МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2023 г.

Создание комплекса передачи данных для электропривода на отечественных компонентах

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-09.03.01.2023.148 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ П.О. Шабуров
«___»_____2023 г.

Автор работы,
студент группы КЭ-405
_____А.А. Чупрунова
«___»_____2023 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2023 г.

Челябинск-2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2023 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Чупруновой Анне Андреевне,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Создание комплекса передачи данных для электропривода на отечественных компонентах» утверждена приказом по университету от 25 апреля 2023 №753-13/12, приложение №308/10
2. **Срок сдачи студентом законченной работы: 14 июня 2023 г.**
3. **Исходные данные к работе:**

   Обеспечить основной функционал комплекса:

   1. Настроить возможность авторизации пользователя по паролю и фамилии.
   2. Настроить возможность просмотра текущих характеристик электропривода.
   3. Настроить возможность изменения текущих характеристик электропривода (не для всех пользователей).
   4. Для хранения данных использовать базу данных.
   5. Обеспечить взаимодействие между аппаратной и программной составляющими комплекса.

4. **Перечень подлежащих разработке вопросов:**
    1. Анализ предметной области.
    2. Определение требований.
    3. Проектирование.
    4. Реализация.
    5. Тестирование.

Дата выдачи задания: 1 декабря 2022 г.

Руководитель работы _____/ *П.О. Шабуров/*

Студент _____/ *А.А. Чупрунова /*

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Анализ предметной области	01.03.2023	
Определение требований	15.03.2023	
Проектирование	01.04.2023	
Реализация	01.05.2023	
Тестирование	15.05.2023	
Компоновка текста работы и сдача на нормоконтроль	22.05.2023	
Подготовка презентации и доклада	01.06.2023	

Руководитель работы _____*/П.О. Шабуров /*

Студент _____ */А.А. Чупрунова /*

АННОТАЦИЯ

Чупрунова А.А. Создание комплекса передачи данных для электропривода на отечественных компонентах. – Челябинск ЮУрГУ, ВШЭКН; 2023, 17 ил., 60 с., библиограф. список – 16 наим., 10 прил.

Целью выпускной квалификационной работы является создание комплекса передачи данных для электропривода на отечественных компонентах. Комплекс представляет собой кроссплатформенное приложение, устанавливаемое на персональный компьютер пользователя, и микроконтроллер, который является информационной частью электропривода. Комплекс предназначен для обмена информацией между программной и аппаратной его частями с использованием различных протоколов передачи данных.

При разработке комплекса был проведен обзор существующих аналогов, определены функциональные и нефункциональные требования к нему, выполнены этапы проектирования, реализации и тестирования.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Сегодня ни один технологический процесс в любой области мировой промышленности нельзя представить без применения электрического привода (далее - электропривода). Так, например, они используются в энергетике, металлургии, транспорте, сельском хозяйстве, нефтяной, горной и химической промышленностях и т.д. Именно применение электропривода позволяет автоматизировать производство, что существенно повышает эффективность любой компании, а также качество выпускаемой ей продукции или оказываемых услуг. Помимо этого, доля потребления электроприводами мировой вырабатываемой энергии составляет более 50% в зависимости от типа предприятия и устройства его производства. Такой процент не только говорит о важности их применения, но и заставляет задуматься о рациональности использования энергетических ресурсов и более подробно изучить процессы, происходящие внутри электропривода с целью их усовершенствования.

Электропривод представляет собой систему, с помощью которой приводятся в движение рабочие органы машин для выполнения определенного технологического процесса. Принцип его действия основан на преобразовании электрической энергии в механическую, механической энергии в электрическую и на управлении этим процессом во время полного цикла производства.

В систему электропривода входят следующие компоненты:

- Электрический двигатель (далее – электродвигатель). Является основным звеном электропривода. Внутри него происходит преобразование электрической энергии в механическую;
- Механический преобразователь. Отвечает за передачу выработанной механической энергии к рабочему органу управляемой машины;
- Система управления. Её функция заключается в запуске/остановке электропривода, получении его характеристик, регулировании параметров (таких как скорость вращения, сила тока, потребляемая мощность, температура двигателя, температура силовых ключей, момент двигателя, угол поворота и т.д.) с целью

достижения наибольшей производительности процесса при наименьших затратах электроэнергии.

Целью выполнения данной работы является создание комплекса передачи данных для электропривода на отечественных компонентах. Следовательно, предполагается разработать систему управления электроприводом. Она должна состоять из двух частей:

- Программная часть. Представляет собой кроссплатформенное программное обеспечение, содержащее пользовательский интерфейс, которое устанавливается на персональный компьютер клиента. Оно будет получать данные с аппаратной части электропривода, преобразовывать их и выводить на экран в понятном для пользователя виде. Пользователь будет иметь возможность их анализировать и при необходимости изменять, передавая на аппаратную часть.
- Аппаратная часть. Эта часть будет представлена микроконтроллером. Он отвечает непосредственно за получение данных с электропривода и их отправку на него. Также микроконтроллеры являются программируемыми устройствами, что позволяет легко настраивать и обновлять программное обеспечение в соответствии с требованиями и изменениями в системе управления электропривода.

Для достижения поставленной цели были обозначены следующие задачи:

1. Анализ существующих аналогов и выявление их преимуществ и недостатков.
2. Формирование функциональных и нефункциональных требований.
3. Проектирование приложения.
4. Реализация программно-аппаратной части комплекса.
5. Тестирование комплекса.

Актуальность данной разработки заключается в том, что сегодня любое производство использует в своей деятельности электроприводы. Управление электроприводом при помощи системы передачи данных позволит оптимизировать технологический процесс на любом предприятии, сделать его максимально эффективным и при этом уменьшить энергетические расходы, что также является приоритетной задачей. Также системы управления электроприводами обладают

высокой гибкостью и адаптивностью к различным условиям и требованиям процессов. Они могут легко настраиваться и программироваться для разных режимов работы, различных нагрузок и параметров. Это обеспечивает возможность быстрой настройки и перенастройки системы в соответствии с изменяющимися потребностями и требованиями производства.

Кроме того, сегодня в России активно происходит импортозамещение и ведется разработка продуктов на отечественных компонентах. Каждый такой продукт нуждается в специально разработанном для него комплексе, что позволит также отказаться от применения зарубежного программного обеспечения.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Появление регулируемых электроприводов породило большое разнообразие в технологиях управления ими. Для этого применялись двигатели постоянного тока с отличающимися схемами возбуждения: параллельная, последовательная, смешанная или независимая. Также использовались и другие типы двигателей: асинхронный с фазным ротором, двигатель Бушеро и другие. Находили применение и методы регулирования магнитного поля при помощи реостата.

Сегодня существуют комплексы передачи данных для электропривода на основе микроконтроллеров. Такой вариант управления набирает все большую популярность по мере развития микроэлектроники во всем мире. Это связано с тем, что специальные устройства управления встроены на плату микроконтроллера и не требуют дополнительной аппаратуры для решения задач. Также эффективность такого метода управления объясняется широкой системой команд центрального процессора, которая позволяет решать большинство типовых задач.

Сейчас большинство систем управления электроприводом реализовано на базе микроконтроллеров следующих производителей: Intel, Motorola, Siemens, AnalogDevices, STMicroelectronics и TexasInstruments. Однако, существуют решения и от отечественных компаний. Далее они будут рассмотрены более подробно.

1.1 Обзор аналогов

Общей концепцией развития систем управления электроприводами являются модульность, универсальность и широкая функциональность, которые позволяют быстро вписаться в современное производство. Выбрать качественное ПО, которое подойдет конкретному производству, позволяет его анализ по таким критериям, как аппаратная, программная и функциональная совместимость с требованиями заказчика. Поэтому можно выделить следующие пути развития данного направления разработок [1]:

1. Создание единой базы данных настроек для большого числа аппаратных средств в т.ч. электродвигателей, датчиков положения и т.д. Это позволит в разы ускорить настройку привода и избавит инженеров производства от рутинной работы.

2. Разработка автоматической функции поиска оптимальных настроек системы управления электроприводом под конкретный двигатель.

3. Разработка клиент-серверной архитектуры программных средств управления электроприводом. Это позволит удаленно управлять электроприводом и уменьшить время простоя машины. Однако, такая реализация требует надежной сетевой защиты.

4. Разграничение прав доступа к функциональным возможностям. Создание нескольких уровней доступа избавит от вмешательства в работу системы неквалифицированной рабочей силы, что положительно влияет на безопасность использования электропривода в целом.

5. Внедрение системы оповещений о сбоях в работе электропривода, экстренная остановка и вывод сообщений об ошибке.

Всего существует несколько вариантов организации управления электроприводом с персонального компьютера (ПК) [2]. Каждое предприятие выбирает удобное для себя решение, основываясь на сложности проекта, количестве задействованных электроприводов и задачах, решаемых ими.

1.1.1 Развертывание системы управления электроприводом на базе OPC-серверов

OPC (OLE for Process Control, где OLE — Object Linking and Embedding) — набор стандартов, описывающих универсальный фиксированный интерфейс обмена данными с промышленными устройствами. С помощью OPC можно взаимодействовать с оборудованием на уровне фиксированного и описанного в стандарте набора функций.

В качестве примера системы с поддержкой этого стандарта можно привести решения компании Siemens на базе WinCC-сервера. Она позволяет решить задачи

мониторинга и управления, а также предоставляет возможность визуализации техпроцесса.

Но у данного решения есть свои минусы, которые являются существенными для большинства потребителей:

- Стоимость оборудования, которое поддерживает данный стандарт. Зачастую цена такого оборудования в несколько раз превышает стоимость оборудования без поддержки стандарта;
- Потребность в отдельном приобретении программного обеспечения. Этот минус усложняет процесс внедрения оборудования на производство и влечет за собой дополнительные денежные расходы;
- Сложность использования. Внедрение этой системы на производство требует наличие специализированных серверов и интерфейсов. Также необходима постоянная его поддержка и наличие обученного человека на предприятии или проведение курсов повышения квалификации среди сотрудников.

Таким образом, в случае, когда необходимо управлять всего лишь несколькими двигателями и снимать показания с группы датчиков, развертывание полноценного решения на базе OPC может быть неоправданно.

1.1.2 Использование решений на базе SDK (Software Development Kit), поставляемых производителем

Такие системы гораздо дешевле предыдущих и проще в использовании. Кроме того, зачастую они уже идут в комплекте с оборудованием, что ускоряет процесс внедрения комплекса управления в производство.

Наиболее популярными такими системами в настоящее время являются OsmModbusControl SDK [3] и ST Motor Control Workbench [4], работающие с контроллерами серии OSM и STM соответственно.

OsmModbusControl SDK позволяет работать с устройствами протоколу ModbusRTU с помощью двух библиотек:

- OsmModbusController.dll. Отвечает за работу с регистрами устройств;

• OsmStepMotorController.dll. Содержит высокоуровневые команды для управления двигателем.

OsmModbusControl SDK работает под ОС Windows, Linux и MacOS. Все управляющие команды перечислены в документации и позволяют полностью контролировать работу электропривода в реальном времени [5].

ST Motor Control Workbench совместим только с ОС Windows. Этот набор инструментов для разработки содержит в себе библиотеку для управления синхронным двигателем с постоянными магнитами и графический пользовательский интерфейс для настройки параметров библиотеки [6].

ST Motor Control Workbench позволяет пользователям регулировать более 100 параметров электродвигателя, среди которых скорость вращения, частота, ток, режимы управления, ограничение скорости, время ожидания, время перехода от одного режима к другому и т.д. [7].

Таким образом, программное обеспечение управления электроприводом на базе SDK, поставляемого производителем, отличается широким функционалом, быстротой введения в производство и наглядностью при использовании, что является преимуществом данного ПО. Однако, использование таких систем требует привязки к конкретному производителю оборудования и знания инженерами на производстве того языка высокого уровня, который применяется в среде разработки, что может вызывать трудности.

1.1.3 «РИТМ.Электропривод»

Разработка комплекса полунатурного моделирования РИТМ (КПМ «РИТМ») на базе инновационного центра Сколково [8].

Является программным комплексом, разработанным специально для инженеров на различных типах производства. Предназначен для автоматизации настройки параметров систем автоматического управления электропривода. В программе реализованы автоматический и ручной режимы настройки параметров регуляторов разного типа в контурах системы управления электрическим приводом на базе трех широко используемых типов двигателей.

Данный продукт совместим с ОС Windows 10, Windows 7 Service Pack 1, Windows Server 2019 и Windows Server 2016. Для установки данного ПО необходимо 2 Гб на жестком диске компьютера и не менее 4 Гб оперативной памяти.

Сведем результаты аналитического обзора аналогов в таблицу 1.

Таблица 1 – Сравнение аналогов

	Simatic WinCC	**OsmModbusControl SDK**	**ST Motor Control Workbench**	**РИТМ.Электропривод**
Кроссплатформенность	+	-	-	-
Доступность внедрения	-	+	+	-
Отсутствие привязки к производителю оборудования	-	-	-	+
Широкий функционал	+	+	+	+
Простота использования	-	-	+	+

Таким образом, «РИТМ.Электропривод» по своим функциональным возможностям и интерфейсу является наиболее близким к продукту, разрабатываемому в рамках данной ВКР. Однако, он также имеет ряд недостатков, среди которых можно выделить отсутствие поддержки ОС Linux, а также недоступность разделения ролей. Эти характеристики, также являются главным отличием продукта «РИТМ.Электропривод» от создаваемого в рамках данной выпускной квалификационной работы (ВКР) программного обеспечения (ПО).

1.2 Анализ основных технологических решений

1.2.1 Выбор языка программирования

При выборе языка программирования для разработки кроссплатформенного веб-приложения рассматривались следующие варианты:

- JavaScript;
- Python;

- Ruby;
- Java;
- C#.

Каждый язык имеет свои преимущества и недостатки. По результатам их рассмотрения был сделан выбор в пользу наиболее подходящего языка.

JavaScript – язык программирования, поддерживающий функции объектно-ориентированного программирования. Широко применяется для разработки веб-приложений.

Преимущества:

- Позволяет вести разработку под различные веб-браузеры;
- Широкая популярность по всему миру, что подразумевает большое сообщество программистов, способных помочь при решении проблемы;
- Интерактивность.

Недостатки:

- Уязвимость в безопасности;
- Несовершенство системы управления памятью;
- Различие версий JavaScript, поддерживаемых различными браузерами может вызвать некорректную работу приложения в зависимости от используемого браузера.

Python – язык программирования высокого уровня, решающий широкий спектр задач в программировании.

Преимущества:

- Читабельность кода и легкость изучения синтаксиса;
- Широкая популярность по всему миру, что подразумевает большое сообщество программистов, способных помочь при решении проблемы;
- Большое количество библиотек, ускоряющих процесс разработки;
- Масштабируемость.

Недостатки:

- Низкая производительность и скорость работы в отличие от языков конкурентов. Существенный минус при работе с электроприводом;
- Сильная зависимость от интерпретатора.

Ruby – интерпретируемый, рефлективный язык программирования высокого уровня.

Преимущества:

- Большое количество библиотек, ускоряющих процесс разработки.

Недостатки:

- Сложность изучения синтаксиса;
- Низкая популярность в сравнении с языками-конкурентами, не такое широкое пользовательское сообщество;
- Низкая производительность и скорость работы в отличие от языков конкурентов. Существенный минус при работе с электроприводом.

Java - типизированный язык программирования высокого уровня, поддерживающий принципы объектно-ориентированного программирования.

Преимущества:

- Совместимость с различными платформами;
- Широкая популярность по всему миру, что подразумевает большое сообщество программистов, способных помочь при решении проблемы;
- Хорошая система безопасности.

Недостатки:

- Сложность разработки из-за объемного кода;
- Требовательность к ресурсам памяти и процессорного времени.

C# - язык программирования общего назначения, поддерживающий принципы объектно-ориентированного программирования.

Преимущества:

- Широкий выбор средств для разработки (т.к. язык C# связан с платформой .NET);

- Высокая скорость работы и производительность;
- Широкая популярность по всему миру, что подразумевает большое сообщество программистов, способных помочь при решении проблемы;
- Большое количество документации.

Недостатки:

- Зависимость от фреймворков.

В целом, C# представляет собой мощный и эффективный язык программирования для веб-разработки, особенно в экосистеме Microsoft и платформе .NET. Именно он был выбран для разработки программной части комплекса в рамках данной работы.

1.2.2 Выбор системы управления базой данных

Для работы с веб-приложением существуют различные системы управления базами данных (далее - СУБД). Каждая из них имеет свои преимущества и недостатки. При выборе СУБД для программной части комплекса рассматривались следующие варианты:

- MySQL;
- PostgreSQL;
- Microsoft SQL Server.

MySQL - СУБД от компании Oracle.

Преимущества:

- Широкий функционал;
- Безопасность;
- Бесплатное использование;
- Кроссплатформенность;
- Скорость обработки данных.

Недостатки:

- Сложность обработки больших объемов данных;
- Отсутствие полноценной поддержки стандарта SQL.

Microsoft SQL Server - СУБД от компании Microsoft.

Преимущества:

- Высокая производительность;
- Высокая масштабируемость;
- Широкое сообщество разработчиков по всему миру;
- Богатый функционал.

Недостатки:

- Высокая цена;
- Необходимость развертывания собственного сервера;
- Работает только с ОС Windows;
- Сложность в использовании для новичков.

PostgreSQL - объектно-реляционная СУБД.

Преимущества:

- Безопасность;
- Масштабируемость;
- Бесплатное использование.

Недостатки:

- Низкая производительность при чтении из базы.

Каждая из рассмотренных СУБД имеет свои достоинства и недостатки. Для выбора оптимальной СУБД следует определить какие ее параметры являются определяющими для разрабатываемого комплекса:

- Безопасность. Поскольку приложение будет использоваться на закрытом предприятии и доступ к нему будет иметь строго ограниченное количество человек, защита хранимых данных является определяющим фактором;
- Скорость работы. Скорость реакции на команды оператора электропривода также играет важную роль;
- Кроссплатформенность;
- Бесплатное использование;

• Простота в использовании. Этот фактор позволяет сэкономить время и ресурсы на обучение сотрудников по обращению с СУБД.

Исходя из рассмотренных выше факторов, оптимальной СУБД для использования в данной разработке является MySQL от компании Oracle.

1.2.3 Выбор веб-сервера

Для реализации серверной части приложения был выбран локальный сервер «Денвер», с помощью которого можно вести разработку на домашнем адресе сети без выхода в интернет.

На выбор именно этого веб-сервера повлияли следующие его преимущества:

• Проект разработан российской группой программистов, что гарантирует постоянную его поддержку и независимость от зарубежных компаний;

• Веб-сервер прост в установке, имеет интуитивно понятный набор файлов и инструкции, позволяющие быстро разобраться с тем, как с ним работать;

• Маленький размер установочных файлов, что облегчает работу системе (особенно актуально для рабочих компьютеров на предприятиях, где крайне важна их производительность);

• Большое сообщество русскоязычных программистов, взаимодействие с которыми облегчает и ускоряет работу с веб-сервером.

Таким образом, веб-сервер «Денвер» является наиболее оптимальным вариантом для реализации серверной части приложения в комплексе.

1.2.4 Выбор микроконтроллера

Так как целью выпускной квалификационной работы является создание комплекса передачи данных для электропривода именно на отечественных компонентах, то в рамках выполнения работы был произведен обзор рынка отечественной электронной промышленности.

В результате обзора было выявлено, что на данный момент в Российской Федерации только группа компаний «Микрон» занимается производством

микроконтроллеров на отечественных компонентах. Также было установлено, что производство микроконтроллеров выполняется исключительно на заказ.

По этой причине принято решение, что для проектирования, реализации и тестирования аппаратной части комплекса будут использоваться зарубежные микроконтроллеры.

Выбор проводился среди микроконтроллеров следующих производителей:

- STMicroelectonics;
- Arduino;
- Texas Instruments.

После проведения анализа производителей была выбрана отладочная плата STM32F3discovery на базе микроконтроллера STM32F303VCT6. На выбор повлияли следующие ее характеристики:

- Доступность;
- Большое количество периферии;
- Широкая линейка средств программирования и разработки под семейство таких микроконтроллеров;
- Популярность по всему миру;
- Большое количество документации [9];
- Наличие пользовательского USB порта;
- Возможность подключения внешних устройств.

Выбранная отладочная плата представлена на рисунке 1.

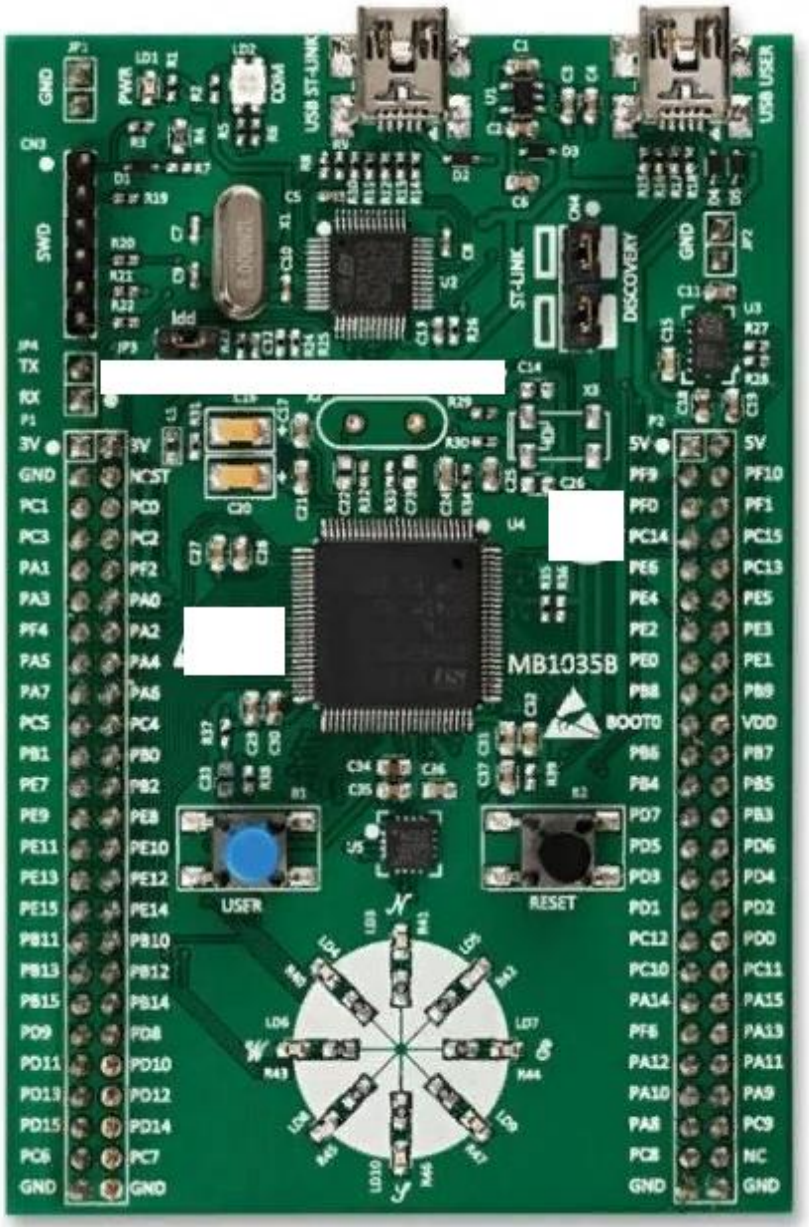


Рисунок 1 – Выбранная отладочная плата

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 Функциональные требования

1. Получение на вход от оператора параметров, регулирующих следующие характеристики электропривода:

- Сила тока;
- Потребляемая мощность;
- Скорость вращения;
- Температура двигателя;
- Температура силовых ключей;
- Момент двигателя;
- Угол поворота (для синхронного двигателя).

2. Считывание с информационной части электропривода его характеристик, описанных в п.1.

3. Построение графиков по полученным характеристикам.

4. Авторизация пользователей по их имени, фамилии и паролю.

5. Выбор типа используемого двигателя (синхронный или асинхронный), который влияет на список отображаемых параметров.

6. Запись параметров электропривода в базу данных для дальнейшего их использования.

7. Запись информации о новых сотрудниках в базу данных и удаление оттуда уволившихся.

8. Уведомление на окне авторизации при неправильном вводе фамилии, имени или пароля сотрудником.

2.2 Нефункциональные требования

1. Готовое ПО должно быть кроссплатформенным и работать на ОС Microsoft Windows и Linux;

2. ПО должно принимать и передавать данные от аппаратной части комплекса по следующим интерфейсам физического уровня: RS-232.

3. Оперативная память: 4 GB.

4. Использование безопасной базы данных.

2.3 Требования к пользователям

Пользователями являются операторы, регулирующие работу электропривода. Запись пользователей в базу данных и, соответственно, регистрация их в системе осуществляется data-специалистом предприятия. Пользователи не имеют прав для самостоятельной регистрации. Для работы с системой пользователи вводят свои имя, фамилию и пароль, выданный им data-специалистом предприятия.

3 ПРОЕКТИРОВАНИЕ

3.1 Описание процесса

Процесс взаимодействия пользователя с системой начинается с его авторизации. Сотрудник предприятия вводит свои имя, фамилию и пароль в окно авторизации. Далее эти данные с направляются в базу данных и сравниваются с имеющимися в ней значениями. При нахождении совпадающих значений пользователь авторизовывается в системе.

Далее происходит переход на страницу с выбором типа используемого двигателя (асинхронный или синхронный). Выбор происходит из двух значений выпадающего списка. Для каждого типа двигателя указываются соответствующие ему параметры.

В зависимости от должности пользователя система предоставляет разные варианты представления страницы. Всего пользователи поделены на две роли (в зависимости от должностей): старший инженер и младший инженер.

Старший инженер может получать значения с электропривода, изменять их и направлять обратно. Для того, чтобы система получила данные, управляющий контроллер должен физически их отправить на сервер через api или форму, а для получения текущей конфигурации должны сделать соответствующие api запросы. Младшему инженеру доступно только получение данных. Пользователь, работающий под ролью младшего инженера, не может менять параметры электропривода.

У пользователей с ролью старшего инженера есть возможность сохранять требуемые параметры настройки электропривода в базу данных для более быстрой его настройки в дальнейшем.

Для наглядного представления описания процесса были спроектированы диаграммы с использованием нотации UML [10]. Такие диаграммы позволяют детально рассмотреть разрабатываемый процесс со всех сторон [11]. На рисунке 2 изображена диаграмма последовательностей, на рисунке 3 - диаграмма прецендентов, на рисунке 4 - диаграмма деятельности.

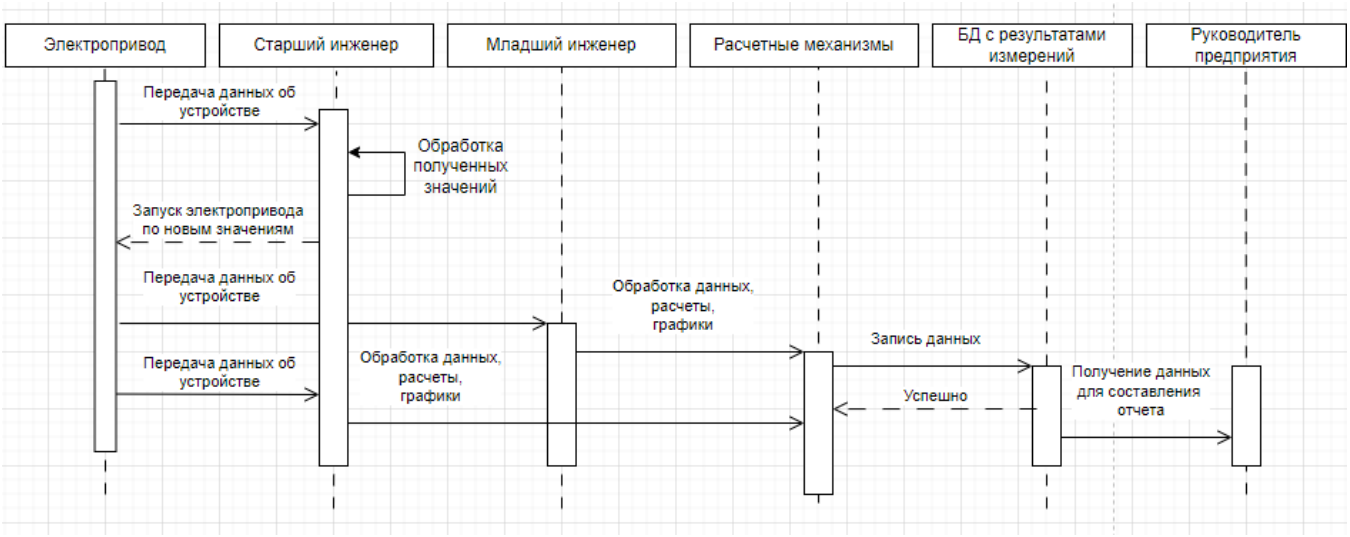


Рисунок 2 - Диаграмма последовательностей

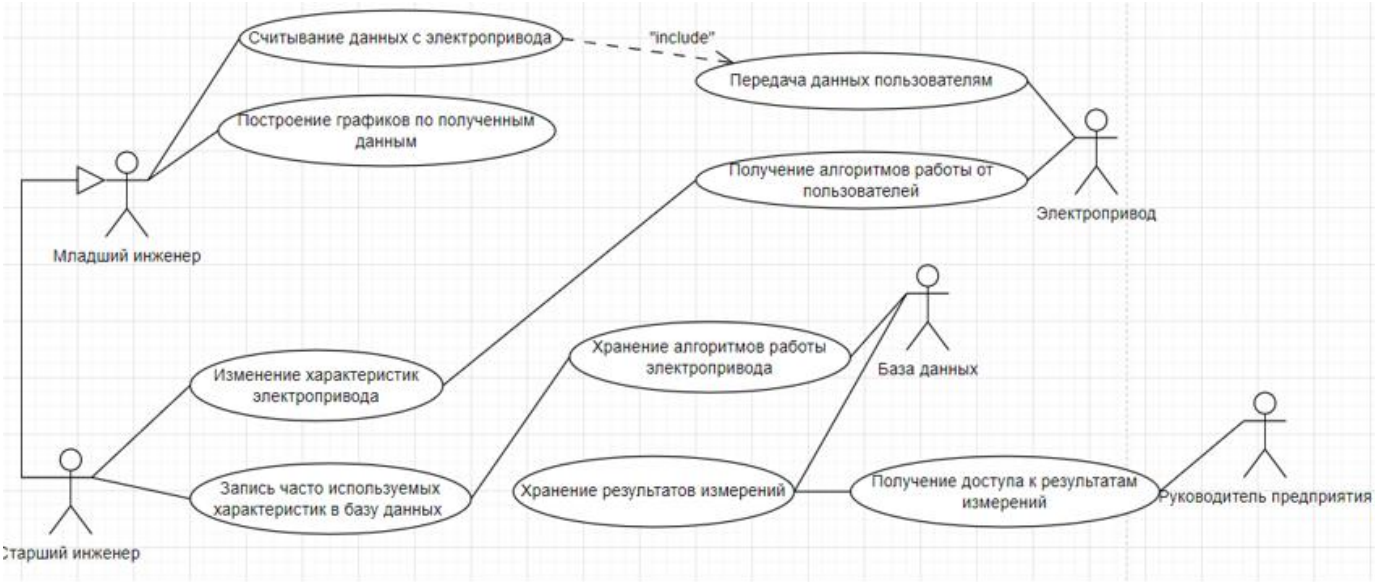


Рисунок 3 - Диаграмма прецендентов

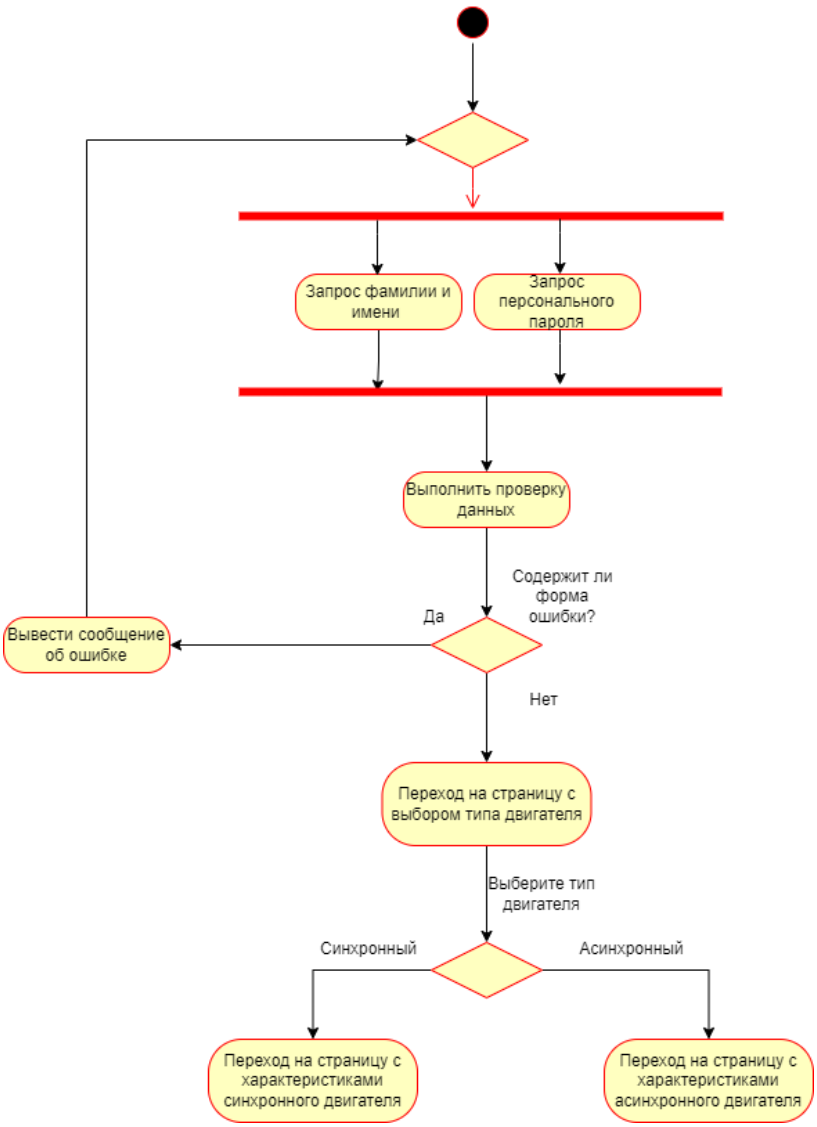


Рисунок 4 - Диаграмма деятельности

3.2 Проектирование программной части

Программная часть разрабатываемого приложения представляет собой серверную и клиентскую части.

Серверная часть состоит из базы данных и веб-сервера, на котором происходит обработка данных.

Для реализации клиентской части комплекса передачи данных на языке C# был выбран архитектурный паттерн MVC (Model View Controller) [12]. Использование этого паттерна при разработке позволяет разделить данные, пользовательский интерфейс и управляющие компоненты приложения на три независимые части.

Преимущество такой архитектуры в том, что такое разделение позволяет проще ориентироваться в коде и быстрее находить ошибки и создавать новые страницы и команды. Разделение на три независимые части позволяет разработчику изменять каждую из них в случае необходимости, не опасаясь, что другие могут от этого повредиться.

Вид (View) - представляет собой пользовательский интерфейс.

Модель (Model) - метод, который содержит в себе основные операции (т.е. функционал).

Контроллер (Controller) - взаимодействует с пользователем, обрабатывает его действия и считывает данные.

На рисунке 5 представлено схематичное изображение взаимодействия этих компонентов.

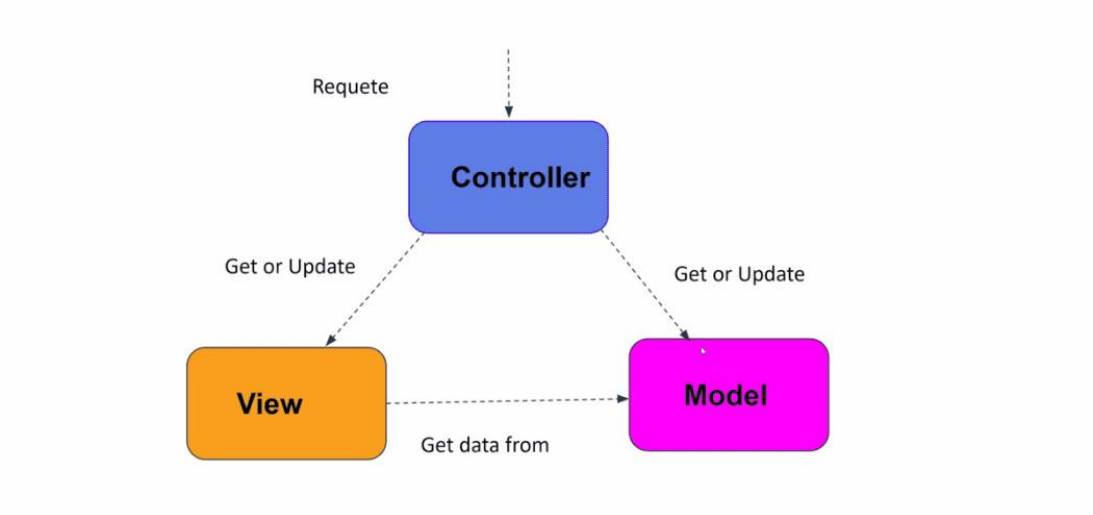


Рисунок 5 - Схематичное изображение взаимодействия компонентов модели MVC

Для разработки с применением этого паттерна использовалась среда разработки Visual Studio и фреймворк ASP.NET Core.

3.3 Проектирование аппаратной части

Электропривод состоит из электродвигательного, механического, передаточного, электрического преобразовательного и информационно-управляющего устройств.

Под аппаратной частью комплекса передачи данных в рамках данной работы подразумевается информационно-управляющая составляющая электропривода. Она состоит из электронных и микропроцессорных систем управления, обеспечивающих автоматическое выполнение заданного технологического процесса [13].

Устройство электропривода, с которым велась работа в рамках проекта, представлено на рисунке 6. Информационный канал электропривода находится вне зоны пунктирных линий.

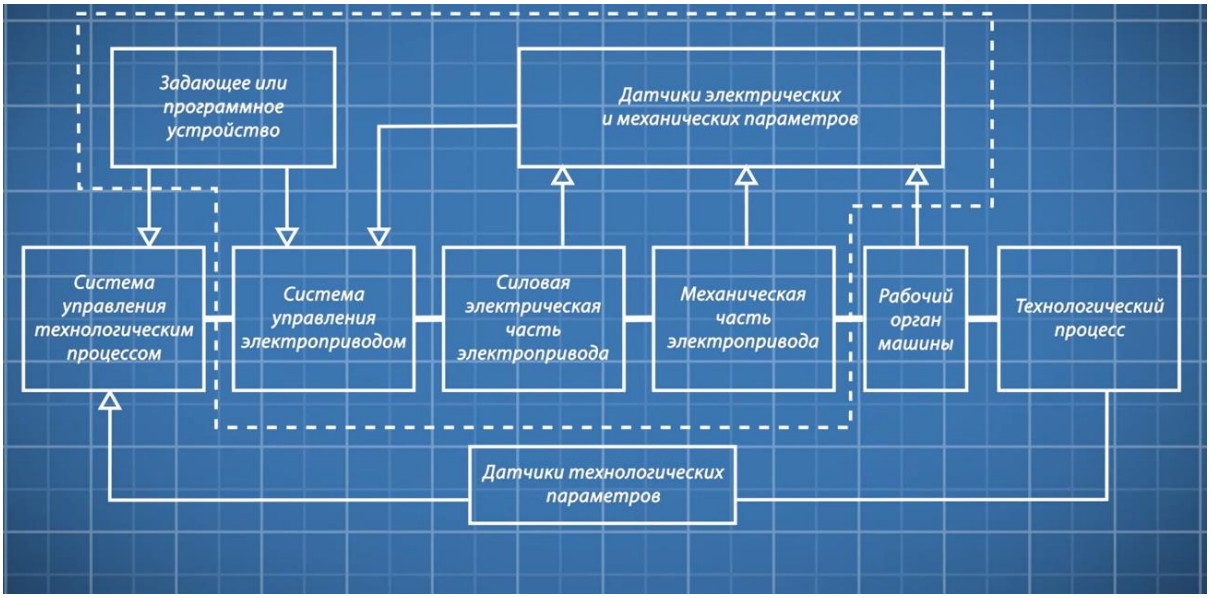


Рисунок 6 - Устройство электропривода

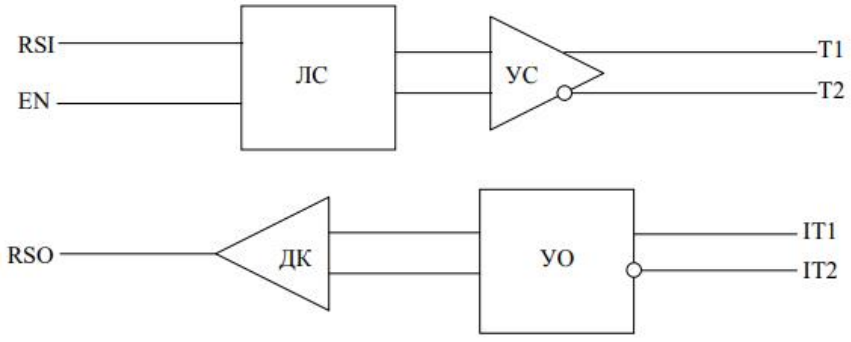
Для проектирования аппаратной части комплекса в данной работе используется микросхема, которая является универсальным асинхронным приемопередатчиком с гальванической развязкой. Именно с её помощью данные будут передаваться с приложения на компьютере на систему управления электроприводом и обратно. Отбор микросхем для проектирования аппаратной части проходил по следующим критериям:

1. Отечественный производитель.
2. Большое количество поддерживаемых протоколов передачи данных.
3. Независимые линии приема и передачи.
4. Возможность одновременной работы на прием и передачу данных.

После анализа рынка отечественной микроэлектроники и поиска микросхемы с нужными характеристиками была выбрана микросхема 5559ИН12У от предприятия ОАО «Научно-производственное объединение «Физика»» [14].

Микросхема 5559ИН12У является универсальным асинхронным приемопередатчиком с гальванической трансформаторной развязкой. Микросхема поддерживает линию любого асинхронного протокола передачи данных, например, RS-232, RS-422, RS485, UART. Линии приема и передачи независимые.

На рисунке 7 изображена структурная схема выбранной в рамках проектирования микросхемы.



ЛС – входная логическая схема передатчика;
УС – выходной усилитель передатчика;
УО – входной усилитель приемника;
ДК – дифференциальный компаратор приемника.

Рисунок 7 – Структурная схема микросхемы 5559ИН12У

4 РЕАЛИЗАЦИЯ

Для реализации был выбран язык программирования C#, СУБД - MySQL, веб-сервер «Денвер» и отладочная плата STM32F3discovery на базе микроконтроллера STM32F303VCT6.

4.1 База данных

База данных под названием «Electric_drive» находится по адресу локальной сети и содержит в себе две таблицы:

• Employees. Таблица хранит информацию о сотрудниках и содержит следующие поля: id, name, position, password. Структура таблицы employees представлена на рисунке 8;

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно	Действие
☐	1	id	int(11)			Нет	Нем	AUTO_INCREMENT	Изменить Удалить
☐	2	name	varchar(255)	utf8_general_ci		Нет	Нем		Изменить Удалить
☐	3	position	varchar(255)	utf8_general_ci		Нет	Нем		Изменить Удалить
☐	4	password	varchar(255)	utf8_general_ci		Нет	Нем		Изменить Удалить

Рисунок 8 - Структура таблицы employees

• Measurements. Таблица хранит информацию о характеристиках электропривода и содержит следующие поля: id, employee_id, temperature, rotation_speed, angle, engine_torque, amperage. Структура таблицы measurements представлена на рисунке 9.

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно	Действие
☐	1	id	int(11)			Нет	Нем	AUTO_INCREMENT	Изменить Удалить
☐	2	employee_id	int(11)			Нет	Нем		Изменить Удалить
☐	3	temperature	float			Нет	Нем		Изменить Удалить
☐	4	rotation_speed	float			Нет	Нем		Изменить Удалить
☐	5	angle	float			Нет	Нем		Изменить Удалить
☐	6	engine_torque	float			Нет	Нем		Изменить Удалить
☐	7	amperage	float			Нет	Нем		Изменить Удалить

Рисунок 9 - Структура таблицы measurements

Общая схема базы данных представлена на рисунке 10.

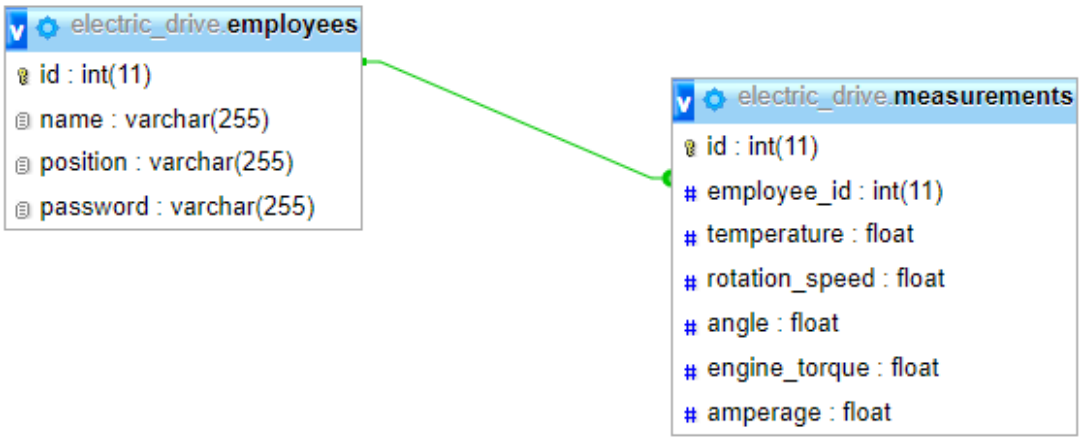


Рисунок 10 – Схема базы данных

4.2 Модели (Models)

Приложение содержит семь моделей:

- Employee - реализует класс Employee, в который передается информация о сотрудниках из базы данных. Листинг модели представлен в приложении А;
- EngineDetailsModel - реализует класс EngineDetailsModel, который содержит информацию о характеристиках асинхронного двигателя. Листинг модели представлен в приложении Б;
- EngineTypeModel - реализует класс EngineTypeModel, который содержит текст выпадающего списка на представлении выбора типа двигателя. Листинг модели представлен в приложении В;
- ErrorViewModel - возвращает ошибку при несоответствии введенных данных наборам данных из СУБД;
- LoginModel - реализует класс LoginModel, который отвечает за авторизацию. Листинг модели представлен в приложении Г;
- MyDbContext - отвечает за подключение к базе данных;
- SyncEngineModel - реализует класс SyncEngineModel, который содержит информацию о характеристиках синхронного двигателя. Листинг модели представлен в приложении Д.

4.3 Представления (Views)

Приложение содержит четыре представления. Представление отвечает за отображение интерфейса приложения пользователю. Всего пользователю доступны четыре страницы:

- Login - представление авторизации;
- EngineType - представление выбора типа используемого двигателя;
- EngineDetails - представление с параметрами асинхронного двигателя;
- SyncEngineDetails - представление с параметрами синхронного двигателя.

Все листинги представлений расположены в приложении Е.

4.4 Контроллеры (Controllers)

Контроллер отвечает за корректную работу представления и связь его с моделью. Количество контроллеров в приложении равно количеству представлений и составляет четыре:

- LoginController - обрабатывает страницу авторизации. Листинг контроллера представлен в приложении Ж;
- EngineTypeController - обрабатывает страницу выбора типа используемого двигателя. Листинг контроллера представлен в приложении К;
- EngineController - обрабатывает страницу с параметрами асинхронного двигателя. Листинг контроллера представлен в приложении Л;
- SyncEngineController - обрабатывает страницу с параметрами синхронного двигателя. Листинг контроллера представлен в приложении М.

4.5 Интерфейс пользователя

Работа с программной частью комплекса начинается со страницы авторизации, куда пользователь должен ввести свои имя, фамилию и пароль. Страница авторизации представлена на рисунке 11.

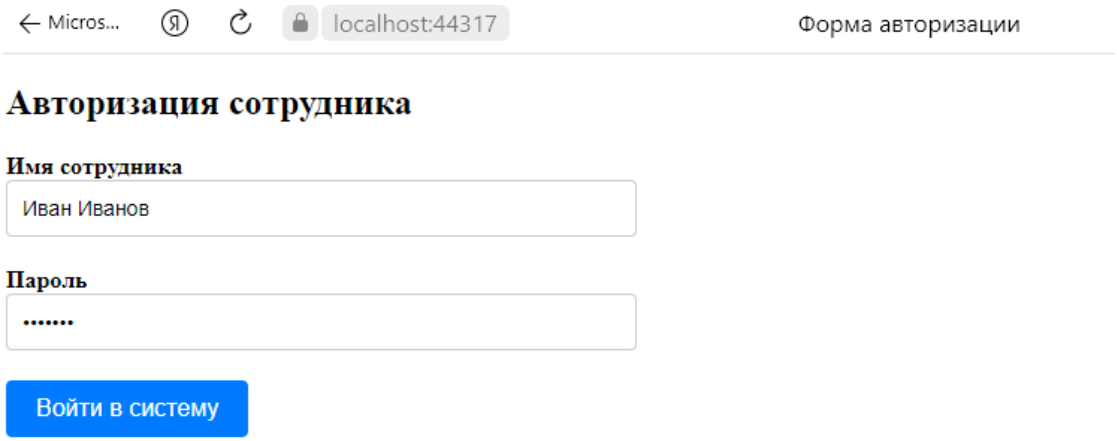


Рисунок 11 – Страница авторизации

При корректном вводе данных пользователь попадает на страницу с выбором типа используемого двигателя, интерфейс которой представлен на рисунке 12.

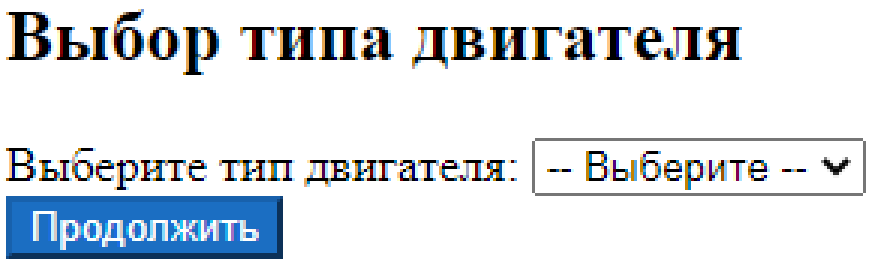


Рисунок 12 – Страница с выбором типа используемого двигателя

Варианты выпадающего списка представлены на рисунке 13.

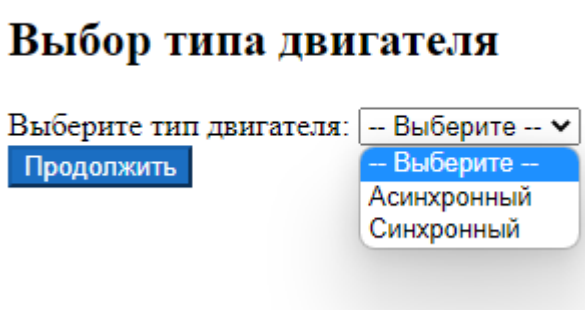


Рисунок 13 – Варианты выпадающего списка

При выборе синхронного типа двигателя пользователь перемещается на страницу с его параметрами, интерфейс которой представлен на рисунке 14.

Параметры синхронного двигателя

Сила тока:	0
Потребляемая мощность:	0
Скорость вращения:	0
Температура двигателя:	0
Температура силовых ключей:	0
Момент двигателя:	0
Угол поворота:	0

Рисунок 14 – Страница с параметрами синхронного двигателя

По умолчанию значения всех параметров равны нулю. Для получения актуальных значений электропривода следует нажать кнопку на используемой отладочной плате. В таком случае значения в таблице изменятся так, как показано на рисунке 15.

Параметры синхронного двигателя

Сила тока:	15
Потребляемая мощность:	1,4
Скорость вращения:	120
Температура двигателя:	90
Температура силовых ключей:	100
Момент двигателя:	15,6
Угол поворота:	30

Рисунок 15 – Страница с новыми параметрами синхронного двигателя

Аналогичный интерфейс и логику работы имеет страница с параметрами асинхронного типа двигателя.

4.6 Аппаратная часть

Аппаратная часть реализовывалась и тестировалась на плате STM32F3Discovery, который подключался к персональному компьютеру с помощью USB-кабеля так, как показано на рисунке 16.

Программа написана таким образом, что позволяет работать с любой микросхемой, поскольку осуществляется побитовая передача данных [15].

Рисунок 16 – Подключение отладочной платы к компьютеру

5 ТЕСТИРОВАНИЕ

Для проверки результатов реализации программно-аппаратного комплекса передачи данных на соответствие требованиям, обозначенным в главе 2, были проведены следующие виды тестирования:

1. Функциональное тестирование. Выполнялась проверка корректности работы всех необходимых функций. На рисунке 17 представлен результат тестирования страницы авторизации при вводе неправильного пароля для входа. Тестирование было автоматизировано и проводилось при помощи Unit тестов из библиотеки OpenQA.Selenium [16].

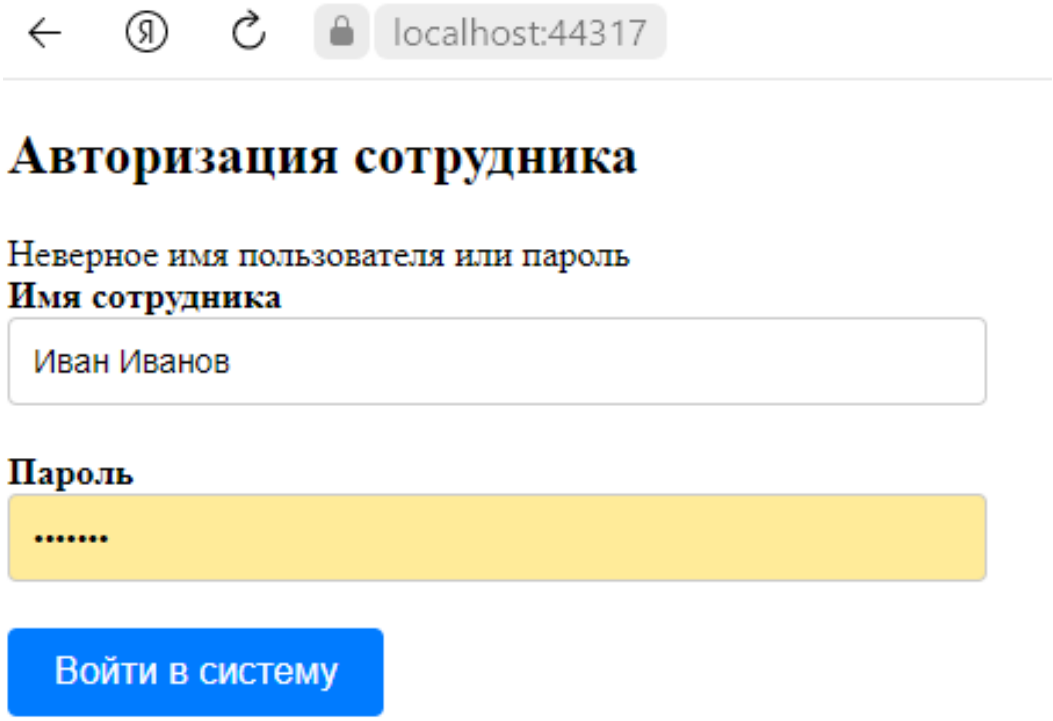


Рисунок 17 – Тестирование формы авторизации сотрудника

2. Системное тестирование. В процессе проведения данного вида тестирования проходила проверка производительности системы.

3. Тестирование в браузере. Программная часть комплекса проверялась на совместимость с такими браузерами, как Google Chrome, Yandex и Microsoft Edge.

Все тесты были пройдены успешно.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной выпускной квалификационной работы был спроектирован комплекс передачи данных для электропривода на отечественных компонентах.

Каждый из этапов, заявленный в календарном плане, был выполнен.

Для достижения поставленной цели были решены следующие задачи:

1. Анализ существующих аналогов и выявление их преимуществ и недостатков.
2. Формирование функциональных и нефункциональных требований.
3. Проектирование.
4. Реализация.
5. Тестирование.

В будущем для усовершенствования комплекса планируется вместо зарубежной микросхемы STM32F3Discovery использовать отечественную 5559ИН12У от предприятия ОАО «Научно-производственное объединение «Физика»».

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Анализ программных средств управления электроприводами [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/programmnye-sredstva-upravleniya-elektroprivodami/viewer (Дата обращения: 01.02.2023).

2 Управление современным электроприводом с персонального компьютера [Электронный ресурс]. – Режим доступа: https://controleng.ru/e-lektroprivod/upravlenie-sovremenny-m-e-lektroprivodom-s-personal-nogo-komp-yutera/ (Дата обращения: 01.02.2023).

3 Программа управления шаговым двигателем [Электронный ресурс]. – Режим доступа: https://stepmotor.ru/soft-2 (Дата обращения: 02.02.2023).

4 Комплект для разработки программного обеспечения STM32 [Электронный ресурс]. – Режим доступа: https://www.st.com/en/embedded-software/stsw-stm32100.html (Дата обращения: 10.02.2023).

5 Комплект средств для разработки программного обеспечения OsmModbusControl SDK [Электронный ресурс]. – Режим доступа: https://onitex.ru/old/downloads/sdk.html (Дата обращения: 13.02.2023).

6 STM32 MC Workbench Документация [Электронный ресурс]. – Режим доступа: https://wiki.st.com/stm32mcu/wiki/STM32MotorControl:STM32_MC_Workbench#General_information (Дата обращения: 15.02.2023).

7 Перевод справочной документации ST Motor Control Workbench [Электронный ресурс]. – Режим доступа: https://russianblogs.com/article/4835638775/(Дата обращения: 17.02.2023).

8 Электропривод. Программный пакет для моделирования и настройки системы управления электрическим приводом [Электронный ресурс]. – Режим доступа: https://kpm-ritm.ru/motor (Дата обращения: 20.02.2023).

9 Новиелло К. Освоение STM32 [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/16485874/ (Дата обращения: 19.02.2023).

10 Фаулер М. [Fowler M.] UML. Основы: пер. с англ. // Типы диаграмм. СПб: Символ-Плюс, 2004. – С. 80–123.

11 Буч, Грейди Язык UML. Руководство пользователя / Грейди Буч , Джеймс Рамбо , Айвар Джекобсон. - М.: ДМК, 2015. - 432 с.

12 Miles R. C# Programming Yellow Book [Электронный ресурс]. – Режим доступа: https://777russia.ru/book/uploads/ПРОГРАММИРОВАНИЕ/C%23/C%23%206.0%20Yellow%20Book%20Banana%20Edition%207.0%2C%202015.pdf (Дата обращения: 19.04.2023).

13 Кабдин Н.Е., Сторчевой В.Ф. ЭЛЕКТРОПРИВОД. Учебник для вузов // Основы теории электропривода. 2021. С. 27–96.

14 «НПО «ФИЗИКА»». Документация на микросхему 5559ИН12У [Электронный ресурс]. – Режим доступа: https://npofizika.ru/pdf/5559IN12U.pdf (Дата обращения: 03.05.2023).

15 Левенталь, Л. Введение в микропроцессоры: Программное обеспечение, аппаратные средства и программирование / Л. Левенталь. - М.: Энергоатомиздат, 2008. - 464 с.

16 Керниган, Брайан Практика программирования / Брайан Керниган , Роб Пайк. - М.: Вильямс, 2015. - 288 с.

ПРИЛОЖЕНИЯ

Приложение А

Исходный код модели Employee

```
using System.ComponentModel.DataAnnotations;
namespace DiplomProj.Models
{
    public class Employee
    {
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        [Required]
        public string Position { get; set; }
        [Required]
        public string Password { get; set; }
    }
}
```

Приложение Б

Исходный код модели EngineDetailsModel

```
using System;
using System.Collections.Generic;
namespace DiplomProj.Models
{
   public class EngineDetailsModel
   {
      public string EngineType { get; set; }
      public decimal CurrentValue { get; set; }
      public decimal PowerValue { get; set; }
      public decimal RotationSpeed { get; set; }
      public decimal EngineTemperature { get; set; }
      public decimal KeyTemperature { get; set; }
      public decimal Torque { get; set; }
   } }
```

Приложение В

Исходный код модели EngineTypeModel

```
using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace DiplomProj.Models
{
   public class EngineTypeModel
   {
      public string SelectedEngineType { get; set; }
      public List<SelectListItem> EngineTypes { get; set; }
      public EngineTypeModel()
      {
         EngineTypes = new List<SelectListItem>
      {
         new SelectListItem { Value = "Asynchronous", Text = "Асинхронный"
},
         new SelectListItem { Value = "Synchronous", Text = "Синхронный" }
      };
   }  }
}
```

Приложение Г

Исходный код модели LoginModel

```
using System.ComponentModel.DataAnnotations;
namespace DiplomProj.Models
{
    public class LoginModel
    {
        [Key]
        public int Id { get; set; }

        [Required(ErrorMessage = "Please enter your username.")]
        [StringLength(20, ErrorMessage = "The {0} must be at least {2} and at most {1} characters long.", MinimumLength = 3)]
        public string Username { get; set; }
        [Required(ErrorMessage = "Please enter your password.")]
        [DataType(DataType.Password)]
        [StringLength(20, ErrorMessage = "The {0} must be at least {2} and at most {1} characters long.", MinimumLength = 6)]
        public string Password { get; set; }
    }
}
```

Приложение Д

Исходный код модели SyncEngineModel

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace DiplomProj.Models
{
    public class SyncEngineModel
    {
        public double CurrentValue { get; set; }
        public double PowerValue { get; set; }
        public double RotationSpeed { get; set; }
        public double EngineTemperature { get; set; }
        public double KeyTemperature { get; set; }
        public double Torque { get; set; }
        public double RotationAngle { get; set; }
    }
}
```

Приложение Е

Исходный код представлений

Приложение Е.1

Исходный код представления Login.cshtml

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace DiplomProj.Models
{
   public class SyncEngineModel
   {
      public double CurrentValue { get; set; }
      public double PowerValue { get; set; }
      public double RotationSpeed { get; set; }
      public double EngineTemperature { get; set; }
      public double KeyTemperature { get; set; }
      public double Torque { get; set; }
      public double RotationAngle { get; set; }
   }
}
```

Приложение Е.2

Исходный код представления EngineType.cshtml

```
@{
   ViewData["Title"] = "Выбор двигателя";}
@model DiplomProj.Models.EngineTypeModel
<h2>Выбор типа двигателя</h2>
<form method="post" asp-controller="SyncEngine" asp-action="SyncEngineDetails"
id="engineForm" onsubmit="return validateEngineType()">
   <div class="form-group">
      <label asp-for="SelectedEngineType">Выберите тип двигателя:</label>
      <select asp-for="SelectedEngineType" asp-items="Model.EngineTypes"
class="form-control" id="engineTypeSelect">
         <option value="">-- Выберите --</option>  </select>
   </div>
   <div class="form-group">
      <button type="submit" class="btn btn-primary">Продолжить</button>
   </div>
</form>
<script>
   function validateEngineType() {
      var selectedValue = document.getElementById("engineTypeSelect").value;
      if (!selectedValue) {
         return false;  }
      else if (selectedValue === "Asynchronous") {
         window.location.href = "Engine\Details"; // Перенаправление на страницу
EngineDetails при выборе "Асинхронный"
         return false; // Отменяем отправку формы
      } }
</script>
```

Приложение Е.3

Исходный код представления EngineDetails.cshtml

```
@{
   ViewData["Title"] = "Асинхронный двигатель";
}
@model DiplomProj.Models.EngineDetailsModel
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>Параметры асинхронного двигателя</title>
   <link rel="stylesheet" href="styles.css"> <!-- Подключение CSS-файла -->
</head>
<body>
   <div class="container">
      <h2>Параметры асинхронного двигателя</h2>
      <table>
         <tr>
            <td>Сила тока:</td>
            <td>@Model.CurrentValue</td>
         </tr>
         <tr>
            <td>Потребляемая мощность:</td>
            <td>@Model.PowerValue</td>
         </tr>
         <tr>
            <td>Скорость вращения:</td>
            <td>@Model.RotationSpeed</td>
         </tr>
```

```
            <tr>
                <td>Температура двигателя:</td>
                <td>@Model.EngineTemperature</td>
            </tr>
            <tr>
                <td>Температура силовых ключей:</td>
                <td>@Model.KeyTemperature</td>
            </tr>
            <tr>
                <td>Момент двигателя:</td>
                <td>@Model.Torque</td>
            </tr>
        </table>
    </div>
</body>
</html>
```

Приложение Е.4

Исходный код представления SyncEngineModel.cshtml

```
@{
   ViewData["Title"] = "Синхронный двигатель";
}
@model DiplomProj.Models.SyncEngineModel
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>Параметры синхронного двигателя</title>
   <link rel="stylesheet" href="styles.css"> <!-- Подключение CSS-файла -->
</head>
<body>
   <div class="container">
      <h2>Параметры синхронного двигателя</h2>
      <table>
         <tr>
            <td>Сила тока:</td>
            <td>@Model.CurrentValue</td>
         </tr>
         <tr>
            <td>Потребляемая мощность:</td>
            <td>@Model.PowerValue</td>
         </tr>
         <tr>
            <td>Скорость вращения:</td>
            <td>@Model.RotationSpeed</td></tr>
         <tr>
```

```
            <td>Температура двигателя:</td>
            <td>@Model.EngineTemperature</td></tr>
        <tr>
            <td>Температура силовых ключей:</td>
            <td>@Model.KeyTemperature</td>
        </tr>
        <tr>
            <td>Момент двигателя:</td>
            <td>@Model.Torque</td> </tr>
        <tr>
            <td>Угол поворота:</td>
            <td>@Model.RotationAngle</td></tr>
      </table>
   </div>
</body>
</html>
```

Приложение Ж

Исходный код контроллера LoginController

```
using DiplomProj.Models;
using DiplomProj.Data;
using Microsoft.AspNetCore.Authorization;
namespace DiplomProj.Controllers
{
    public class LoginController : Controller
    {
        private readonly MyDbContext _context;
        public LoginController(MyDbContext context)
        {
            _context = context;
        }
        // GET: /Login/
        public IActionResult Index()
        {
            return View("Login");
        }
        public async Task<IActionResult> Index(LoginModel model)
        {
            if (ModelState.IsValid)
            {
                var user = await _context.Employees.FirstOrDefaultAsync(u => u.Name ==
model.Username && u.Password == model.Password);

                if (user != null)
                {
                    // Пользователь найден, сохраняем его идентификатор в куки
```

```
                Response.Cookies.Append("UserId", user.Id.ToString());

                return RedirectToAction("EngineType", "Engine");
            }
        }
        ViewData["ErrorMessage"] = "Неверное имя пользователя или пароль";
        return View("Login", model);
    }
    [Authorize]
    public IActionResult Logout()
    {
        Response.Cookies.Delete("UserId");
        return RedirectToAction("Index", "Home");
    }
  }
}
```

Приложение К

Исходный код контроллера EngineTypeController

```
using Microsoft.AspNetCore.Mvc;
using DiplomProj.Models;

namespace DiplomProj.Controllers
{
    public class EngineTypeController : Controller
    {
        // GET: /EngineType
        public IActionResult Index()
        {
            var model = new EngineTypeModel();
            return View(model);
        }
        // POST: /EngineType/EngineDetails
        [HttpPost]
        public IActionResult EngineDetails(EngineTypeModel model)
        {
            if (ModelState.IsValid)
            {
                if (model.SelectedEngineType == "Асинхронный")
                {
                    return RedirectToAction("EngineDetails", "Engine");
                }
            }
            return View("Index", model);
        }}
}
```

Приложение Л

Исходный код контроллера EngineController

```
using Microsoft.AspNetCore.Mvc;
using DiplomProj.Models;
using Microsoft.AspNetCore.Mvc.Rendering;
namespace DiplomProj.Controllers
{
    public class EngineController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
        public IActionResult EngineType()
        {
            var engineTypes = new List<SelectListItem>
            {
                new SelectListItem { Value = "Синхронный", Text = "Синхронный"
},
                new SelectListItem { Value = "Асинхронный", Text =
"Асинхронный" }
            };
            var model = new EngineTypeModel
            {
                EngineTypes = engineTypes
            };
            return View(model);
        }
        [HttpPost]
```

```
        public IActionResult SelectEngineType(EngineTypeModel model)
        {
            if (model.SelectedEngineType == "Асинхронный")
            {
                return RedirectToAction("Details", new { engineType =
model.SelectedEngineType });
            }
            return RedirectToAction("Index", "Home");
        }
        public IActionResult EngineDetails(string engineType)
        {
            var model = new EngineDetailsModel
            {
                EngineType = engineType,
                CurrentValue = 0,
                PowerValue = 0,
                RotationSpeed = 0,
                EngineTemperature = 0,
                KeyTemperature = 0,
                Torque = 0
            };
            return View("EngineDetails", model);
        } }
    }
```

Приложение М

Исходный код контроллера SyncEngineController

```
using DiplomProj.Models;
using RJCP.IO.Ports;

namespace DiplomProj.Controllers
{
    public class SyncEngineController : Controller
    {
        private readonly string comPortName = "COM5";
        private SerialPortStream serialPort;
        private Random random;
        public IActionResult SyncEngineDetails()
        {
            var model = new SyncEngineModel();

            return View(model);
        }
        [HttpPost]
        public async Task<IActionResult> ListenToCOMPort()
        {
            try
            {
                serialPort = new SerialPortStream(comPortName);
                serialPort.BaudRate = 9600;
                serialPort.Parity = Parity.None;
                serialPort.DataBits = 8;
                serialPort.StopBits = StopBits.One;
                serialPort.Open();
```

```
    random = new Random();
    while (true)
    {
        var buffer = new byte[1];
        var receivedData = string.Empty;

        while (true)
        {
            await serialPort.ReadAsync(buffer, 0, 1);
            var receivedChar = (char)buffer[0];

            if (receivedChar == '\n')
                break;

            receivedData += receivedChar;
        }

        var model = ParseReceivedData(receivedData);
        if (model != null)
        {
            UpdateModelWithRandomValues(model);
            return View("SyncEngineDetails", model);
        }
    }
}
catch (Exception ex)
{
    // Обработка ошибок и закрытие COM-порта
    if (serialPort != null && serialPort.IsOpen)
```

```
            serialPort.Close();
        return BadRequest(ex.Message);
    }
}
private SyncEngineModel ParseReceivedData(string receivedData)
{
    var values = receivedData.Split(',');

    if (values.Length == 7)
    {
        var model = new SyncEngineModel();

        if (double.TryParse(values[0], out double currentValue))
            model.CurrentValue = currentValue;

        if (double.TryParse(values[1], out double powerValue))
            model.PowerValue = powerValue;

        if (double.TryParse(values[2], out double rotationSpeed))
            model.RotationSpeed = rotationSpeed;

        if (double.TryParse(values[3], out double engineTemperature))
            model.EngineTemperature = engineTemperature;

        if (double.TryParse(values[4], out double keyTemperature))
            model.KeyTemperature = keyTemperature;

        if (double.TryParse(values[5], out double torque))
            model.Torque = torque;
```

```
            if (double.TryParse(values[6], out double rotationAngle))
                model.RotationAngle = rotationAngle;

            return model;
        }
        return null;
    }
    private void UpdateModelWithRandomValues(SyncEngineModel model)
    {

        model.CurrentValue = random.NextDouble();
        model.PowerValue = random.NextDouble();
        model.RotationSpeed = random.NextDouble();
        model.EngineTemperature = random.NextDouble();
        model.KeyTemperature = random.NextDouble();
        model.Torque = random.NextDouble();
        model.RotationAngle = random.NextDouble();
    }
    protected override void Dispose(bool disposing)
    {
        if (serialPort != null)
        {
            serialPort.Dispose();
            serialPort = null;
        }
        base.Dispose(disposing);
    }
  }
}
```