования сложных систем;
- для хранения информации в базах данных и генерации отчётов;
- для взаимодействия с другими приложениями в рамках концепции COM/DCOM/OLE.

Специальный компонент *LabVIEW Application Builder* позволяет создавать LabVIEW-программы, пригодные для выполнения на тех компьютерах, на которых не установлена полная среда разработки. Для работы таких программ

требуется бесплатно распространяемый компонент «LabVIEW Runtime Engine» и, при необходимости, драйверы используемых внешних устройств.

К недостаткам данного средства можно отнести то, что этот продукт с закрытым исходным кодом. Версии для Windows, начиная с 8.2, требуют активации. Хотя для Linux и MAC такой необходимости нет, но поддержка данных платформ в свою очередь является ограниченной: нет драйверов, нет специальных toolkit-программ.

1.3.3 logi.CAD

Инженерное программное обеспечение для создания приложений контроллеров для промышленной автоматизации. Системы всех видов могут быть запрограммированы в соответствии с IEC 61131-3, от микроконтроллера до различных OEM-платформ и многоядерных промышленных ПК.

logi.CAD 3 вместе с системой выполнения logi.RTS, позволяет реализовать экономически эффективные, мощные платформы контроллеров (пользовательские ПЛК) на различных аппаратных системах. Это приводит к решениям, которые точно настроены в соответствии с требованиями производителей машин и систем.

Являясь открытой, масштабируемой и мощной системой, logiCAD 3 специально разработан для удовлетворения разнообразных требований наших OEM-партнеров. Открытая платформа легко адаптируется, что позволяет эффективно разрабатывать независимые от производителя решения для автоматизации. Это повышает производительность проектирования на каждом этапе проекта.

logi.CAD 3 основан на самых современных технологиях

- Модульная конструкция обеспечивает гибкое расширение
- Поддержка миграции
- Простая аппаратная интеграция
- Лучшее для всех

Доступный и подходящий даже для компактных приложений

- Гибкие модели лицензирования
- Долгосрочная доступность
- Не зависит от размера компана
- Личная поддержка

Современный фреймворк Eclipse позволяет подключать инструменты исходного кода и анализа, а также легко интегрировать непрерывную интеграцию и тестовые среды.

- Мощный редактор ST (структурированный текст)
- Современный графический редактор FBD (функциональная блок-схема)
- Доступна интеграция C и C++
- Полное хранение данных в виде простого текста - никаких проприетарных двоичных форматов данных
- Поддержка новейших систем управления исходным кодом (например, GIT)
- Поддержка новейших систем проверки кода (Google Gerrit)
- Возможны подключения к системам отслеживания проблем (например, JIRA)
- Статический анализ кода
- Модульный тестовый фреймворк для функций и функциональных блоков
- SDK для настройки logi.Система программирования CAD 3

Благодаря Java и Eclipse, logi.CAD 3 обеспечивает долгосрочную доступность на всех платформах. Для Windows®доступна немедленная загрузка. Платформы Linux и Mac OS® X доступны по запросу.

1.3.4 CODESYS

Основой комплекса CODESYS является среда разработки прикладных программ для программируемых логических контроллеров (ПЛК). Она распространяется бесплатно и может быть без ограничений установлена на нескольких рабочих местах.

В CODESYS для программирования доступны все пять определяемых стандартом IEC 61131-3 (МЭК 61131-3) языков.

В CODESYS реализован ряд других расширений спецификации стандарта IEC 61131-3. Самым существенным из них является поддержка Объектно-ориентированного программирования (ООП).

Встроенные компиляторы CODESYS генерируют машинный код (двоичный код), который загружается в контроллер. Поддерживаются основные 16- и 32-разрядные процессоры: Infineon C166, TriCore, 80x86, ARM (архитектура), PowerPC, SH, MIPS (архитектура), Analog Devices Blackfin, TI C2000/28x и другие.

При подключении к контроллеру среда программирования переходит в режим отладки. В нем доступен мониторинг/изменение/фиксация значений переменных, точки останова, контроль потока выполнения, горячее обновление кода, графическая трассировка в реальном времени и другие отладочные инструменты.

CODESYS версии V3 построен на базе так называемой платформы автоматизации: CODESYS Automation Platform. Она позволяет изготовителям оборудования развивать комплекс путём подключения собственных плагинов.

Расширенная профессиональная версия среды разработки носит название CODESYS Professional Developer Edition. Она включает поддержу UML-диаграмм классов и состояний, подключение системы контроля версий Subversion, статический анализатор и профилировщик кода. Распространяется по лицензии.

Инструмент CODESYS Application Composer позволяет перейти от программирования практических приложений к их быстрому составлению. Пользователь составляет собственную базу объектов, соответствующих определенным приборам, механическим узлам машины и т. п. Каждый объект включает программную реализацию и визуальное представление. Законченное

приложение составляется из необходимых объектов, конфигурируется и автоматически генерируется программа на языках МЭК 61131-3.

CODESYS Automation Server - это облачная платформа автоматизации для контроллеров с CODESYS. Обеспечивает: удаленный мониторинг данных ПЛК, контроль исправности ПЛК, обновление ПО ПЛК по расписанию, резервное копирование проектов и параметров, контроль версий, удаленное формирование нарядов для обслуживания на местах.

После проведенного обзора была сформирована сводная таблица, описывающая преимущества и недостатки вышеперечисленных средств на основе принципиальных для разработки виртуальных лабораторных стендов для ПЛК МЭК-61131 критериев (прил. 2). По результатам проведенного анализа было принято решение, что наиболее подходящей целевой средой для разработки ВЛС является среда CODESYS.

1.4. Возможности CODESYS по визуализации

Основными компонентами по визуализации в среде CODESYS являются:

- Менеджер визуализации;
- Профиль визуализации;
- Редактор визуализации;
- Пул изображений.

1.4.1 Менеджер визуализации

Компонент менеджер визуализации создается либо вручную, либо в момент добавления в проект первого экрана для визуализации. Он отвечает за настройки, которые применяются ко всем экранам визуализации. К таким установкам относятся:

- Общие установки (стиль, язык, использование мультикасаний, unicode и пр.)
- Dialog Settings (диалоги ввода по умолчанию)
- Визуализации (загрузка экранов визуализации на устройство)
- Управление пользователями (ограничения и права)

- Установки списка текстов (шрифты, размеры, языки)
- Горячие клавиши

1.4.2 Профиль визуализации

Профиль визуализации занимается настройкой версий библиотек которые используются в проекте для визуализации. Содержание профиля можно увидеть в “Инструменты” - “Репозитории визуальных элементов” (все проекты должны быть закрыты). От выбранных параметров зависит то, какие примитивы будут доступны для создания визуализации а также количество их настроек. Сменить профиль визуализации, используемый в проекте можно посредством изменения установок проекта (рис. 1).

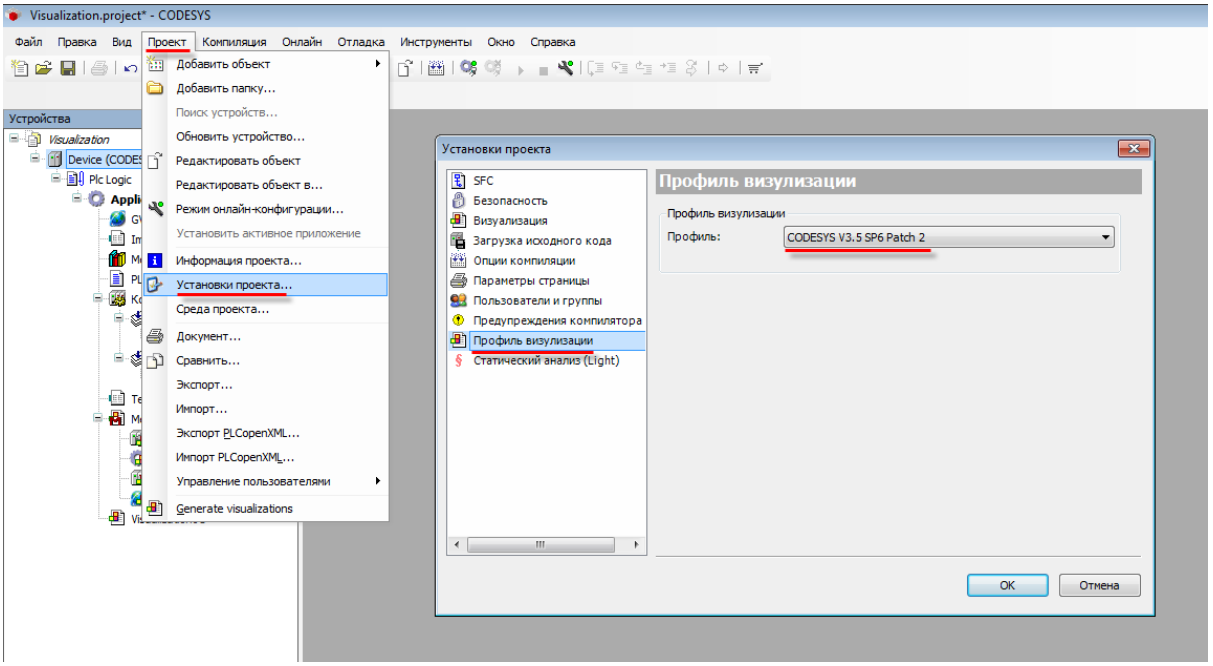

Рисунок 1 Выбор профиля визуализации, используемого в проекте

1.4.3 Редактор визуализации

Основная работа по созданию визуализации проводится в редакторе визуализации, который открывается в случае выбора любого экрана визуализации проекта.

Интерфейс редактора визуализации включает в себя:

1. Рабочую область. Основной экран, на котором происходит взаимодействие с объектами визуализации.

2. Область объявления переменных. Панель над рабочей областью, позволяет определить входные и выходные переменные для текущего экрана.
3. Панель инструментов редактора. Боковая панель, которая содержит набор графических примитивов.
4. Панель свойств. Боковая панель (переключение с панелью инструментов), которая содержит параметры выбранного графического элемента.

Взаимодействие с элементами визуализации происходит посредством компьютерной мыши. С элементом возможны следующие манипуляции:

- Перемещение
- Вращение
- Изменения размера
- Изменения текста (опционально)
- Перемещения центра (для относительных манипуляций)

Взаимодействие может происходить с несколькими элементами одновременно, если их выделить вместе. Сделать это можно двумя способами:

- Зажав ЛКМ, обвести их рамкой
- Зажать Shift и последовательно нажимать на элементы ЛКМ

Также в процессе работы доступны команды редактора визуализации, обратиться к которым можно через:

- Вкладку “Визуализация” в строке меню CODESYS
- ярлыки, расположенные чуть ниже
- контекстном меню “Редактора визуализации”

Команды визуализации предоставляют интерфейс к следующим возможностям:

1. Создать глобальный пул изображений. Действие распространяется на все приложения проекта (в отличии от созданных на Панели устройств для конкретного приложения)
2. Создать глобальный список текстов. Действие распространяется на все приложения проекта (в отличии от созданных на Панели устройств для конкретного приложения)
3. Порядок. Определяет размещение элементов относительно друг друга.
4. Выравнивание. Выравнивает элементы и настраивать их отступы и междустрочные интервалы (доступно при выборе нескольких элементов).
5. Группа. Объединяет несколько элементов, позволяя в дальнейшем работать с ними как с одним (доступно при выборе нескольких элементов).
6. Разгруппировать. Противоположна команде “Группа”, объединенный элементы становятся обособленными (доступно для сгруппированных элементов).
7. Содержимое фрейма. Позволяет выбрать экраны визуализации, которые будут отображаться в данных элементах (доступно для элементов Фрейм и Набор вкладок).
8. Фон. Позволяет выбрать цвет фона экрана визуализации или фоновое изображение.
9. Размножить. Создает группы одинаковых элементов.

1.4.4 Пул изображений

Пул изображений является вспомогательным компонентом в системе визуализации. Он используется для загрузки и хранения в проекте ряда графических файлов, которые могут быть применены в качестве фоновых изображений, новых примитивных элементов, переключателей или пиктограмм для Менеджера тревог. CODESYS поддерживает большинство популярных графических расширений (jpg/jpeg, png, svg и т. д.)

Добавленные в проект изображения и информация о них хранится в в таблице (Пула изображений) в виде списка (рис. 2).

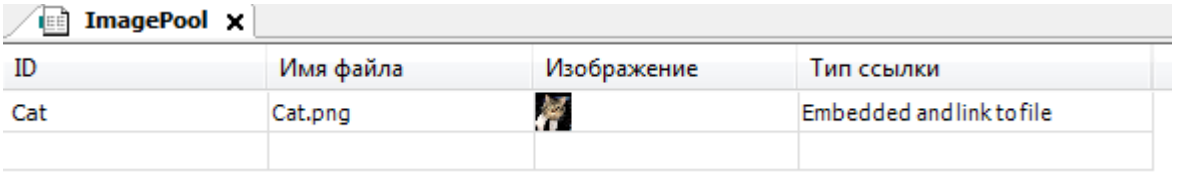

Рисунок 2 Пул изображений после добавления файла

2. Основная работа

2.1. Требования к ВЛС

В качестве базовых требований к лабораторному стенду были выдвинуты следующие требования, обеспечивающие разумный баланс между наглядностью создаваемых имитаторов робототехнических комплексов, развитостью средств программирования алгоритмов управления и трудоемкостью решаемых задач.

- возможность визуализировать анимированные 2D объекты;
- реализация поведения имитатора объекта управления;
- независимость спецификации алгоритма управления и модели объекта.

Требования к этапу занятий студентов со стендом следующие:

- должны присутствовать как ручной, так и автоматический режимы управления;
- алгоритм управления должен быть задан на языке poST.

2.2. Архитектура ВЛС на базе CODESYS

При реализации виртуального стенда на базе CODESYS была предложена следующая архитектура (рис. 3).

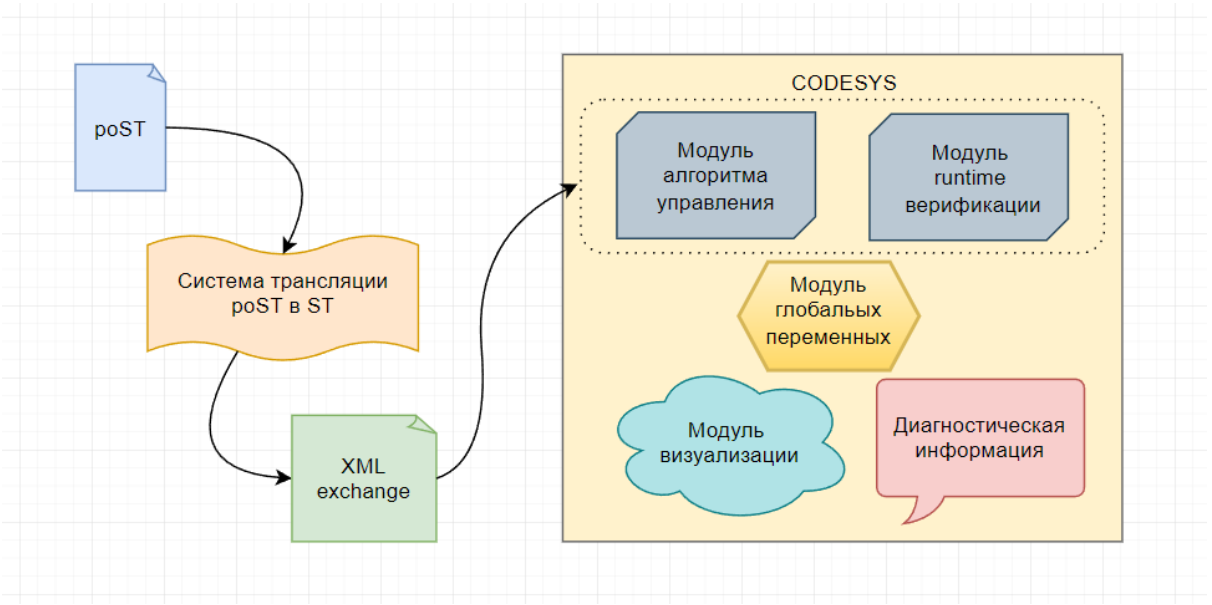

Рисунок 3 Архитектура ВЛС

В процессе практических занятий по курсу “Процесс-ориентированное программирование” студент создает реализацию программы на языке poST (на самом деле, благодаря поддержке концепции IEC 61131-3, программный компонент может разрабатываться на любом языке, поддерживаемом стандартом), описывающий работу контроллера. После, данный код поступает в систему трансляции poST в ST, где на выходе генерируется PLC open XML Exchange файл, структура которого определяется стандартом IEC 61131-10. В качестве такой системы может выступать находящийся в открытом доступе онлайн транслятор языка poST (http://post2st.iae.nsk.su) или разрабатываемая под руководством ИАиЭ среда разработки poST IDE.

Преимуществом PLC open XML Exchange файлов является простота экспорта и импорта проектов в средствах IEC 61131. CODESYS является средой, полностью поддерживающей данный стандарт. Благодаря этому практикант беспрепятственно может перенести необходимые модули в разработанный лабораторный стенд посредством всего пары кликов.

В самом проекте CODESYS располагается модуль визуализации, отображающий процесс работы киберфизической системы, панель с диагностической информацией, модуль глобальных переменных, благодаря которым происходит взаимосвязь объекта управления и блока управления.

2.3. Методика создания ВЛС

В целом методика основана на концепции, предложенной М. Джексоном, которая рассматривающим киберфизическую систему как программно-аппаратный артефакт. Это очень близко к концептуальному подходу Джексона, согласно которому киберфизические системы это артефакты, нацеленные на реальный мир и изменяющие его. Логическим следствием этой точки зрения является то, что требования относятся к взаимосвязям в реальном мире — домену приложения, а не к программной системе или даже к интерфейсу с программной системой [12]. В этой

концепции созданный артефакт встраивается в реальный мир и преображает его.

Процедура создания комплекта (рис. 4) включает в себя семь этапов.

1. Описание задачи на естественном языке
2. Создание киберфизической диаграммы, определяющей отношения между тремя компонентами: средой, оборудованием и программным обеспечением.
3. Определение того, какие инструменты визуализации будут использоваться
4. Создание симулятора объекта управления
5. Формулировка требований к управляющему ПО
6. Внедрение управляющего программного обеспечения в соответствии с требованиями
7. Комплексное тестирование трехкомпонентной системы «среда-аппаратное обеспечение-программное обеспечение»

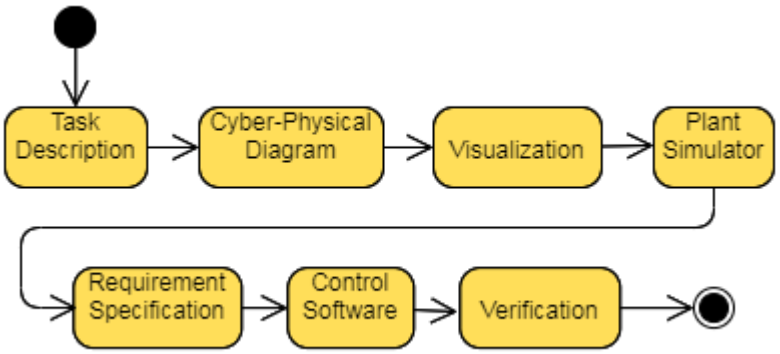

Рисунок 4 Схема рабочего процесса

В качестве задачи для автоматизации был выбран механизм шлюзового затвора.

Описание задачи Шлюз

Шлюз (рис. 5) — это гидротехническое сооружение на судоходных и водных путях, которое обеспечивает переход судов из одного водного бассейна (бьефа) в другой с различными уровнями воды в них. С двух сторон шлюз ограничен воротами, между которыми располагается герметичная шлюзовая

камера с регулируемым уровнем воды. Камера шлюза имеет габариты, достаточные для размещения в ней одного судна. За счет клапанов уровень воды в шлюзовой камере может изменяться. Ворота представляют собой металлические щиты, расположенные на обоих концах камеры. Они служат для пропускания и выпускания судов и герметизируют камеру во время переправки кораблей. Для контроля положения ворот используются датчики.

Одним из самых важных в конструкции шлюзов является водопроводное устройство. Оно предназначено для наполнения и опустошения камеры. В качестве такого устройства используются клапаны.

Принцип работы шлюза заключается в следующем: сначала открываются входные ворота, и судно заходит внутрь камеры и швартуется к тумбам. Затем входные ворота закрываются, и открывается перепускной клапан, вызывая падение или подъём уровня воды в камере с находящимся в ней кораблем. Для контроля состояния перепускных клапанов используются датчики. При этом уровень воды в шлюзовой камере выравнивается с уровнем воды в соседнем резервуаре по направлению следования судна. Для контроля выравнивания уровня воды в камере и бьефом тоже используются датчики. Когда вода достигает необходимого уровня, равного тому, который имеется за выходными воротами камеры, ворота открываются, через семафор подается разрешающий сигнал на перемещение судна и судно выходит из шлюза.

Через шлюз может проходить только один корабль. Шлюз оснащен датчиком наличия корабля на нижнем, движущемся наверх, и верхнем, движущемся вниз, уровнях и шлюзовой камере.

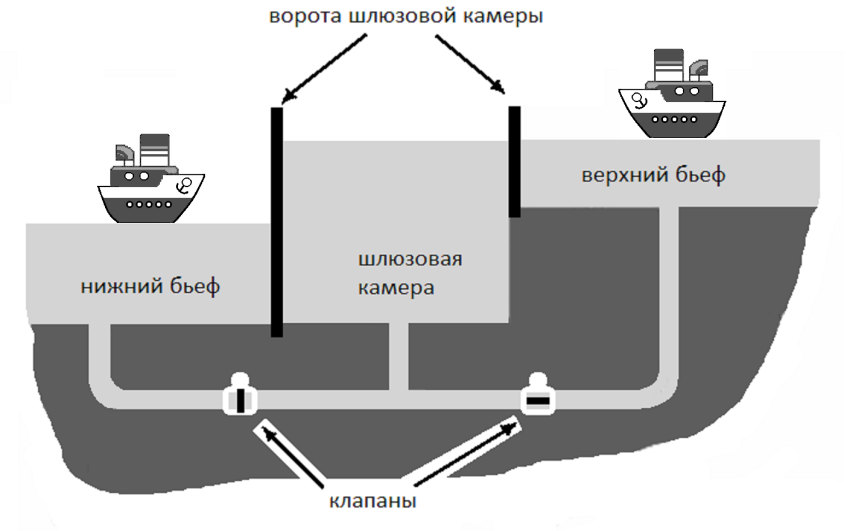

Рисунок 5 Шлюз

Киберфизическая диаграмма

Следующим шагом является создание киберфизической диаграммы. Киберфизическая схема (рис. 6) рассматривает систему как три взаимодействующих компонента. Корабли здесь действуют как Окружающая среда (Environment). Если физически возможно, корабли могут перемещаться между нижним бьефом, камерой и верхним бьефом в соответствии со светофорами и правилами плавания.

На схеме шлюз представляет установка (Plant). В соответствии с фактическим положением кораблей формируются Controls – сигналы, указывающие на состояние внешней среды (shipInHigh, shipInLow, shipInChmbr). В соответствии с фактическим состоянием объекта управления датчики (Sensors) отображают информацию об уровне воды в камере (atHigh, atLow), положениях затворов (HighGateOpened, LowGateOpened, HighGateClosed, LowGateClosed) и состояниях клапанов (HighValveOpened, LowValveOpened). Приводы ворот (openHighGate, openLowGate) и кламанов (openHighValve, openLowValve) используются для управления состояниями

открытия и закрытия. Для управления светофорами используются сигналы (Low2ChmbrLight, Chmbr2LowLight, Chmbr2HighLight, High2ChmbrLight).

Контроллер (Controller) определяет поведение системы в соответствии с управляющей программой. На основе информации, полученной от управления и датчиков, контроллер изменяет состояние системы через исполнительные устройства и индикаторы.

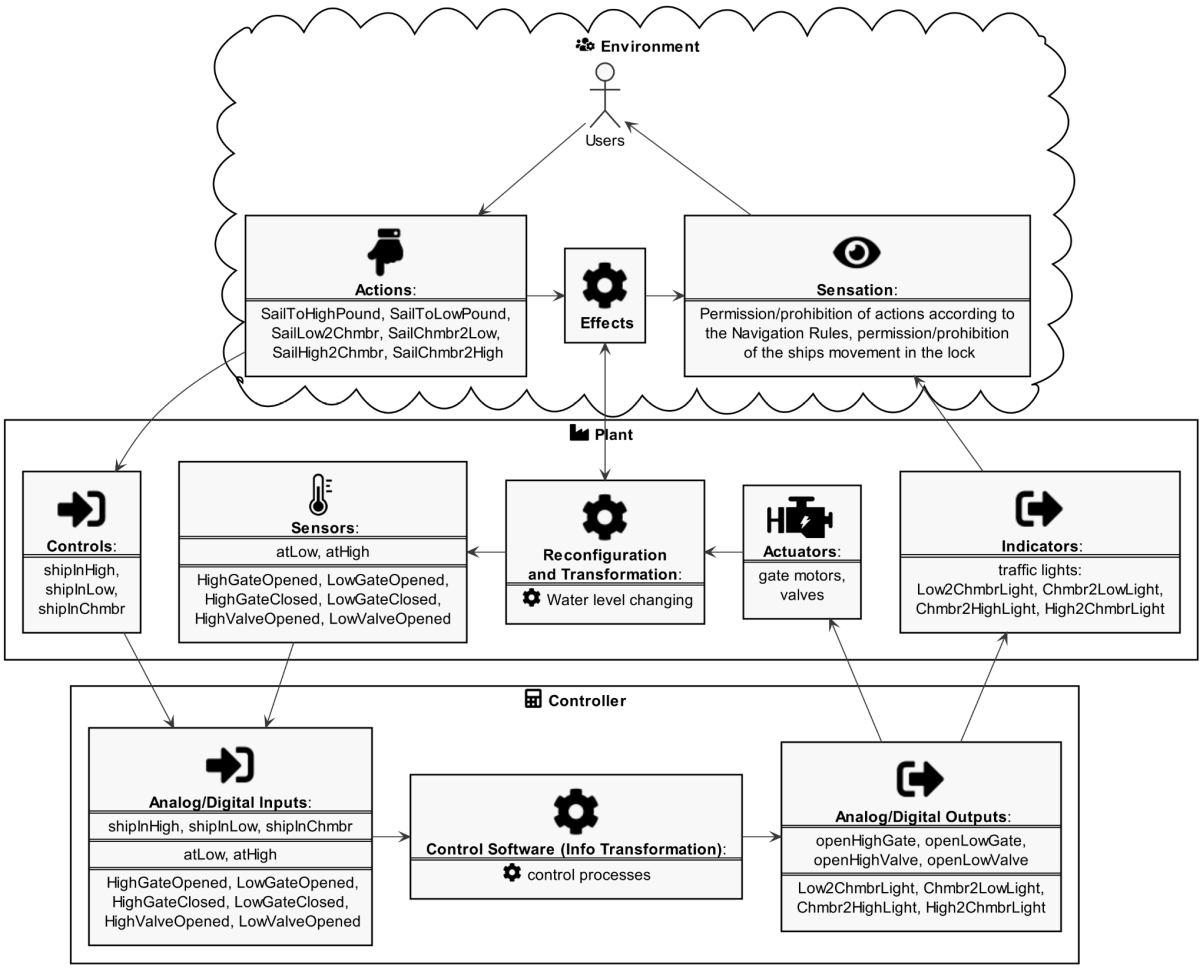

Рисунок 6 Киберфизическая схема шлюзового затвора

Цифровой двойник

Симулятор, или цифровой двойник, состоит из визуализации и поведенческой модели. Для шлюза было определено, что мы визуализируем изменение уровня воды в камере и состояния клапанов. Кнопки ввода предназначены для подачи команд управления кораблем на графический

интерфейс. Элементы управления, датчики, приводы, индикаторы также являются частью интерфейса оператора.

В автоматическом режиме обучаемому доступны только команды управления кораблем. В ручном режиме управления, когда учащийся выступает в роли контролера, ему также разрешено изменять состояния исполнительных механизмов и индикаторов.

Установка моделируется пятью независимыми языковыми процессами poST (UpperGateSim, LowerValveSim, UpperValveSim, LowerGateSim, WaterLevelSim), которые на основе состояния исполнительных механизмов определяют положения ворот и уровень воды, а также формируют сигналы датчиков.

Требобвания и контрольное программное обеспечение в poST

После создания симулятора объекта, который уже содержит некоторый алгоритм управления, неявно заданный через органы управления, датчики, исполнительные механизмы и индикаторы можно приступить к формированию требований к Контроллеру. Это императивные требования, указанные с помощью EDTL-нотации [14]. Был использован EDTL, потому что он обеспечивает:

- просту формулировки требований за счет настройки шести атрибутов,
- описание требований только через входные/выходные переменные,
- независимость методов, используемых для последующей проверки,
- математическую обоснованность, обеспечивающую непротиворечивость и полноту требований.

Сформулируем следующие требования на естественном языке:

- Ворота ниже по потоку должны быть закрыты, если уровни камеры и ниже по потоку не совпадают.
- Ворота выше по потоку должны быть закрыты, если уровни камеры и выше по потоку не совпадают.

- Должен быть активирован только один привод, либо один из затворов, либо один из клапанов.
- После открытия клапана он закрывается при срабатывании датчика уровня соответствующих уровней.
- Через 3 секунды после подачи сигнала на открытие ворот загорается соответствующий светофор (если в камере есть корабль, выйти, если нет, войти).
- Светофор гаснет после закрытия соответствующих ворот.
- После открытия ворота остаются открытыми до тех пор, пока в камере или с противоположной стороны не появится корабль.

Req ID	**Trigger event**	**Release event**	**Final event**	**Allowable delay**	**Invariant**	**Reaction**
R1	**NOT** *atLow*	**FALSE**	**TRUE**	**TRUE**	LowGateClosed	**TRUE**
R2	**NOT** atHigh	**FALSE**	**TRUE**	**TRUE**	HighGateClosed	**TRUE**
R3	**TRUE**	**FALSE**	**TRUE**	**TRUE**	**XXOR** (openHighGate, openLowGate, openHighValve, openLowValve)	**TRUE**
R4	**RE** openHighValve	**FALSE**	atHigh	**TRUE**	openHighValve	**NOT** openHighValve

R5	**RE** openHighGate **AND** shipInChmbr	**FALSE**	τ(3s)	**TRUE**	**TRUE**	Chmbr2LowLight
R6	**RE** LowGateClosed	**FALSE**	**TRUE**	**TRUE**	**TRUE**	**NOT** Chmbr2HighLight **AND** **NOT** High2ChmbrLight
R7	**RE** LowGateOpened	shipInChmbr **OR** shipInLow	**TRUE**	**TRUE**	LowGateOpened	**TRUE**

Таблица 1 - Табличная форма требований.

Программа управления (контроллер), написанная на языке poST, включает 15 взаимодействующих процессов (рис. 7). После инициализации индикаторов и исполнительных механизмов процесс Init запускает процесс Waiting4Ship.

Процесс Waiting4Ship отслеживает корабли через элементы управления и обрабатывает перемещение кораблей между пулами с помощью процессов UpControl и DownControl. Эти процессы открывают проход между бассейнами и шлюзовой камерой с помощью процессов Water2Low и Water2High. После обеспечения физической возможности движения в шлюзовую камеру процессы контролируют движение судов через светофоры (процессы LightOFF, Low2ChmbrON, Chmbr2LowON, High2ChmbrON, Chmbr2HighON). Физическая возможность входа/выхода судов в шлюзовую камеру осуществляется за счет непосредственного управления клапанами (сигналы HighValveOpened,

LowValveOpened) и процессами управления воротами (OpenLowGate, CloseLowGate, OpenHighGate, CloseHighGate).

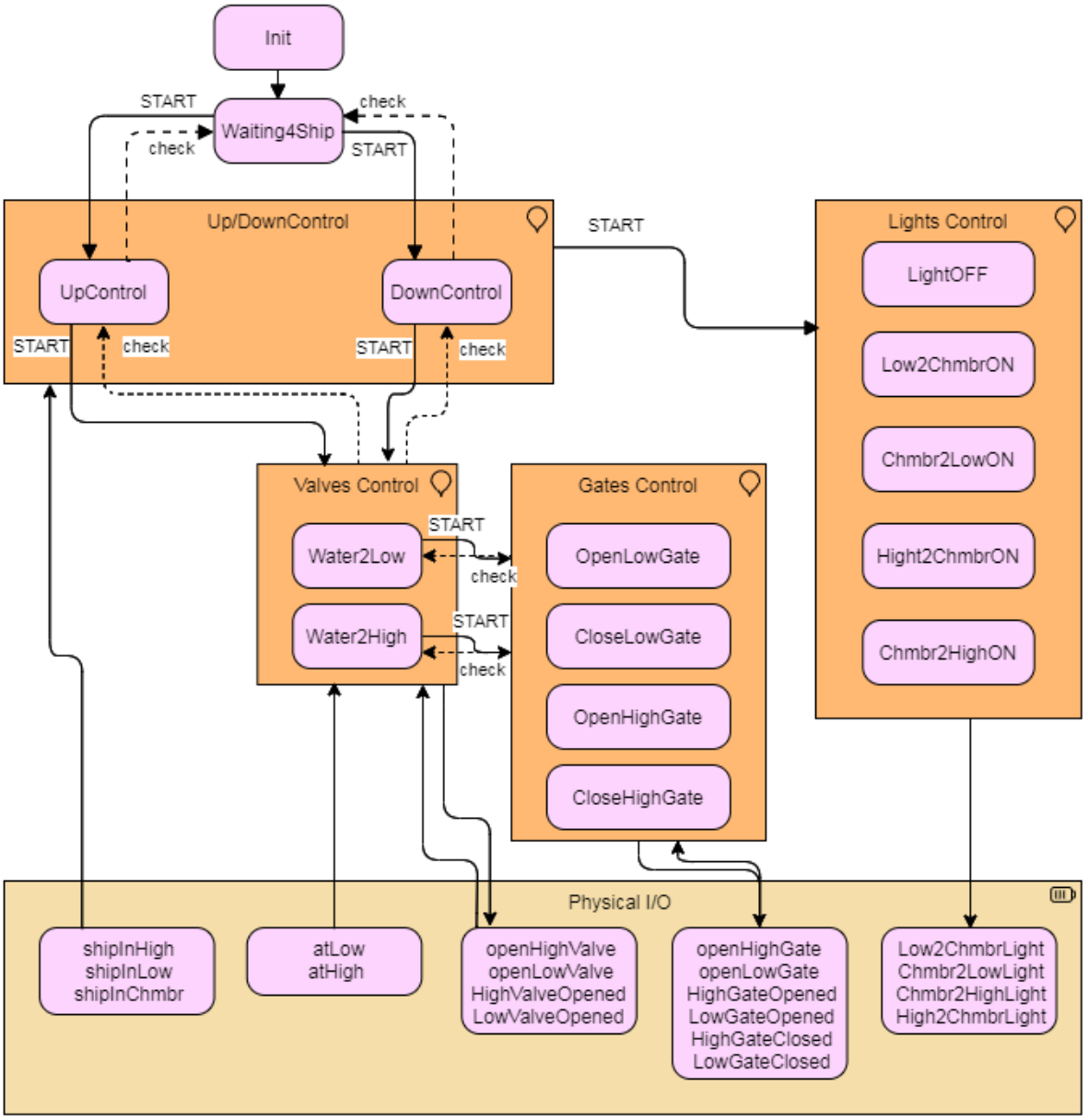

Рисунок 7 Технологическая схема управления шлюзом

Симулятор установки и алгоритм управления были реализованы с использованием poST/Eclipse IDE [9]. Поведение системы управления динамически проверялось с помощью CODESYS (рис. 10).

Программа контроллера реализует алгоритм управления и предназначена для загрузки в ПЛК. Установка представляет собой цифрового двойника шлюза

и моделирует физические процессы внутри установки. Она реализует блок реконфигурации и трансформации (рис. 6) для обеспечения поведения объекта.

Поведение системы управления было динамически проверено с использованием режима моделирования CODESYS [15] (рис. 8). Здесь взаимодействие между Установкой и Контроллером организовано через глобальные переменные, хранящиеся в отдельном GVL-файле.

По этой схеме контроллер устанавливает выходные переменные. Имитатор установки считывает эти переменные, затем пересчитывает уровень воды или координаты ворот в случае движения и формирует входные сигналы, такие как состояния датчиков уровня воды и ворот. Среди входных и выходных сигналов есть элементы управления и индикаторы (судовые датчики и огни), реализующие интерфейс с пользователем. При использовании в реальном физическом ПЛК и предприятия, эти глобальные переменные будут сопоставлены с входными/выходными портами ПЛК и будут представлять входные/выходные сигналы ПЛК. Состояние шлюзового оборудования отображается с помощью средств визуализации CoDeSys (рис. 10).

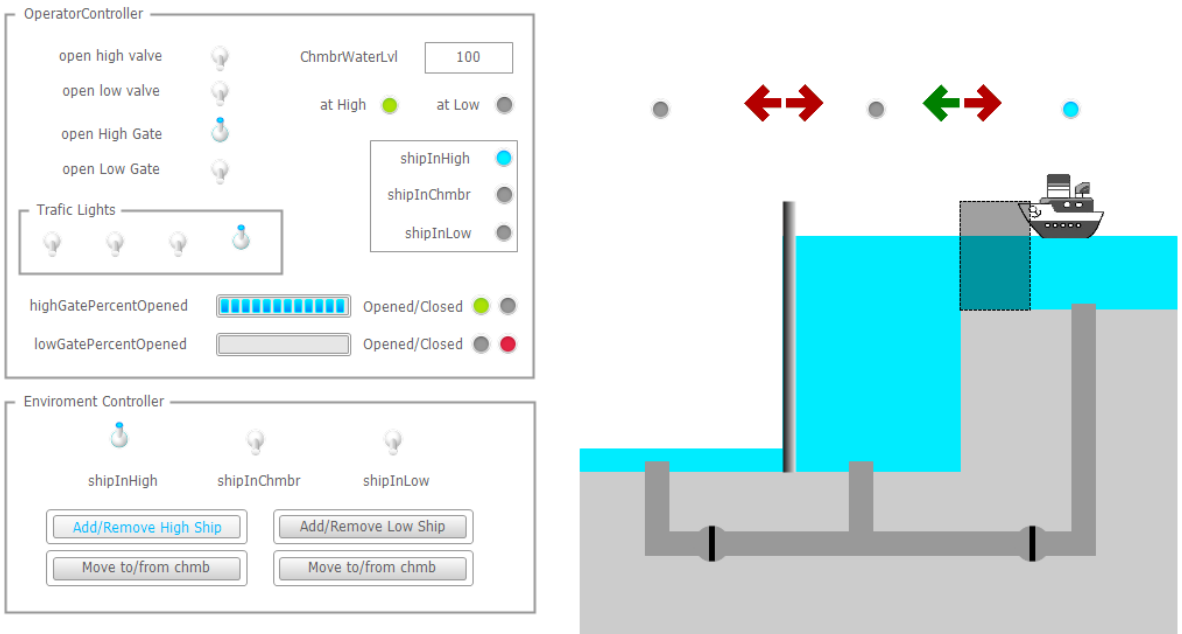

Рисунок 8 Визуализация учебного комплекта Sluice в CODESYS

ЗАКЛЮЧЕНИЕ

В рамках работы были рассмотрены способы организации лабораторных стендов для языков ПЛК и IDE, поддерживающих IEC 61131-3. Было обосновано использование средства CODESYS. Был определен список требований, основные из которых – наличие средств визуализации и мониторинга функционирования программы, использование poST в качестве языка описания и модели объекта управления, и алгоритма, независимость связей между алгоритмом управления и моделью объекта управления. В качестве пилотной была выбрана задача автоматизации трехэтажного лифта. Был создан прототип виртуального лабораторного стенда, в котором программы создаются с помощью онлайн транслятора языка poST в язык IEC 61131-3 ST (http://post2st.iae.nsk.su), а затем переносятся в среду CODESYS через нативный для IEC 61131-3 механизм XML Exchange.

Представленный подход позволяет выработать у студентов общие навыки разработки киберфизических систем. Параллельно осваивается широко используемый пакет CODESYS. Созданные ВЛС с точки зрения физической модели установки и требований представляют особый интерес. Сами по себе они позволяют ставить перед учениками задачи, направленные исключительно на программирование.

Работа была представлена на международной научной студенческой конференции и включена в список тезисов за 2022 год (прил. 4).

Дальнейшим направлением работы будет обеспечение плавной интеграции языка pOST в CODESYS или другую IDE, например. грамм. с открытым исходным кодом [16].

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлен с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

Родченко Анна Сергеевна _____

ФИО студента *Подпись студента*

« ____ »_____20 __г.

(заполняется от руки)

СПИСОК ЛИТЕРАТУРЫ

1. A. Gershberg, S. Podyapoliskij, and L. Sorkin, “Kompiyuternyj trenazherdlya obucheniya operatorov ustanovki kataliticheskogo riformingav ooo po ”kirishinefteorgsintez”,” Avtomatizaciya v promyshlennosti, no. 7, p. 52, 2003.
2. A. R. Akparibo, A. Appiah, O. Fosu-Antwi et al., “Development of a programmable logic controller training platform for the industrial control of processes,” American Academic Scientific Research Journal for Engineering, Technology, and Sciences, vol. 15, no. 1, pp. 186–196, 2016.
3. B. Riera, P. Marang´e, F. Gellot, O. Nocent, A. Magalhaes, and B. Vigario, “Complementary usage of real and virtual manufacturing systems for safe plc training,” IFAC Proceedings Volumes, vol. 42, no. 24, pp. 89–94, 2010.
4. T. Georgieva, H. Yahoui, N. Bencheva, P. Daskalov, D. Banjerdpongchai, and P. Kittisupakorn, “Innovative plc training laboratory for developing industry 4.0 skills,” in 2022 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON). IEEE, 2022, pp. 505–509.
5. R. Moreno and R. Mayer, “Interactive multimodal learning environments,” Educational psychology review, vol. 19, no. 3, pp. 309–326, 2007.
6. J. Travis, LabVIEW for everyone. Pearson Education India, 2009.
7. I. Anureev, N. Garanina, T. Liakh, A. Rozov, V. Zyubin, and S. Gorlatch, “Two-step deductive verification of control software using reflex,” in International Andrei Ershov Memorial Conference on Perspectives of System Informatics. Springer, 2019, pp. 50–63.
8. R. Langmann and M. Stiller, “The plc as a smart service in industry 4.0 production systems,” Applied Sciences, vol. 9, no. 18, p. 3815, 2019.
9. V. E. Zyubin, A. S. Rozov, I. S. Anureev, N. O. Garanina, and V. Vyatkin, “post: A process-oriented extension of the iec 61131-3 structured text language,” IEEE Access, vol. 10, pp. 35 238–35 250, 2022.

10. M. Tiegelkamp and K.-H. John, IEC 61131-3: Programming industrial automation systems. Springer, 2010.
11. D. H. Hanssen, Programmable logic controllers: a practical approach to IEC 61131-3 using CODESYS. John Wiley & Sons, 2015.
12. M. Jackson, Problem frames: analysing and structuring software development problems. Addison-Wesley, 2001.
13. F. White, “Fluid mechanics seventh edition by frank m. white,” Power, 2011.
14. V. Zyubin, I. Anureev, N. Garanina, S. Staroletov, A. Rozov, and T. Liakh, “Event-driven temporal logic pattern for control software requirements specification,” in Fundamentals of Software Engineering, H. Hojjat and M. Massink, Eds. Cham: Springer International Publishing, 2021, pp. 92–107.
15. R. C. Harwell and K. L. Sparks, “Iec 61131-3, codesys standardize control logic: ease control programming across multiple controller platforms using iec 61131-3-based codesys programming software,” Control Engineering, pp. 20–22, 2011.
16. E. Tisserant, L. Bessard, and M. De Sousa, “An open source iec 61131-3 integrated development environment,” in 2007 5th IEEE International Conference on Industrial Informatics, vol. 1. IEEE, 2007, pp. 183–187

ПРИЛОЖЕНИЯ

Приложение 1.

Подход	Возможность модификации	Изменение среды для тестирования	Стоимость создания	Трудоемкость сопровождения	Безопасность	Имитация отказов	Методы обучения
На реальном объекте	Как правило не предусмотрена	Изменение физических условий	Реальный объект и ПЛК	Техническое обслуживание и материалы	Возможна поломка дорогостоящего оборудования	Намеренный вызов отказов оборудования	Объяснительно-иллюстративный Репродуктивный Проблемный
С использованием физического	Как правило не предусмотрена	Возможно наличие физических переклю	Контроллер и ПО для него, часто проприетарные	Техническое обслуживание производит	Возможна поломка контроллера	Возможно наличие физических	Объяснительно-иллюстративный

имитатора		чателей		разработчик		переключателей	Репродуктивный Проблемный
С использованием программно-аппаратного имитатора (HIL)	Изменение программы или визуализации	Возможно предусмотреть в программе	Real-time ОС и оборудование для неё, оборудование для подключения инструментов управления	Техническое обслуживание ПК, на котором запущен имитатор	Сбой ОС или программы, поломка ПК	Возможно предусмотреть в программе	Объяснительно-иллюстративный Репродуктивный Проблемный Эвристический
С использованием	Изменение программы или	Возможно предусмотреть в	Разработка цифрового	Обновление цифрового	Сбой программы	Возможно предусмотрет	Объяснительно-иллюст

цифрового двойника	визуализации	программе	двойника	двойника		ь в программе	ративный Репродуктивный Проблемный Эвристический Исследовательский

Таблица 2 - Подходы к организации практических занятий по программированию ПЛК

Приложение 2.

ПО\требования	Поддерживаемые языки	Средства визуализации	Возможность модификации пользователем	Способ распространения	Возможность подключения библиотек
CODESYS	IEC	Графически	Файл	Среда	Есть

	61131-3	е примитивы и анимация	проекта позволяет изменять код	разработки бесплатная, покупка аддонов	
SimInTech	Свой	Графически е примитивы и анимация	Имеется	По заявкам	Есть
LabVIEW	Свой	2D и 3D визуализац ия	В результате получается исполняемы й файл	Бесплатный Community Edition, но его нельзя использовать для образовательн ых целей	Есть
logi.CAD	Часть языков IEC 61131-3 Интеграци я C и C++	Через сторонние аддоны	Имеется	Базовая версия бесплатна	Есть, но их создание возможно только в профессион альной лицензии

Таблица 3 - Сводная таблица, преимуществ и недостатков средств для разработки ВЛС

Приложение 3. poST-код

```
VAR_GLOBAL (* VAR_INPUT *)
 (* Ship sensors *)
```

```
 shipInHigh  : BOOL;
 shipInLow : BOOL;
 shipInChmbr  : BOOL;
(* Water level sensors *)
 atLow  : BOOL;
 atHigh : BOOL;
(* Gate / Valves sensors *)
 HighGateOpened : BOOL;
 LowGateOpened : BOOL;
 HighGateClosed : BOOL;
 LowGateClosed : BOOL;
 HighValveOpened : BOOL;
 LowValveOpened : BOOL;
(* END_VAR
VAR *) (* VAR_OUTPUT *)
(* sluice gates *)
 openHighGate : BOOL;
 openLowGate : BOOL;
(* sluice valves *)
 openHighValve : BOOL;
 openLowValve : BOOL;
(* traffic lights *)
 Low2ChmbrLight : BOOL;
 Chmbr2LowLight : BOOL;
 Chmbr2HighLight : BOOL;
 High2ChmbrLight : BOOL;
END_VAR

PROGRAM Plant (* Sluice *)
```

```
(* Plant *)
PROCESS Init
  STATE begin
    (* inputs: *)
    shipInHigh  := FALSE;
    shipInLow := FALSE;
    shipInChmbr  := FALSE;
    atLow  := TRUE;
    atHigh := FALSE;
    HighGateOpened := FALSE;
    LowGateOpened := FALSE;
    HighValveOpened := FALSE;
    LowValveOpened := FALSE;
(* outputs: *)
    openHighGate := FALSE;
    openLowGate := FALSE;
    (* sluice valves *)
    openHighValve := FALSE;
    openLowValve := FALSE;
    (* traffic lights *)
    Low2ChmbrLight := FALSE;
    Chmbr2LowLight := FALSE;
    Chmbr2HighLight := FALSE;
    High2ChmbrLight := FALSE;
    START PROCESS UpperGateSim;
    START PROCESS LowerGateSim;
    START PROCESS UpperValveSim;
    START PROCESS LowerValveSim;
    START PROCESS WaterLevelSim;
```

```
    STOP;
  END_STATE
END_PROCESS

PROCESS UpperGateSim
  VAR CONSTANT
    GATE_SPEED : REAL := 0.5;
    GATE_OPEN_COORD : REAL := 100;
  END_VAR
  VAR
    coord : REAL := 0.0;
  END_VAR
  STATE check_open_close LOOPED
    IF openHighGate THEN
      coord := coord + GATE_SPEED;
    ELSE
      coord := coord - GATE_SPEED;
    END_IF
    IF coord <= 0.0 THEN
      coord := 0.0;
    END_IF
    IF coord >= GATE_OPEN_COORD THEN
      coord := GATE_OPEN_COORD;
    END_IF
    IF coord = GATE_OPEN_COORD THEN
      HighGateOpened := TRUE;
    ELSE
      HighGateOpened := FALSE;
    END_IF
```

```
    IF coord = 0.0 THEN
      HighGateClosed := TRUE;
    ELSE
      HighGateClosed := FALSE;
    END_IF
  END_STATE
END_PROCESS

PROCESS LowerGateSim
  VAR CONSTANT
    GATE_SPEED : REAL := 0.5;
    GATE_OPEN_COORD : REAL := 100;
  END_VAR
  VAR
    coord : REAL := 0.0;
  END_VAR
  STATE check_open_close LOOPED
    IF openLowGate THEN
      coord := coord + GATE_SPEED;
    ELSE
      coord := coord - GATE_SPEED;
    END_IF
    IF coord <= 0.0 THEN
      coord := 0.0;
          END_IF
    IF coord >= GATE_OPEN_COORD THEN
      coord := GATE_OPEN_COORD;
    END_IF
    IF coord = GATE_OPEN_COORD THEN
```

```
      LowGateOpened := TRUE;
    ELSE
      LowGateOpened := FALSE;
    END_IF
    IF coord = 0.0 THEN
      LowGateClosed := TRUE;
    ELSE
      LowGateClosed := FALSE;
    END_IF
  END_STATE
END_PROCESS

PROCESS UpperValveSim
  STATE check_open_close LOOPED
    IF openHighValve = TRUE THEN
      HighValveOpened := TRUE;
    ELSE
      HighValveOpened := FALSE;
    END_IF
  END_STATE
END_PROCESS

PROCESS LowerValveSim
  STATE check_open_close LOOPED
    IF openLowValve = TRUE THEN
      LowValveOpened := TRUE;
    ELSE
      LowValveOpened := FALSE;
    END_IF
```

```
  END_STATE
END_PROCESS

PROCESS WaterLevelSim
  VAR CONSTANT
    OUTFLOW_RATE : REAL := 0.5;
    UPPER_LEVEL : REAL := 100;
    LOWER_LEVEL  : REAL := 0;
  END_VAR
  VAR
    coord : REAL := 0.0;
  END_VAR
  STATE check_open_close LOOPED
    IF openLowValve THEN
      coord := coord - OUTFLOW_RATE;
    END_IF
    IF openHighValve THEN
      coord := coord + OUTFLOW_RATE;
    END_IF
    IF coord >= UPPER_LEVEL THEN
      coord := UPPER_LEVEL;
    END_IF
    IF coord <= LOWER_LEVEL THEN
      coord := LOWER_LEVEL;
    END_IF
    IF coord = LOWER_LEVEL THEN
      atLow := TRUE;
    ELSE
      atLow := FALSE;
```

```
    END_IF
    IF coord = UPPER_LEVEL THEN
      atHigh := TRUE;
    ELSE
      atHigh := FALSE;
    END_IF
  END_STATE
END_PROCESS
END_PROGRAM

(*=============================================*)
(*=============================================*)
(*============== CONTROLLER ===================*)
(*=============================================*)
(*=============================================*)


PROGRAM Controller
PROCESS Init (* initial process *)
  STATE begin
    openHighGate := FALSE;
    openLowGate := FALSE;
    (* sluice valves *)
    openHighValve := FALSE;
    openLowValve := FALSE;
    (* traffic lights *)
    Low2ChmbrLight := FALSE;
    Chmbr2LowLight := FALSE;
    Chmbr2HighLight := FALSE;
```

```
    High2ChmbrLight := FALSE;

    START PROCESS Waiting4Ship;
    STOP;
  END_STATE
END_PROCESS


PROCESS Waiting4Ship
  STATE init
    (* TO BE EFFECTIVE: *)
    IF (shipInHigh AND shipInLow) THEN
      IF (atHigh OR atLow) THEN
        IF atHigh THEN
          START PROCESS DownControl;
          SET STATE CheckDown;
        END_IF
        IF atLow THEN
          START PROCESS UpControl;
          SET STATE CheckUp;
        END_IF
      ELSE
          START PROCESS DownControl;
          SET STATE CheckDown;
      END_IF
    ELSE (* PROCESS THE FIRST *)
      IF shipInHigh THEN
        START PROCESS DownControl;
        SET STATE CheckDown;
```

```
      END_IF
      IF shipInLow THEN
        START PROCESS UpControl;
        SET STATE CheckUp;
      END_IF
    END_IF
  END_STATE
  STATE CheckUp
    IF PROCESS UpControl IN STATE STOP THEN
      START PROCESS DownControl;
      SET STATE CheckDown;
    END_IF
  END_STATE
  STATE CheckDown
    IF PROCESS DownControl IN STATE STOP THEN
      START PROCESS UpControl;
      SET STATE CheckUp;
    END_IF
  END_STATE
END_PROCESS

PROCESS UpControl
  STATE begin
    START PROCESS LightOFF;
    IF shipInHigh OR shipInChmbr THEN
      START PROCESS Water2High;
      SET NEXT;
    ELSE
      STOP;
```

```
  END_IF
END_STATE
STATE Wait4High
  IF (PROCESS Water2High IN STATE STOP) THEN
    START PROCESS OpenHighGate;
    SET NEXT;
  END_IF
END_STATE
STATE Wait4Open
  IF (PROCESS OpenHighGate IN STATE STOP) THEN
    IF shipInChmbr THEN
      START PROCESS Chmbr2HighON;
      SET NEXT;
    ELSE
      START PROCESS High2ChmbrON;
      SET STATE Wait4ShipGetInChmbr;
    END_IF
  END_IF
END_STATE
STATE Wait4ShipGetOutChmbr
  IF (NOT shipInChmbr) THEN
    IF (NOT shipInHigh) THEN
      START PROCESS CloseHighGate; (*CLOSE GATE!*)
      SET NEXT;
    ELSE
      START PROCESS High2ChmbrON;
      SET STATE Wait4ShipGetInChmbr;
    END_IF
  END_IF
```

```
  END_STATE
  STATE Wait4ShipGetInChmbr
    IF (shipInChmbr) THEN
      START PROCESS CloseHighGate; (*CLOSE GATE!*)
      SET NEXT;
    END_IF
  END_STATE
  STATE Wait4Close
    IF (PROCESS CloseHighGate IN STATE STOP) THEN
      STOP;
    END_IF
  END_STATE
END_PROCESS

PROCESS DownControl
  STATE begin
    START PROCESS LightOFF;
    IF shipInLow OR shipInChmbr THEN
      START PROCESS Water2Low;
      SET NEXT;
    ELSE
      STOP;
    END_IF
  END_STATE
  STATE Wait4Low
    IF (PROCESS Water2Low IN STATE STOP) THEN
      START PROCESS OpenLowGate;
      SET NEXT;
    END_IF
```

```
END_STATE
STATE Wait4Open
  IF (PROCESS OpenLowGate IN STATE STOP) THEN
    IF shipInChmbr THEN
      START PROCESS Chmbr2LowON;
      SET NEXT;
    ELSE
      START PROCESS Low2ChmbrON;
      SET STATE Wait4ShipGetInChmbr;
    END_IF
  END_IF
END_STATE
STATE Wait4ShipGetOutChmbr
  IF (NOT shipInChmbr) THEN
    IF (NOT shipInLow) THEN
      START PROCESS CloseLowGate; (*CLOSE GATE!*)
      SET STATE Wait4Close;
    ELSE
      START PROCESS Low2ChmbrON;
      SET STATE Wait4ShipGetInChmbr;
    END_IF
  END_IF
END_STATE
STATE Wait4ShipGetInChmbr
  IF (shipInChmbr) THEN
    START PROCESS CloseLowGate; (*CLOSE GATE!*)
    SET NEXT;
  END_IF
END_STATE
```

```
  STATE Wait4Close
    IF (PROCESS CloseLowGate IN STATE STOP) THEN
      STOP;
    END_IF
  END_STATE
END_PROCESS


(* Gate/Valve Control Processes *)
PROCESS Water2Low
  STATE CheckConditions
    IF (atLow) THEN
      STOP;
    ELSE
      IF (NOT HighGateClosed) THEN
        ERROR;
      ELSE
        openLowValve := TRUE;
        openHighValve := FALSE;
        SET NEXT;
      END_IF
    END_IF
  END_STATE
  STATE CheckLevel
    IF (atLow) THEN
      openLowValve := FALSE;
      STOP;
    END_IF
  END_STATE
```

```
END_PROCESS

PROCESS Water2High
  STATE CheckConditions
    IF (atHigh) THEN
      STOP;
    ELSE
      IF (NOT LowGateClosed) THEN
        ERROR;
      ELSE
        openLowValve := FALSE;
        openHighValve := TRUE;
        SET NEXT;
      END_IF
    END_IF
  END_STATE
  STATE CheckLevel
    IF (atHigh) THEN
      openHighValve := FALSE;
      STOP;
    END_IF
  END_STATE
END_PROCESS

PROCESS OpenLowGate
  STATE CheckConditions
    IF (LowGateOpened) THEN
      STOP;
    ELSE
```

```
      IF (NOT atLow) THEN
        ERROR;
      ELSE
        openLowGate := TRUE;
        SET NEXT;
      END_IF
    END_IF
  END_STATE
  STATE CheckOpeness
    IF (LowGateOpened) THEN
      STOP;
    END_IF
  END_STATE
END_PROCESS

PROCESS CloseLowGate
  STATE CheckConditions
    IF (LowGateClosed) THEN
      STOP;
    ELSE
      openLowGate := FALSE;
      SET NEXT;
    END_IF
  END_STATE
  STATE CheckOpeness
    IF (LowGateClosed) THEN
      STOP;
    END_IF
  END_STATE
```

```
END_PROCESS

PROCESS OpenHighGate
  STATE CheckConditions
    IF (HighGateOpened) THEN
      STOP;
    ELSE
      IF (NOT atHigh) THEN
        ERROR;
      ELSE
        openHighGate := TRUE;
        SET NEXT;
      END_IF
    END_IF
  END_STATE
  STATE CheckOpeness
    IF (HighGateOpened) THEN
      STOP;
    END_IF
  END_STATE
END_PROCESS

PROCESS CloseHighGate
  STATE CheckConditions
    IF (HighGateClosed) THEN
      STOP;
    ELSE
      openHighGate := FALSE;
      SET NEXT;
```

```
    END_IF
  END_STATE
  STATE CheckOpeness
    IF (HighGateClosed) THEN
      STOP;
    END_IF
  END_STATE
END_PROCESS


(* Light Control Processes *)
PROCESS LightOFF
  STATE init
    Low2ChmbrLight := FALSE;
    Chmbr2LowLight := FALSE;
    Chmbr2HighLight := FALSE;
    High2ChmbrLight := FALSE;
    STOP;
  END_STATE
END_PROCESS
PROCESS Low2ChmbrON
  STATE init
    Low2ChmbrLight := TRUE;
    Chmbr2LowLight := FALSE;
    Chmbr2HighLight := FALSE;
    High2ChmbrLight := FALSE;
    STOP;
  END_STATE
END_PROCESS
```

```
PROCESS Chmbr2LowON
  STATE init
    Low2ChmbrLight := FALSE;
    Chmbr2LowLight := TRUE;
    Chmbr2HighLight := FALSE;
    High2ChmbrLight := FALSE;
    STOP;
  END_STATE
END_PROCESS
PROCESS High2ChmbrON
  STATE init
    Low2ChmbrLight := FALSE;
    Chmbr2LowLight := FALSE;
    Chmbr2HighLight := FALSE;
    High2ChmbrLight := TRUE;
    STOP;
  END_STATE
END_PROCESS
PROCESS Chmbr2HighON
  STATE init
    Low2ChmbrLight := FALSE;
    Chmbr2LowLight := FALSE;
    Chmbr2HighLight := TRUE;
    High2ChmbrLight := FALSE;
    STOP;
  END_STATE
END_PROCESS
END_PROGRAM
```

Приложение 4. Текст тезисов МНСК.

В ИАиЭ СО РАН активно разрабатываются языковые и инструментальные средства процесс-ориентированного программирования. Один из недавних результатов – язык poST, предназначенный для спецификации управляющих программ промышленными объектами в рамках концепции IEC 61131-3.

Язык poST изучается в курсе НГУ “Процесс-ориентированное программирование”. Для формирования практических навыков требуется организация лабораторных работ. Ранее лабораторные работы для курса “Процесс-ориентированное программирование” проводились на базе концепции виртуального объекта управления и реализованы в среде LabVIEW. В качестве языка программирования использовался Си-подобный язык Reflex, не используемый при программировании ПЛК. В связи с появлением языка poST и переориентацией исследований на IEC 61131-3 концепцию, более привлекательной выглядит организация лабораторных работ на базе одного из существующих IDE IEC 61131-3.

В рамках работы были рассмотрены способы организации лабораторных стендов для языков ПЛК и IDE, поддерживающих IEC 61131-3. Было обосновано использование средства CODESYS. Был определен список требований, основные из которых – наличие средств визуализации и мониторинга функционирования программы, использование poST в качестве языка описания и модели объекта управления, и алгоритма, независимость связей между алгоритмом управления и моделью объекта управления. В качестве пилотной была выбрана задача автоматизации трехэтажного лифта. Был создан прототип виртуального лабораторного стенда, в котором программы создаются с помощью онлайн транслятора языка poST в язык IEC 61131-3 ST (http://post2st.iae.nsk.su), а затем переносятся в среду CODESYS через нативный для IEC 61131-3 механизм XML Exchange. Разработанный подход обеспечивает сокращение трудоемкости организации учебного процесса и формирование

навыков работы с языками IEC 61131-3 и процесс-ориентированным языком poST.

В дальнейшем планируется определить подход к автоматической динамической верификации poST-программы, разработанной студентом-практикантом, и задокументировать процедуру создания нового виртуального лабораторного стенда.