Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись
« ____» _____ 2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника
код – наименование направления

Система управления динамическими испытаниями автомобильных узлов. Программная часть
тема

Руководитель	_____ подпись, дата	доцент, канд.техн.наук должность, ученая степень	В.Г. Серёдкин инициалы, фамилия
Выпускник	_____ подпись, дата		Е.В. Соколов инициалы, фамилия
Нормоконтролер	_____ подпись, дата	доцент, канд.техн.наук должность, ученая степень	В.Г. Серёдкин инициалы, фамилия

Красноярск 2022

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись
« ___» _____ 2022 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Соколову Егору Вячеславовичу

фамилия, имя, отчество

Группа КИ18-09Б номер Направление (специальность) 09.03.01 код

Информатика и вычислительная техника

наименование

Тема выпускной квалификационной работы Система управления динамическими испытаниями автомобильных узлов. Программная часть.

Утверждена приказом по университету № _____от _____

Руководитель ВКР В. Г. Серёдкин, канд. техн. наук, доцент, доцент кафедры ВТ ИКИТ СФУ

инициалы, фамилия, должность, учёное звание и место работы

Исходные данные для ВКР: Оборудование, предоставленное ПИ СФУ для динамических испытаний, прототип модуля сбора данных.

Перечень разделов ВКР: Анализ предметной области. Математическое обоснование проводимых экспериментов. Программная часть. Проведение испытаний.

Перечень графического материала: Презентация в формате Microsoft Power Point, видеоматериал результатов работы прототипа системы.

Руководитель ВКР _____ В. Г. Серёдкин

подпись инициалы, фамилия

Задание принял к исполнению _____ Е.В. Соколов

подпись инициалы, фамилия

«___» _____ 20__г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Система управления динамическими испытаниями автомобильных узлов. Программная часть» содержит 58 страниц текстового документа, 4 таблицы, 31 рисунок, 10 использованных источников и 13 страниц приложения А.

PROCESSING, MODBUS, ГРАФИЧЕСКИЙ ИНТЕРФЕЙС, КАЛИБРОВКА, ИНВЕРТОР, ЧАСТОТА, ГРАФИК, БАЗА ДАННЫХ, КОМПИЛЯЦИЯ, МИКРОКОНТРОЛЛЕР, АМОРТИЗАТОР

Объект выпускной квалификационной работы: процесс создания программного обеспечения для микроконтроллерной системы и проведение испытаний с его использованием.

Предмет выпускной квалификационной работы: программное обеспечение системы приема данных микроконтроллерной системы, преобразование форматов и управления динамическими испытаниями автомобильных узлов.

Целью данной выпускной квалификационной работы является разработка программного обеспечения системы управления динамическими испытаниями автомобильных узлов.

Выпускная квалификационная работа выполнена в соответствии с индивидуальным заданием. В процессе выполнения работы решены задачи:

1. Выполнен анализ предметной области проекта «Система управления динамическими испытаниями автомобильных узлов».
2. Приведено математическое обоснование проводимых экспериментов.
3. Разработано программное обеспечение для системы динамических испытаний.
4. Постановлены эксперименты и проведены динамические испытания электромагнитного амортизатора.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

За последние 20 лет конструкция гидравлических амортизаторов, применяемая в подвесках транспортных средств практически, не изменились и, по сути, себя исчерпала. Рабочие характеристики амортизаторов наиболее зависят от параметров рабочего тела – масла, а так как работа амортизатора основана на жидкостном трении, то вследствие гашения колебаний происходит нагрев масла с последующим изменением характеристик масла и самого амортизатора. Преобразованная энергия колебаний в процессе работы рассеивается в окружающую среду в среднем величина которой составляет 500-1000 Вт. К тому же гидравлические амортизаторы склонны к отказу уплотнения штока и утечки рабочей жидкости.

Альтернативой выступают электромагнитные амортизаторы, способные преобразовывать основную долю высвобождаемой энергии в полезную электрическую, что наиболее актуально для электрических и гибридных силовых установок транспортных средств.

В настоящее время основным разработчиком электромагнитной подвески является Амар Боуз (основатель Bose Corporation), создавший электромагнитные амортизационные стойки, в которых роль как упругого элемента, так и демпфера выполняют статоры, а роторы находятся в жёсткой связи с каждым из колес. Весь комплекс получает сигналы от центрального контроллера, который анализирует данные, получаемые от различных датчиков, и управляет системой по заложенным алгоритмам.

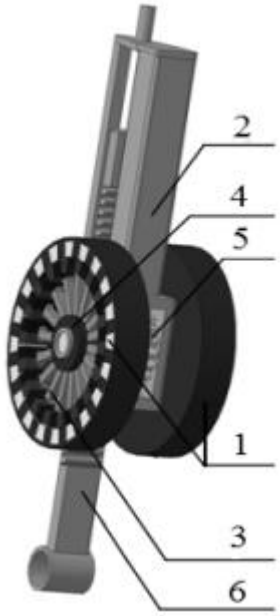
Ещё одним разработчиком электромагнитной подвески является Технологический университет Эйндховена совместно с фирмой SKF. Разработчиками предлагается схема подвески с использованием электромагнитной «капсулы» только в качестве демпфера, а функция упругого элемента остаётся за обычной пружиной. Система состоит из электромагнитного актуатора, управляющего блока и источника питания. Общее энергопотребление системы около 500 Вт. В случае отказа электромагнитного амортизатора, подвеска продолжает работать как обычная пружинная, только без демпфирования.

К преимуществам электромагнитной подвески относятся возможность адаптации к дефектам дорожного покрытия, путём изменения жёсткости, и управление скоростью реакции подвески. Главным недостатком этих систем является существенное энергопотребление, необходимое для поддержки работы системы. Отказ работы системы из-за отсутствия демпфирования приведёт к дискомфортному и небезопасному передвижению транспортного средства, в крайнем случае движение автомобиля станет невозможным.

1 Анализ предметной области

1.1 Анализ задания и обоснование выбранного решения

В Политехническом институте Сибирского федерального университета ведётся разработка электромагнитного амортизатора, который способен преобразовывать энергию колебаний в электрическую для передачи её в бортовую сеть автомобиля, адаптировать подвеску к условиям движения ТС и обеспечивать её работоспособность без отбора энергии из бортовой сети.



1 – генераторы; 2 – корпус; 3 – роторы генератора; 4 – обгонная муфта; 5 – мультипликатор; 6 – зубчатая рейка

Рисунок 1 – Принципиальная схема электромагнитного амортизатора вращательного типа

На данном этапе разработки предполагается следующий вариант конструктивного исполнения электромагнитного амортизатора с применением генератора вращательного типа (рисунок 1), представляющая собой два независимо друг от друга работающих генератора (для ходов сжатия и отбоя), закреплённых на корпусе. Привод роторов осуществлён через обгонные муфты, установленные на мультипликатор, связанный с зубчатой рейкой. Применение в данной схеме обгонных муфт и двух не связанных между собой генераторов обусловлено необходимостью снижения влияния динамических нагрузок при смене направления движения зубчатой рейки.

Данная схема возможна в довольно компактном исполнении за счёт увеличения частоты вращения роторов генераторов, что положительно отражается на КПД всего узла в целом.

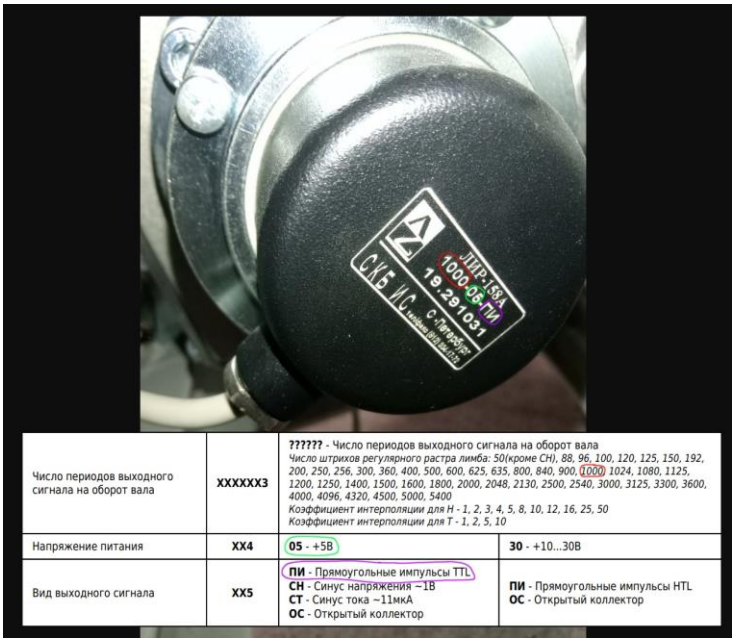
С целью изучения эффективности данного амортизатора в Политехнический институт Сибирского федерального университета была доставлена стендовая установка SS20 S400AM, произведенная компанией Centurion (рисунок 4), которая включает в себя: исследуемый генератор, привод, источник питания для обмотки возбуждения генератора, систему частотного преобразования и динамометр для определения момента сопротивления генератора (все элементы приведены в рисунке 5), а также тензорезистивный датчик (рисунок 2) и угловой энкодер (рисунок 3).

Параметры датчика	Единицы измерения	Значения параметров	
Наибольший предел измерения (НПИ)	т	0,5; 1; 2; 3; 5; 7; 10; 20	
Класс точности по ГОСТ 8.631-2013		С1	С3
Число поверочных интервалов		1000	3000
Минимальный поверочный интервал		НПИ / 5000	НПИ / 10000
Рабочий коэффициент передачи (РКП)	мВ/В	2 ±0,005; 1±0,0025 (10 т); 1,5 ±0,0040 (20 т)	2 ±0,002; 1±0,0010 (10 т); 1,5 ±0,0015 (20 т)
Начальный коэффициент передачи (НКП)	% от РКП	< 3	< 3
Комбинированная погрешность	% от РКП	≤ ±0,040	≤ ±0,020
Ползучесть (30 мин.)	% от РКП	≤ ±0,049	≤ ±0,025
Изменение НКП от температуры	% от РКП/°С	≤ ±0,0028	≤ ±0,0014
Изменение РКП от температуры	% от РКП/°С	≤ ±0,0022	≤ ±0,0011
Наибольшее напряжение питания постоянного тока	В	12	
Сопротивление входное	Ом	380 ±15	
Сопротивление выходное	Ом	350 ±1	
Сопротивление изоляции	ГОм	≥ 5	
Номинальный диапазон температур	°С	-10... +40	
Диапазон температур эксплуатации и хранения	°С	-50... +50	
Степень защиты по ГОСТ 14254		IP65	
Допустимая перегрузка в течение не более 1 часа	% от НПИ	25	
Разрушающая нагрузка	% от НПИ	300	
Материал датчика		Легированная сталь	

Массо-габаритные параметры

НПИ, т	L1, мм	L2, мм	B1, мм	B2, мм	D, мм	M, мм	H, мм	Масса датчика, кг	Длина кабеля, м
0,5; 1; 2	94	9	32	42	96	M16	90	1,5	3
3, 5, 7	120	12	50	60	126	M24	120	4	10
10	140	15	74	84	154	M30x2	140	9,3	
20								8	

Рисунок 2 – Тензорезистивный датчик Тензо-М С2



Число периодов выходного сигнала на оборот вала	XXXXXX3	?????? - Число периодов выходного сигнала на оборот вала *Число штрихов регулярного растра лимба: 50(кроме СН), 88, 96, 100, 120, 125, 150, 192, 200, 250, 256, 300, 360, 400, 500, 600, 625, 635, 800, 840, 900, 1000, 1024, 1080, 1125, 1200, 1250, 1400, 1500, 1600, 1800, 2000, 2048, 2130, 2500, 2540, 3000, 3125, 3300, 3600, 4000, 4096, 4320, 4500, 5000, 5400* *Коэффициент интерполяции для Н - 1, 2, 3, 4, 5, 8, 10, 12, 16, 25, 50* *Коэффициент интерполяции для Т - 1, 2, 5, 10*	
Напряжение питания	XX4	**05** - +5В	**30** - +10...30В
Вид выходного сигнала	XX5	**ПИ** - Прямоугольные импульсы TTL **СН** - Синус напряжения ~1В **СТ** - Синус тока ~11мкА **ОС** - Открытый коллектор	**ПИ** - Прямоугольные импульсы HTL **ОС** - Открытый коллектор

Рисунок 3 – Угловой энкодер ЛИР-158А

Однако заводское ПО, идущее в комплекте со стендом, не соответствовало нужным требованиям: отсутствовала возможность анализа «сырых» показателей, отсутствовала возможность измерения рекуперативных свойств и температуры амортизаторов. Кроме того, стенд не позволял более гибко настраивать параметры диагностики, таких, как скорость вращения мотора с маховиком и длительность испытания.

Первоначально поставленная задача проекта предполагала рефакторинг данного заводского ПО с целью добавления требуемого заказчиком функционала. Однако впоследствии было принято решение о разработке с нуля новой микроконтроллерной системы управления динамическими испытаниями автомобильных амортизаторов и нового ПО по следующим причинам:

- Исходное заводское ПО не обладало открытым исходным кодом и никак не позволяло наиболее гибко взаимодействовать с собой;
- Любые изменения исходных параметров ПО могло повлечь за собой несовместимость с уже установленными компонентами аппаратной части системы;
- Заводская аппаратная часть системы, а именно микроконтроллерный блок, также обладал недостатками: один из них заключается в недостаточной точности отображения данных при диагностике автомобильных амортизаторов.

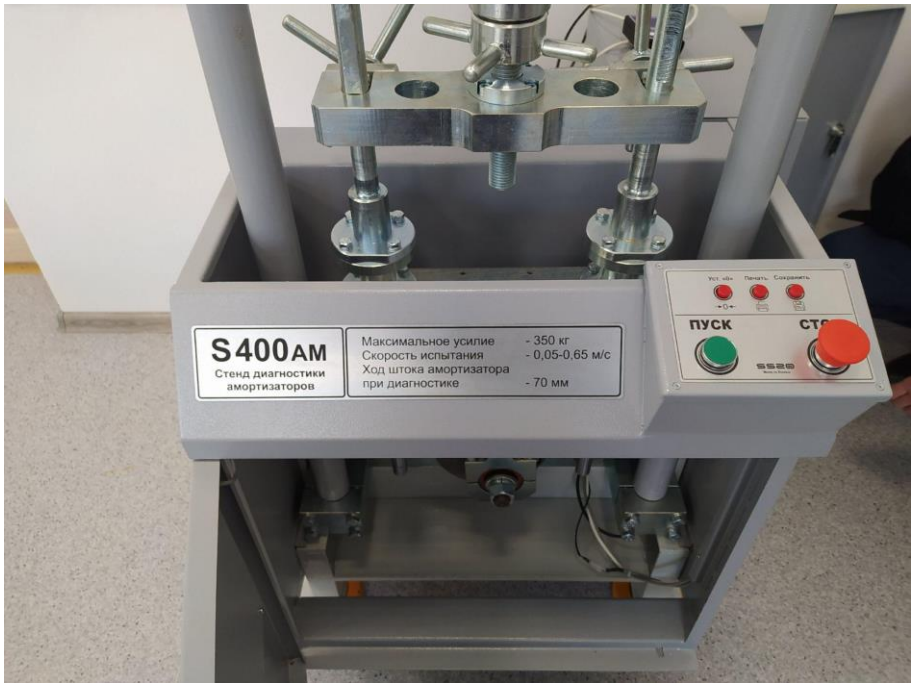


Рисунок 4 – Стенд диагностики амортизаторов SS20 S400 AM

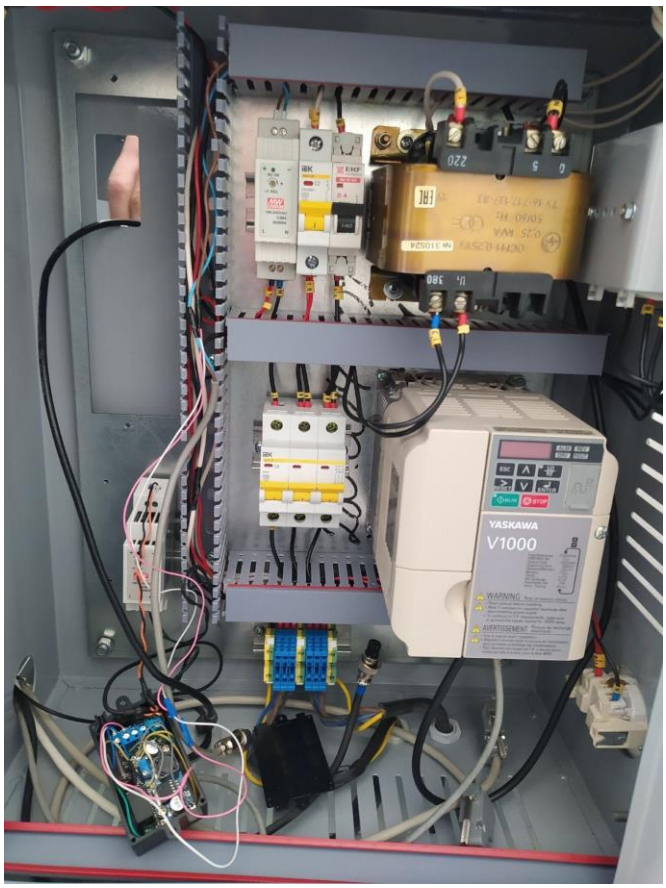


Рисунок 5 – Система питания стенда для испытания автомобильных амортизаторов

Помимо всего, для облегчения разработки проекта было принято решение разбить проект на три взаимосвязанных и взаимозависимых подпроекта: моделирование и испытания, аппаратная часть и программная часть.

1.2 Постановка требовании при выполнении задачи

Целью текущей бакалаврской работы является разработка системы управления динамическими испытаниями автомобильных амортизаторов на микроконтроллерной основе. Разработанная система должна обеспечивать мониторинг испытания автомобильных амортизаторов и управление им.

Система должна удовлетворять следующим требованиям:

- Состав системы должен включать в себя микроконтроллер, обрабатывающий данные с датчиков, персональный компьютер
- При передаче данных на персональный компьютер с микроконтроллера должна сохраняться их целостность.
- Должно обеспечиваться устойчивое соединение датчиков и исполнительных устройств.
- Система должна иметь более низкие показатели энергозатрат по сравнению с аналогами системы.

- Стоимость системы должна быть ниже, чем стоимость систем аналогов.
- Точность данных должна варьироваться в пределах допустимых значений погрешности.
- Вывод данных и управление исполнительными устройствами должен осуществляться на простой в использовании пользовательский интерфейс.
- Система должна выполнять следующие задачи:
- Обеспечить сбор и обработку в реальном времени для представления их в удобном для пользователя виде.
- Обеспечить управление работой испытательной установки.

1.3 Цели создания системы

Целями создания данной системы являются:
- Автоматизация процесса диагностики автомобильного амортизатора;
- Исправление существующих недоработок стендовой установки и её системы диагностики;
- Расширение функционала стендовой установки на основе заводских параметров.

1.4 Функциональные возможности

- Отслеживание рекуперируемой энергии;
- Отслеживание температурных свойств;
- Наличие нескольких тестовых режимов работы для проверки агрегата;
- Сопоставления данных с математической моделью и представление их в различном виде (числа, графики);
- Компоновка собираемых данных для вывода на печать.

1.5 Анализ существующих решений

1.5.1 Стенд для диагностики амортизаторов Авто Оснастка «ПС-63»

Описание**:** стенд (пневматический) для диагностики стоек и амортизаторов «ПС-63» (рисунок 6) предназначен для проверки всех видов амортизаторов, определяет необходимость ремонта и работоспособность после проведения восстановительных работ. Быстро и эффективно производит диагностику амортизатора на наличие неисправности, недостаточного количества масла и газового подпора.

Стоимость 39000р.

Таблица 1 – Основные характеристики стенда диагностики амортизаторов ПС-63

Наименование параметра	**Значение**
Максимальное контролируемое усилие, кг/см^2	2,5
Плавная регулировка скоростей	есть
Ход штока амортизатора при диагностике, мм	80
Точность измерения усилий, не более, кг	0,5
Максимальная ширина корпуса испытываемого амортизатора, мм	260
Максимальная длина корпуса испытываемого амортизатора, мм	700
Количество испытываемых амортизаторов, шт	1
Габаритные размеры В*Ш*Г, мм	1260*360*430
Масса, кг	31

Рисунок 6 – Стенд Авто Оснастка «ПС-63»

1.5.2 Стенд для диагностики амортизаторов MSG MS1000+

Особенности стенда MS1000+ (рисунок 7):

- диагностика амортизаторов различных типов конструкций;
- проверка агрегата под нагрузкой на различных скоростях с имитацией работы на движущемся автомобиле;
- сравнение результатов тестирования;

- построение динамограммы поверх уже имеющейся;
- сохранение или распечатка результатов диагностики;
- простое программное обеспечение.

1 – рабочая зона; 2 – механический отсек; 3 – блок заправки амортизаторов азотом;4 – дисплей; 5 – пульт управления; 6 – электрический отсек; 7 – педальный узел управления пневматическими зажимами

Рисунок 7 – Стенд MS1000+ (вид спереди)

Таблица 2 – Основные характеристики стенда диагностики амортизаторов MS1000+

Наименование параметра	**Значение**
Напряжение, В	380±10%
Мощность, кВт	3.7
Тип питающей сети	Трехфазная (3L+N+PE)
Габаритные размеры В*Ш*Г, мм	2460*1042*482
Вес, кг	350
Управление зажимами стенда	Пневматическое
Рабочее давление пневматической системы стенда, бар	6
Ход штока амортизаторов	Регулируемый
Установка хода штока амортизатора	Ручная
Диапазон установки хода штока амортизатора, мм	0-140
Регулировка высоты установки амортизатора	Ручная

Наименование параметра	**Значение**
Максимальная нагрузка отбой/сжатие, кг	1000
Режимы работы	Ручной или автоматический
Количество режимов проверки	1-6
Типы режимов проверки (по умолчанию)	Скорость 1–30 оборотов Скорость 2–60 оборотов Скорость 3–90 оборотов Скорость 4–120 оборотов Скорость 5–150 оборотов Скорость 6–180 оборотов
Выводимые параметры	– Отбой/сжатие – Температура амортизатора – Ход амортизатора
Вывод данных на печать	Есть
Подключение к интернету	Wi-Fi (802.11 a/b/g/ac)

Диагностический стенд MSG MS1000+ предназначен для тестирования амортизаторов посредством измерения зависимости прикладываемого усилия к исследуемому агрегату от расположения штока по отношению к корпусу амортизатора.

Применение стенда MS1000+ гарантирует полную диагностику амортизаторов, согласно которой возможно установить их работоспособность, а также сформировать четкий план восстановительных работ.

Процесс диагностики амортизатора может производится автоматически или в ручном режиме с различной скоростью: 60, 120, 180 оборотов в минуту. Безопасность крепежа амортизатора на стенде обеспечивается с помощью пневматических зажимов.

Диагностика амортизатора осуществляется путем снятия динамограммы (зависимость прикладываемой силы от положения штока амортизатора). Максимально возможная нагрузка – 1000 кг.

В стенде предусмотрено подключение к Интернету через Wi-Fi. Обновление ПО и техническая поддержка возможны при наличии Интернет-соединения.

Изготовитель может без предварительного уведомления пользователей изменить конструкцию и программное обеспечение.

Гарантийные обязательства на стенд диагностики отменяются, если использовалось компьютерное оборудование или программы, не предназначенные для работы с данным стендом.

Для проведения диагностики необходимо использовать персональный компьютер под управлением операционной системы Windows вместе с закрытым программным обеспечением производителя.

Высокая стоимость 20000$

1.5.3 Стенд для диагностики амортизаторов CENTURION S1000AMR

Рисунок 8 – Стенд CENTURION S1000AMR

Описание стенда (рисунок 8) и его работы:

На стенде диагностики во время проведения испытания амортизатора на ходе сжатия и отбоя производится замер усилия сопротивления, сопровождающийся возможностью построения скоростной характеристики амортизатора и построения рабочей диаграммы.

Программное обеспечение стенда диагностики позволяет адаптировать испытания амортизатора под определенные условия, осуществлять выбор скорости прокачки, количество циклов работы, имеется возможность изменять параметры и типы амортизаторов, предназначенных для проведения испытаний. Доступ к базе данных программного модуля осуществляется как локально, так и по сети.

Стенд контроля и диагностики амортизаторов позволяет определять усилия ходов сжатия и отбоя амортизаторов, строить диаграмму Монро и скоростную характеристику амортизатора, определять давление газового подпора, выдавать результаты замеров в числовом, графическом виде и в виде заключения о соответствии либо несоответствии параметров. Графическое представление результатов измерений можно масштабировать, чтобы проанализировать работу разных узлов амортизатора.

Таблица 3 – Основные характеристики стенда диагностики амортизаторов S1000AMR

Параметр	**Значение**
Максимальное контролируемое усилие, кг	1000
Точность измерения усилий, не более, кг	0,1
Скорость испытания, м/с	0,05–0,65
Максимальная длина корпуса испытываемого амортизатора, мм	500
Максимальный размер корпуса испытываемого амортизатора по ширине, мм	300
Напряжение питания, В	380
Потребляемая мощность, не менее, кВт	3,0
Габаритные размеры, В х Ш х Г, мм	2025 х 815 х 1085

Стоимость 600000р.

1.6 Вывод

В данной главе был проведен анализ предложенного задания и поставленных требований, были поставлены цели и задачи, которые должны быть выполнены при реализации данной выпускной квалификационной работы. Обзор аналогов позволил получить представление какой должна быть система и какой функционал она должна реализовывать.

Наша система будет включать только необходимый функционал для испытаний, стоимость системы будет значительно ниже аналогов, так как будут использоваться распространенные микроконтроллерные решения и датчики. Использование открытых программных библиотек для работы датчиков позволит скорректировать программный код под конкретные условия испытаний.

2 Математическое обоснование проводимых экспериментов

Частота вращения двигателя измеряется в импульсах в секунду. Сначала определяется позиция в первый момент времени, позже, во второй момент времени (через секунду) определяется следующая позиция. Разница между значениями является частотой вращения двигателя.

Обороты в секунду и минуту вычисляются по формулам (1,2), где 4000 количество импульсов на один оборот энкодера.

$$RPS = \frac{Frequency}{4000} \quad (1)$$

$$RPM = 60 * PRS \quad (2)$$

Угловая скорость вычисляется по формуле (3).

$$\omega = 2\pi * RPS \quad (3)$$

Угол перемещения энкодера определяется следующим образом (формула 4), где EncCount – текущее положение энкодера.

$$EncAngle = \frac{360}{4000} * EncCount \quad (4)$$

Температура объекта измеряется цифровым датчиком, принятое значение необходимо делить на 16.

$$Temperature = \frac{value}{16} \quad (5)$$

Скорость амортизатора определяется по формуле 6, где r – радиус кривошипа, а $\lambda = r/L$ – отношение радиуса кривошипа к длине шатуна. [1].

$$v_a = r\omega\left(\sin\varphi + \frac{\lambda}{2}\sin 2\varphi\right) \quad (6)$$

Вычисление напряжения, где 3,3 напряжение работы микроконтроллера, value – значение, считываемое с микроконтроллера, 4095 – максимальное значение регистра, 30 – максимальное напряжение, выдаваемое объектом испытаний.

$$U = 3{,}3 * \frac{value}{4095} * 30 \quad (7)$$

Вычисление силы тока, где 3,3 напряжение работы микроконтроллера value – значение, считываемое с микроконтроллера, 4095 – максимальное

значение регистра, 20 – максимальная сила тока, выдаваемая объектом испытаний.

$$I = 3{,}3 * \frac{value}{4095} * 20 \quad (8)$$

Мощность вычисляется по формуле 9.

$$P = U * I \quad (9)$$

Перемещение штока амортизатора в зависимости от угла поворота кривошипа φ определяется по формуле 10, где r – радиус кривошипа, а $\lambda = r/L$ – отношение радиуса кривошипа к длине шатуна. [1].

$$s_{\text{ш}} = r\left((1 - cos\varphi) + \frac{\lambda}{4}(1 - \cos 2\varphi)\right) \quad (10)$$

В данной главе были рассмотрены формулы, учитывающие кинематику и динамику кривошипно-шатунной конструкции стенда для испытания автомобильного амортизатора, а также формулы, применимые только к электромагнитному амортизатору. Материал данной главы используется в части программного обеспечения, выполняющей вычисления.

3 Программная часть

3.1 Средства разработки

3.1.1 Processing 3 IDE

Среда разработки включает текстовый редактор, компилятор и окно отображения. Это позволяет создавать программное обеспечение в рамках тщательно разработанного набора ограничений.

Среда разработки Processing (рисунок 9) упрощает написание программ. Программы пишутся в текстовом редакторе и запускаются нажатием кнопки «Выполнить». В Processing компьютерная программа называется скетчем. Эскизы хранятся в Sketchbook, который представляет собой папку на вашем компьютере.

Возможности Processing расширяются за счет библиотек и инструментов. Библиотеки позволяют скетчам делать что-то помимо основного кода Processing. Сообщество Processing предоставляет сотни библиотек, которые можно добавить к вашим скетчам, чтобы включить новые функции, такие как воспроизведение звуков, компьютерное зрение и работа с расширенной трехмерной геометрией.

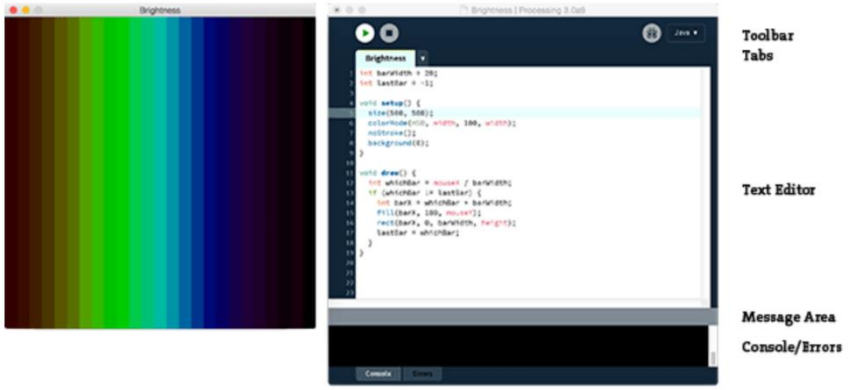


Рисунок 9 – Интерфейс Processing IDE

Для Windows и Linux в IDE существует возможность экспортировать код в исполняемое приложение, чтобы запустить программу на компьютере без Processing IDE [2].

3.1.2 Python - PyQt

PyQt – набор расширений (биндингов) графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python.

PyQt разработан британской компанией Riverbank Computing. PyQt работает на всех платформах, поддерживаемых Qt: Linux и другие UNIX-подобные ОС, Mac OS X и Windows. Существует 3 версии: PyQt6, PyQt5 и PyQt4,

поддерживающие соответствующие версии Qt. PyQt распространяется под лицензиями GPL (2 и 3 версии) и коммерческой.

PyQt практически полностью реализует возможности Qt. Это более 600 классов, более 6000 функций и методов, включая:

- Существующий набор виджетов графического интерфейса;
- стили виджетов;
- доступ к базам данных с помощью SQL (ODBC, MySQL, PostgreSQL, Oracle);
- QScintilla, основанный на Scintilla виджет текстового редактора;
- поддержку интернационализации (i18n);
- парсер XML;
- поддержку SVG;
- интеграцию с WebKit, движком рендеринга HTML;
- поддержку воспроизведения видео и аудио.

В состав PyQt также входит Qt Creator (Qt Designer) – редактор графического интерфейса пользователя. Программа pyuic генерирует код на языке Python из файлов, созданных в Qt Designer. Это делает PyQt очень полезным инструментом для быстрого прототипирования. Кроме того, можно добавлять новые графические элементы управления, написанные на Python, в Qt Designer.

3.2 Сравнение и обоснование выбранного средства разработки

Основное отличие двух сред разработки заключается в используемых языках. Processing, по сравнению с Python является компилируемым языком программирования. Главным преимуществом компилируемых языков программирования является скорость исполнения. Так как конвертация в машинный код обеспечивает гораздо быструю и эффективную работу, чем у интерпретируемых языков, у которых исходный код не преобразуется в машинный для исполнения центральным процессором, а исполняется с помощью программы-интерпретатора. Использование Processing будет гораздо привлекательнее для программы с постоянным исполнением главного цикла.

Вторым преимуществом Processing для дальнейшей поддержки программного кода является статическая типизация, когда в Python, напротив, используется динамическая типизация. В динамически типизированных языках программирования не требуется указывать конкретный тип данных, но языки не определяют его сами. Тип переменной становится известен после того момента, когда у него есть конкретные значения при запуске. Например, функция в Python

```
def f(x, y):
    return x + y
```

может склеивать строки, списки, складывать два целых или вещественных числа, и так далее, и мы не можем понять, что именно произойдет, пока не запустим программу [3].

Для реализации поставленной задачи был использован язык программирования Processing – он открытый исходный код, компилируемый и основан на Java. Processing IDE предоставляет лёгкий в использовании и быстрый в освоении инструментарий для разработчиков, которые хотят программировать анимацию, изображения, интерфейсы и работать с микроконтроллерами и периферийными устройствами, подключенными по последовательному порту. Язык Processing очень прост и отлично подходит для написания программ считывания и отправки информации на микроконтроллер. Язык является кроссплатформенным, требует установленной Java и Processing IDE.

3.3 Раздел разработки ПО

3.3.1 Приём данных с микроконтроллера

3.3.1.1 Описание работы программы приема данных

Алгоритм работы программы состоит из следующих шагов:

- обновление списка доступных последовательных портов и подключение к ним, на первый отправляются данные с микроконтроллера, второй служит для подачи управляющих воздействий на инвертор
- считывание и отображение информации с микроконтроллера. Программа получает пакет данных, состоящих из пяти величин, данные разделены между собой запятой, а в конце строки имеется знак переноса строки. Каждое значение до запятой помещается в массив строк и преобразуется в числовой тип данных для последующей обработки.
- вычисление скорости вращения инкрементального энкодера. Производится с помощью замера изменения счётчика в двух разных моментах времени.
- расчёт оборотов, скорости вращения в секунду и минуту, угловой скорости и угла поворота энкодера.
- отображение данных с микроконтроллера на графиках и гистограммах.

3.3.1.2 Блок-схема работы программы

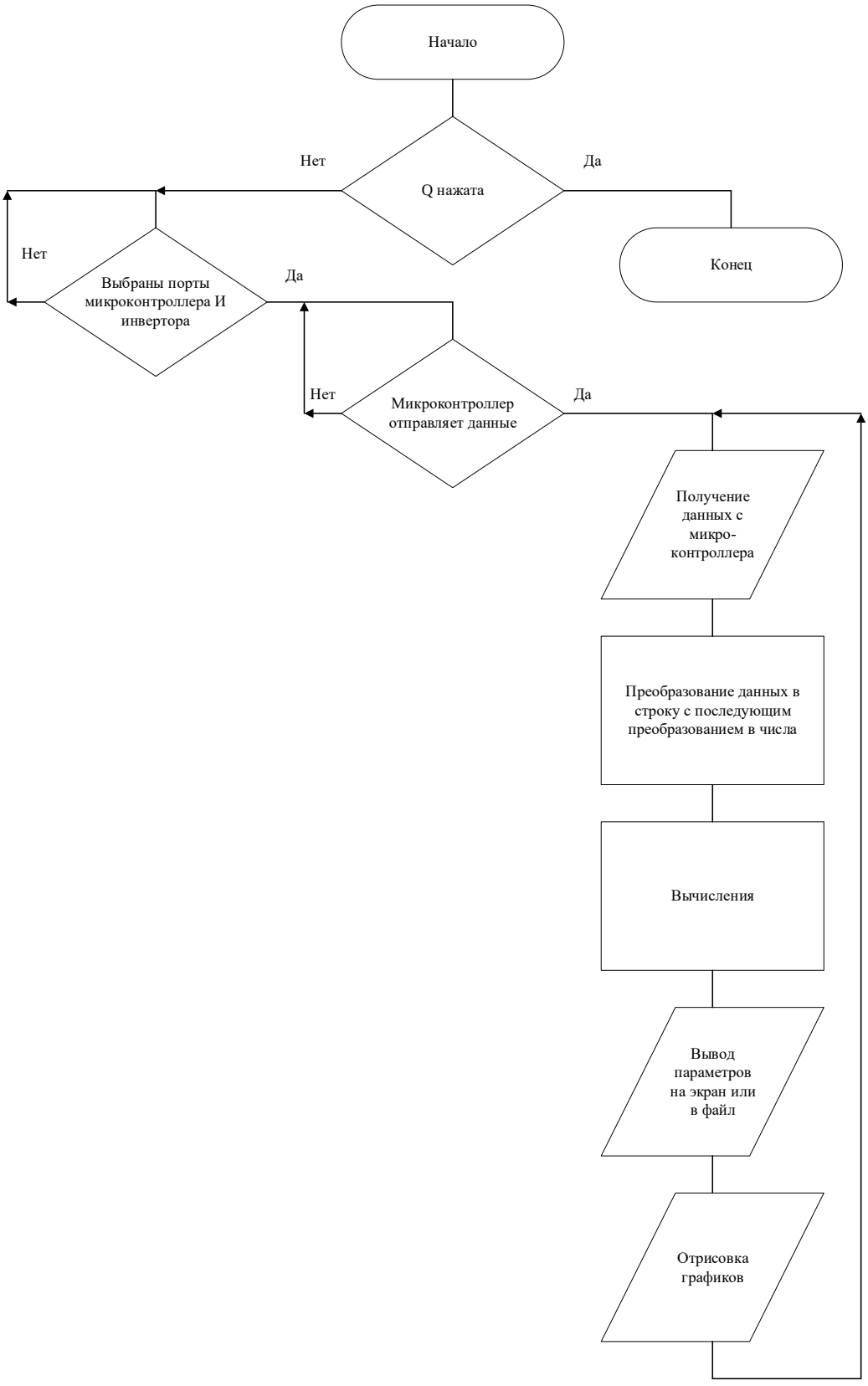


Рисунок 10 – Укрупненная блок-схема работы программы

На рисунке 10 изображена укрупненная блок-схема алгоритма работы программы. В ней отражены основные вычисления и операции ввода/вывода.

3.3.1.3 Проверка работоспособности программы

Работоспособность приёма данных в программе производилось с подключением реального микроконтроллера Arduino UNO к компьютеру. На микроконтроллер была загружена прошивка, изображенная на рисунке 11. В прошивке имеется пять значений, приближенных к реальным данным, которые будут считываться в процессе испытаний автомобильных амортизаторов. Переменные отправляются в последовательный порт через запятую с переносом строки в конце.

```
//примерные реальные получаемые значения
int EncCount = 0;
int EncSpd = 100/20; //импульсов в секунду/20 - отправка 5 раз в секунду
float Tenz = 30;
float Current = 15;
float Voltage = 10;
float Temperature = 35;

float Tenzl;
float Currentl;
float Voltagel;
float Temperaturel;

float FloatRand;

void setup(){
  Serial.begin(9600);
}

void loop() {
FloatRand = random(0,100)/100.0;
EncCount = EncCount + EncSpd;
Tenzl = Tenz + random(1, 100) + FloatRand;
Currentl = Current + random(1, 5) + FloatRand;
Voltagel = Voltage + random(1, 10) + FloatRand;
Temperaturel = Temperature + random(1, 10) + FloatRand;

Serial.print(EncCount);
Serial.print(' ');
Serial.print(Tenzl);
Serial.print(' ');
Serial.print(Currentl);
Serial.print(' ');
Serial.print(Voltagel);
Serial.print(' ');
Serial.println(Temperaturel);
delay(200);
}
```

Рисунок 11 – Прошивка для микроконтроллера

В программе происходит считывание строки данных до символа конца строки (рисунок 12), при этом полученная строка разделяется на массив строк,

которые были разделены между собой пробелом. Далее элемент массива преобразуется в числовой тип данных для дальнейших операций (вывод, вычисления).

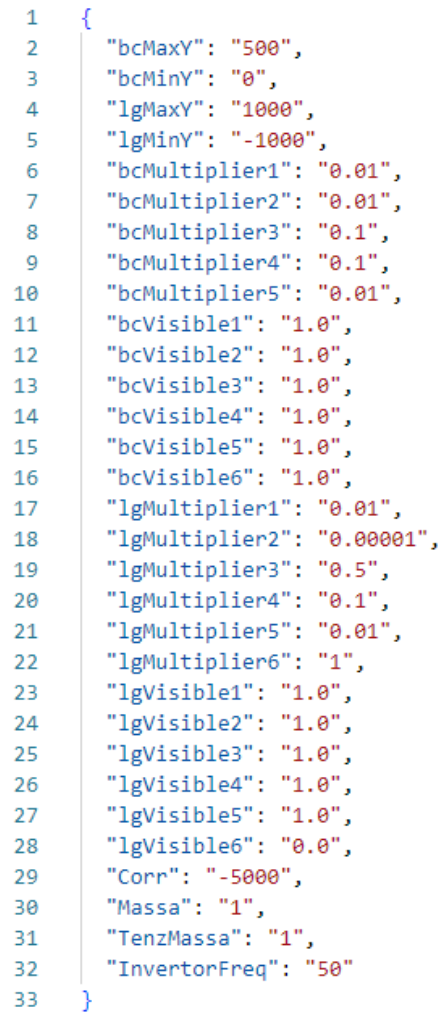
```
String myString = "";
//считывание строки, перенос - конец строки
myString = serial_stm32.readStringUntil('\n');
myString = myString.trim();
//массив строк для разделения данных (по пробелу)
String[] nums = split(myString, " ");
//преоразование массива строк в численное значение
 EncCount = int(nums[0]);
 Tenz = float(nums[1]);
 //3.3/4095
 Current = float(nums[2])*13513.5;
 Voltage = float(nums[3])*13513.5;
 Temperature = float(nums[4])/16;
```

Рисунок 12 – Фрагмент кода программы считывания данных

3.3.1.4 База данных для работы с графическим интерфейсом

Для сохранения ранее введенной информации в текстовых полях пользователем, при запуске программой считываются значения из базы данных, а также они обновляются при изменении. Параметры необходимы для настройки отображения графиков, а также для удобства использования калибровки тензодатчика. Например, здесь находятся максимальные и минимальные значения осей Y, параметр видимости отображаемой линии графика, где 0 – скрыт, 1 – отображается, коэффициенты отображаемых параметров.

Структура файла в формате JSON приведена на рисунке 13.



```
{
  "bcMaxY": "500",
  "bcMinY": "0",
  "lgMaxY": "1000",
  "lgMinY": "-1000",
  "bcMultiplier1": "0.01",
  "bcMultiplier2": "0.01",
  "bcMultiplier3": "0.1",
  "bcMultiplier4": "0.1",
  "bcMultiplier5": "0.01",
  "bcVisible1": "1.0",
  "bcVisible2": "1.0",
  "bcVisible3": "1.0",
  "bcVisible4": "1.0",
  "bcVisible5": "1.0",
  "bcVisible6": "1.0",
  "lgMultiplier1": "0.01",
  "lgMultiplier2": "0.00001",
  "lgMultiplier3": "0.5",
  "lgMultiplier4": "0.1",
  "lgMultiplier5": "0.01",
  "lgMultiplier6": "1",
  "lgVisible1": "1.0",
  "lgVisible2": "1.0",
  "lgVisible3": "1.0",
  "lgVisible4": "1.0",
  "lgVisible5": "1.0",
  "lgVisible6": "0.0",
  "Corr": "-5000",
  "Massa": "1",
  "TenzMassa": "1",
  "InvertorFreq": "50"
}
```

Рисунок 13 – Данные в формате JSON

Программный код записи названия переменных и их значений в БД изображен на рисунке 14. Встроенная функция controlEvent() в библиотеку ControlP5 позволяет отслеживать определенные события. Если произошло изменение в текстовом поле, переключателе, кнопке или группе объектов интерфейса, то в JSON файл записывается название объекта и его значение.

```
void controlEvent(ControlEvent theEvent) {
  if (theEvent.isAssignableFrom(Textfield.class) || theEvent.isAssignableFrom(Toggle.class) || theEvent.isAssignableFrom(Button.class) || theEvent.isGroup()) {
    String parameter = theEvent.getName();
    String value = "";
    if (theEvent.isAssignableFrom(Textfield.class))
      value = theEvent.getStringValue();
    else if (theEvent.isAssignableFrom(Toggle.class) || theEvent.isAssignableFrom(Button.class))
      value = theEvent.getValue()+"";
    else if (theEvent.isGroup())
    {
      parameter = theEvent.getController().getName();
      value = theEvent.getController().getStringValue();
    }

    plotterConfigJSON.setString(parameter, value);
    saveJSONObject(plotterConfigJSON, topSketchPath+"/plotter_config.json");
  }
  setChartSettings();
}
```

Рисунок 14 – Запись значений переменных в БД

С помощью обращения к функции, например .setValue(getConfigString("InvertorFreq")) поле с частотой инвертора получает ранее записанное в БД значение.

```
String getConfigString(String id) {
  String r = "";
  try {
    r = plotterConfigJSON.getString(id);
  }
  catch (Exception e) {
    r = "";
  }
  return r;
}
```

Рисунок 15 – Функция обращения к БД

3.3.1.5 Интерфейс приложения

Запуск приложения осуществлялся на персональном компьютере с операционной системой Windows 10. Сразу после запуска, пользователю отображаются все параметры, считываемые с микроконтроллера и вычисляемые в процессе работы программы (рисунок 16). Для начала работы пользователю требуется нажать по вкладке «Подключение МК» выбрать заранее известный порт и нажать «Открыть порт», далее по такому же алгоритму необходимо соединиться с инвертором во вкладке «Подключение инвертора». После установки связи с двумя устройствами программой будут приниматься пакеты данных от микроконтроллера и отображаться в реальном времени.

Пользователь может нажать на «Закрыть порт» для остановки получения данных и освобождения порта, например для взаимодействия с инвертором в другой программе.

Если какое-либо из устройств было подключено уже после запуска программы, то пользователь может нажать кнопку «Обновить порты» и получить необходимый порт в списке.

При получении данных программой рисуется график (рисунок 17), переключателями пользователь может скрыть или отобразить определенный параметр на гистограмме или графике. Также пользователь может изменять масштаб графика, переписывая значения осей Y. Отображение графиков производится с помощью библиотеки Realtime data plotter [4]. Дополнительно, пользователь может изменять масштаб конкретного отображаемого параметра, устанавливая ему коэффициент, например 0.01 или 100.

У пользователя есть возможность:

- Поставить программу на паузу, нажав клавишу «P»
- Начать запись процесса эксперимента нажатием клавиши «R», после завершения записи в папке с программой появляется каталог с текущей датой в имени, а в каталоге появляется файл с расширением txt, имеющий в

названии время начала записи, данные которого можно скопировать и перенести в редактор Microsoft Excel. В MS Excel по полученным данным допускается построить график и увидеть зависимость любого параметра, например от времени.

- Закрыть программу по нажатию «Q»

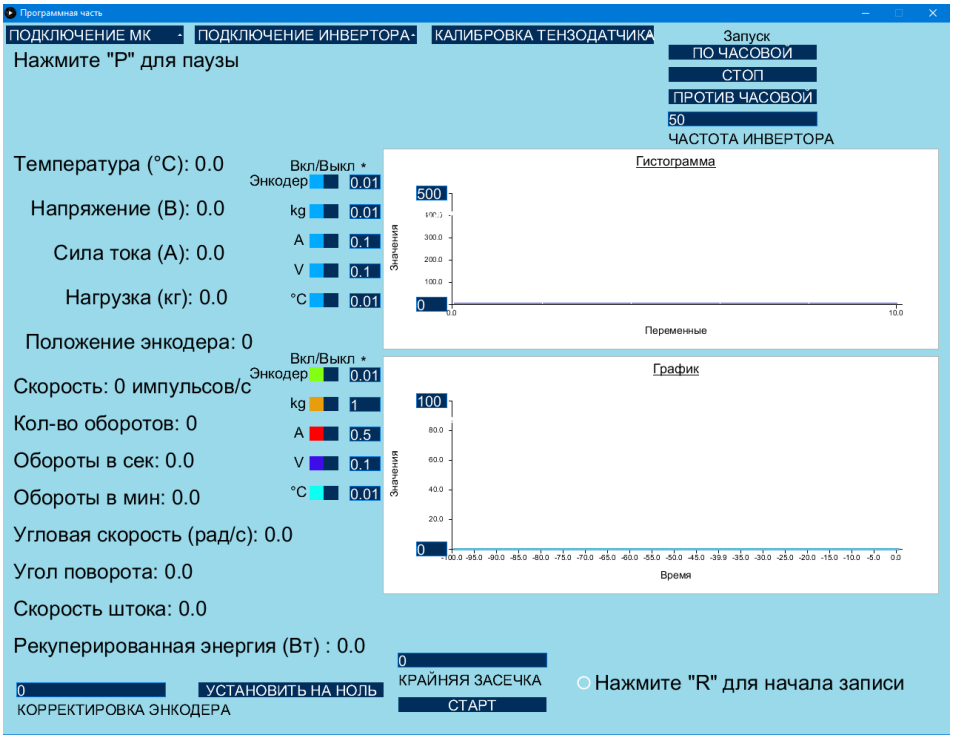


Рисунок 16 – Интерфейс начального экрана

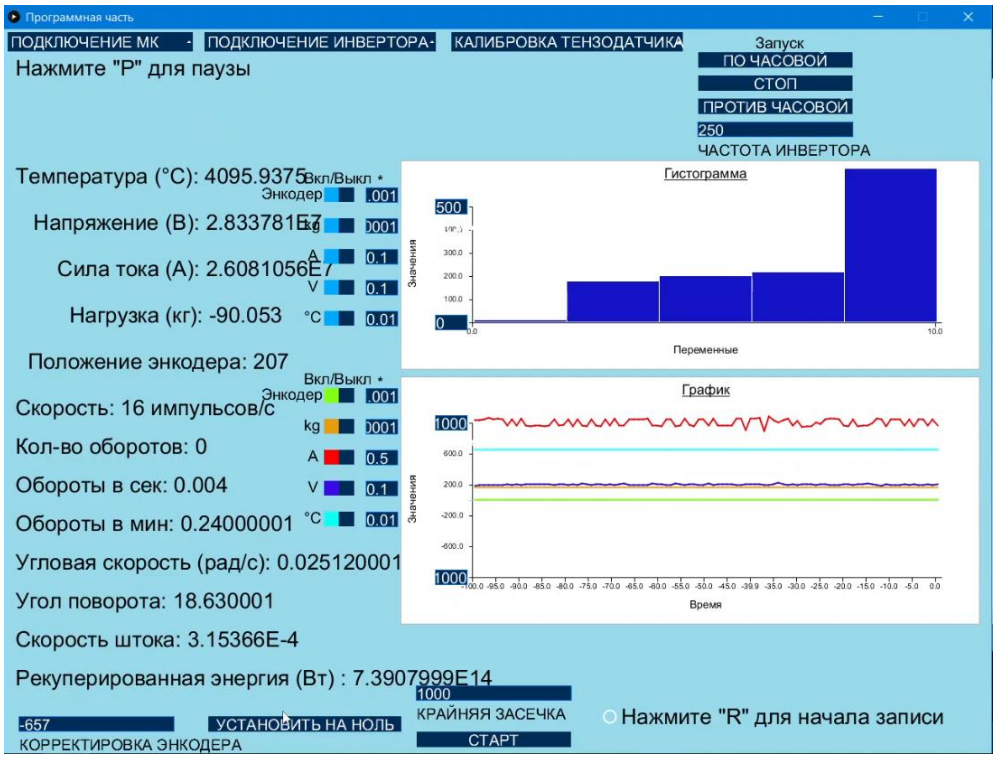


Рисунок 17 – Вывод информации и построение графиков

3.3.2 Управляющее воздействие для системы сбора данных динамических испытаний

3.3.2.1 Протокол MODBUS

В современном мире устройств автоматизации общение или диалог между различными устройствами встречается повсеместно в множестве различных ситуаций, используя различные типы сетей и протоколов. В зависимости от производителя оборудования для автоматизации процессов могут использоваться специфические закрытые протоколы или это может быть общий открытый промышленный протокол.

Преимуществом открытых протоколов является поддержка различными производителями (Siemens, Allen-Bradley, ABB) и поставщиками программного обеспечения (Modbus Utility, Simply Modbus Slave, BaseBlock), проектными и обслуживающими организациями, а также группами энтузиастов, которые улучшают и добавляют новые возможности.

Одним из наиболее распространенных промышленных протоколов связи электронных устройств, используемых сегодня в области автоматизации процессов является Modbus. Универсальность и открытость протокола позволяет интегрировать оборудование разных производителей.

Компания Modicon (в настоящее время принадлежит Schneider Electric) создала интерфейс связи Modbus для многоточечной сети на основе архитектуры ведущий и ведомые. Связь между узлами Modbus построена на основе сообщений типа запрос (request) и ответ (response). Modbus – это открытый стандарт, который описывает способ обмена сообщениями.

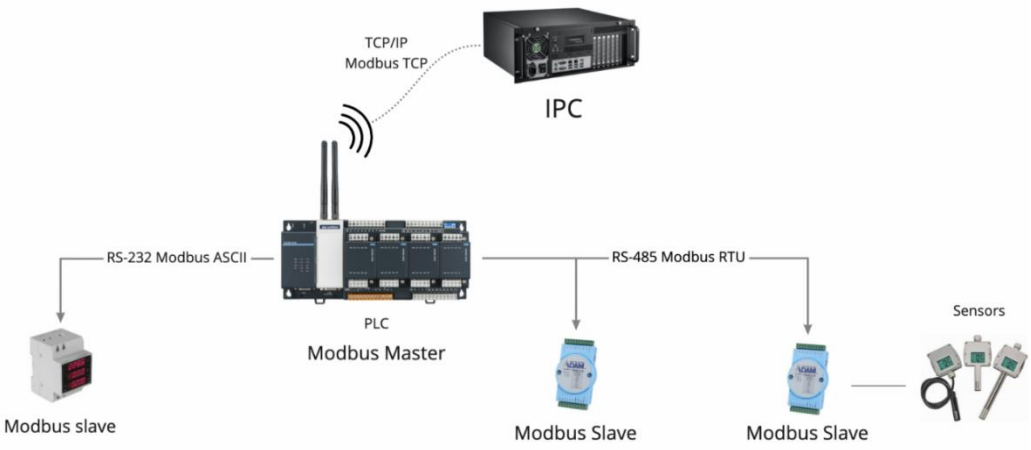


Рисунок 18 – Физический уровень Modbus

На физическом уровне Modbus (рисунок 18) может базироваться на нескольких типах соединений, таких как последовательные интерфейсы RS-232, RS-485, RS-422 (используются протоколы Modbus RTU/ASCII) и через Ethernet (Modbus TCP). Тип соединения выбирается во время приобретения устройств. Изначально интерфейс Modbus работал только через RS-232, но в более поздних

реализациях используется RS-485, потому что он поддерживает большие расстояния, высокие скорости и подключение нескольких устройств в одной сети.

Рисунок 19 – Различия протоколов Modbus

На логическом уровне в протоколе Modbus ASCII данные кодируются символами из таблицы ASCII и передаются в шестнадцатеричном формате. Начало каждого пакета обозначается символом двоеточия, а конец – символами возврата каретки и переноса строки. Это позволяет использовать протокол на линиях с большими задержками и оборудовании с менее точными таймерами.

В протоколе Modbus RTU данные кодируются в двоичный формат, и разделителем пакетов служит временной интервал. Этот протокол критичен к задержкам. При этом, накладные расходы на передачу данных меньше, чем в Modbus ASCII, так как длина сообщений меньше.

В протоколе Modbus TCP структура пакетов схожа с Modbus RTU, данные также кодируются в двоичный формат, и упаковываются в обычный TCP-пакет, для передачи по IP-сетям. Проверка целостности, используемая в Modbus RTU, не применяется, так как TCP уже имеет собственный механизм контроля целостности.

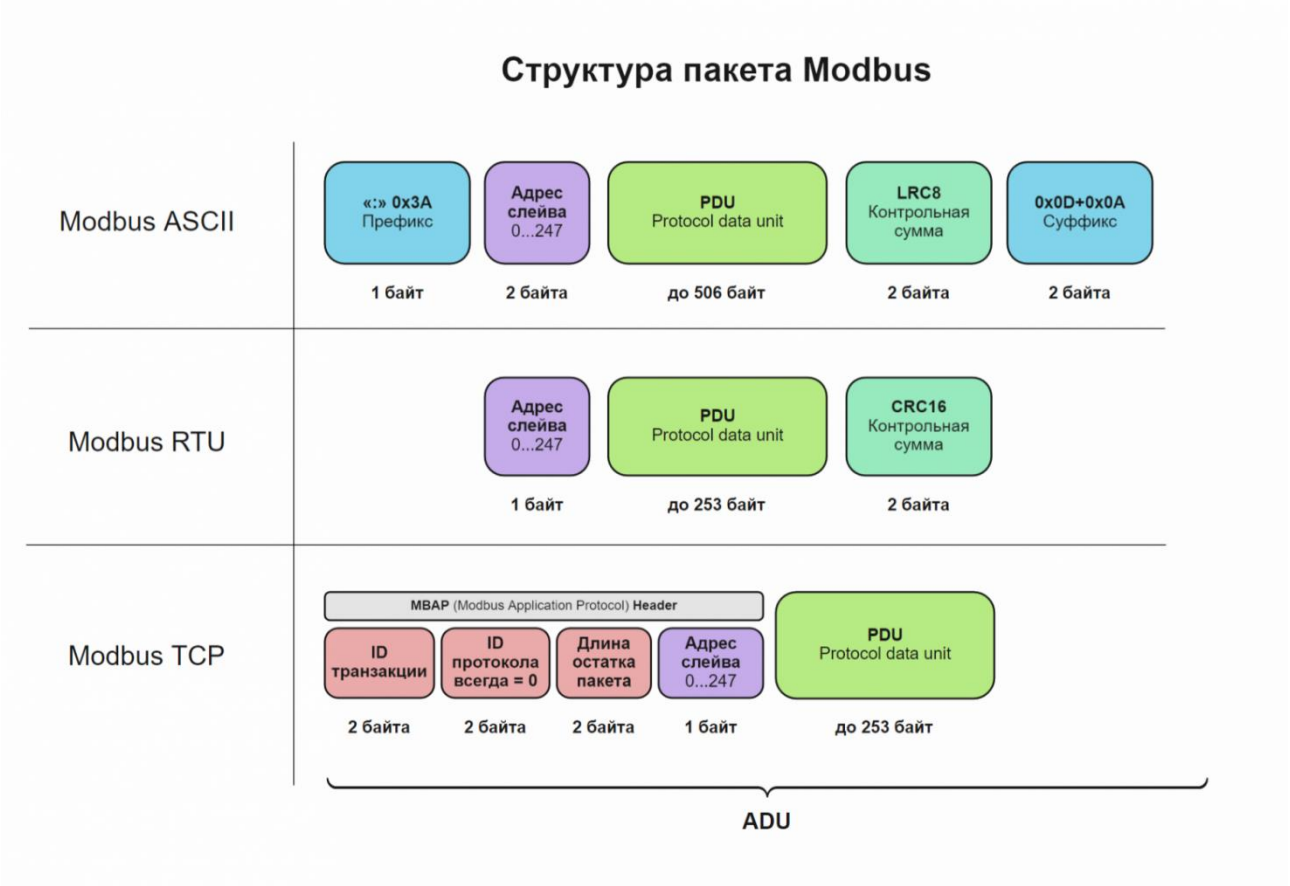


Рисунок 20 – Форматы пакета разных реализаций Modbus

Длина регистра	Read only	Read/Write
1 bit	02 - group read	Coils 01 - group read 05 - write single 15(F) - group write
16 bit	04 - group read	Holding Registers 03 - group read 06 - write single 16(0x10) - group write

Рисунок 21 – Коды функций Modbus PDU

Формат пакета протокола Modbus RTU (Remote Terminal Unit)

Взаимодействие Modbus устройств, следует master-slave модели. Инициировать запросы может только ведущее устройство, ведомые устройства только отвечают на запросы, и не могут самостоятельно начинать передачу данных.

ADU (Application Data Unit) – полный пакет Modbus, с основными частями, такими как заголовками, адресом, PDU, контрольной суммой и маркерами. Отличается, в зависимости от реализации протокола.

PDU (Protocol Data Unit) – основная часть пакета, одинаковая для всех реализаций протокола. Состоит из кода функций (рисунок 21) и данных. Коды функций изображены на рисунке 21.

Адрес устройства (ID) – адрес получателя, то есть ведомого устройства. В одном сегменте Modbus-сети могут находится до 247 устройств. Только slave-устройства имеют различающиеся адреса, master-устройство не имеет адреса. Адрес «0» используется для широковещательных запросов от master, при этом, slave-устройства не могут отвечать на эти широковещательные пакеты.

Контрольная сумма (CRC) – позволяет проверить целостность принятого и отправленного пакета. В Modbus RTU и ASCII представляет собой 2 байта. В Modbus RTU применяется алгоритм CRC16 (вычисляется с помощью умножения на полином), в Modbus ASCII – более простой и менее надежный LRC8. В Modbus TCP контрольная сумма не добавляется в ADU, так как целостность проверяется на уровне TCP [5].

3.3.2.2 Передача данных на инвертор

Протокол необходим для подачи команд старт/стоп и указания определенной частоты вращения двигателя по часовой или против часовой стрелки.

По документации [6] на имеющийся инвертор Yaskawa V1000 были определены адреса интересующих регистров (рисунок 22).

Register No.	Contents	
0000H	Reserved	
0001H	Operation Signals	
	bit 0	H5-12 = 0: Forward Run Command (0 = Stop, 1 = Run) H5-12 = 1: Run Command (0 = Stop, 1 = Forward Run)
	bit 1	H5-12 = 0: Reverse Run Command (0 = Stop, 1 = Run) H5-12 = 1: Forward/Reverse (0 = Stop, 1 = Reverse Run)
	bit 2	External Fault (EF0)
	bit 3	Fault Reset
	bit 4	Multi-Function Input Command 1 ComRef when set for Forward/Stop Note: If H1-01 = 40, then bit 4 becomes ComRef.
	bit 5	Multi-Function Input Command 2 ComCtrl when set for Reverse/Stop Note: If H1-02 = 42, then bit 5 becomes ComCtrl.
	bit 6	Multi-Function Input 3
	bit 7	Multi-Function Input 4
	bit 8	Multi-Function Input 5
	bit 9	Multi-Function Input 6
	bit A	Multi-Function Input 7
	bit B to bit F	Reserved
0002H	Frequency Reference	Varies by the setting units set to o1-03.
0003H	V/f Gain	
0004H-0005H	Reserved	
0006H	PID Target (0.01% signed)	
0007H	Analog Output 1 setting (10 V / 4000 H)	
0008H	Analog Output 2 setting (10 V / 4000 H)	

Рисунок 22 – Адреса и содержимое регистров

В поле вводится скорость в десятичном виде, где 1 соответствует 0.01 Гц.

Рассмотрим команду вращения двигателя по часовой стрелке с частотой 10 Гц. Формат отправляемых данных:

01 10 00 01 00 02 04 00 01 03 E8 63 1D

1 байт – адрес устройства (по умолчанию равен 1)
2 байт – код команды 10 (запись в несколько регистров)
3 и 4 байт – начальный регистр (00 01)
5 и 6 байт – количество регистров (00 02)
7 байт – количество байт информации 4
8 и 9 байт – запись значения 00 01 в первый регистр (команда запуска)
10 и 11 байт – запись значения частоты в шестнадцатеричном виде во второй регистр 1000 = 03E8
12 и 13 байт – контрольная сумма (CRC)

На рисунке 24 показана блок-схема подачи команды вращения двигателя с определенной частотой.

Контрольная сумма пакета вычисляется табличным методом, что уменьшает затраченное на процесс время [7]. На рисунке 23 изображена блок-схема функции вычисления контрольной суммы.

В процессе написания программы корректность вычисления контрольной суммы проверялась на стороннем ресурсе [8].

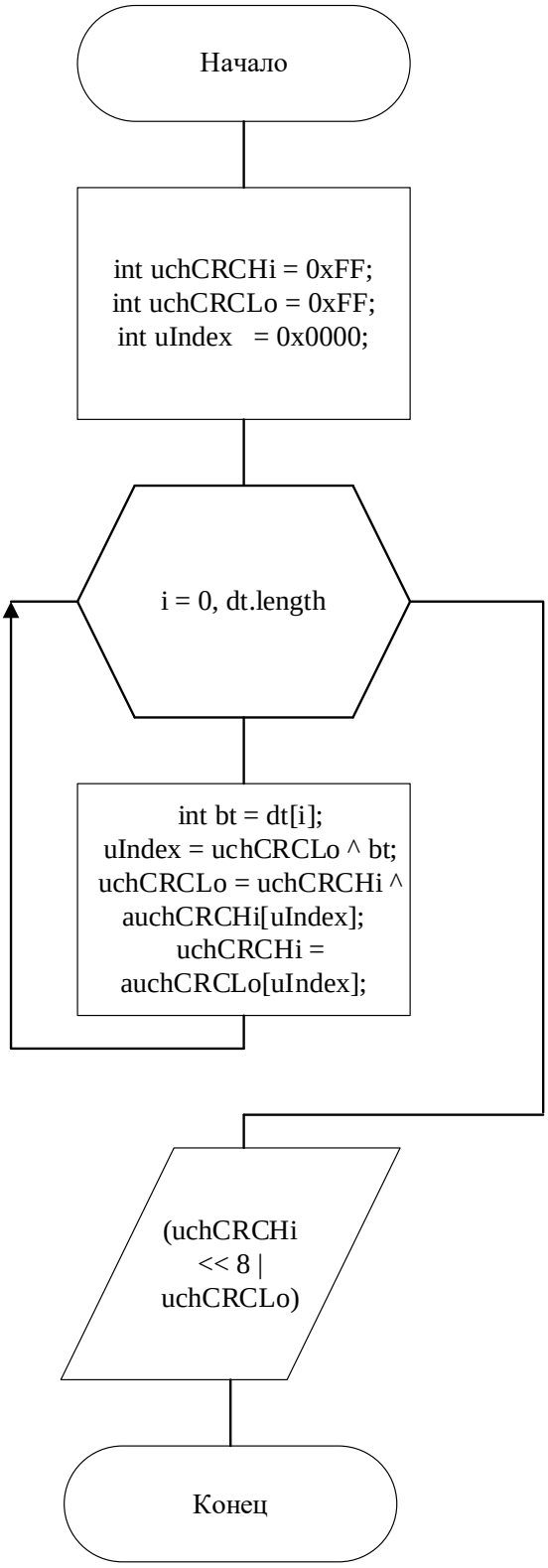


Рисунок 23 – Блок-схема функции вычисления контрольной суммы

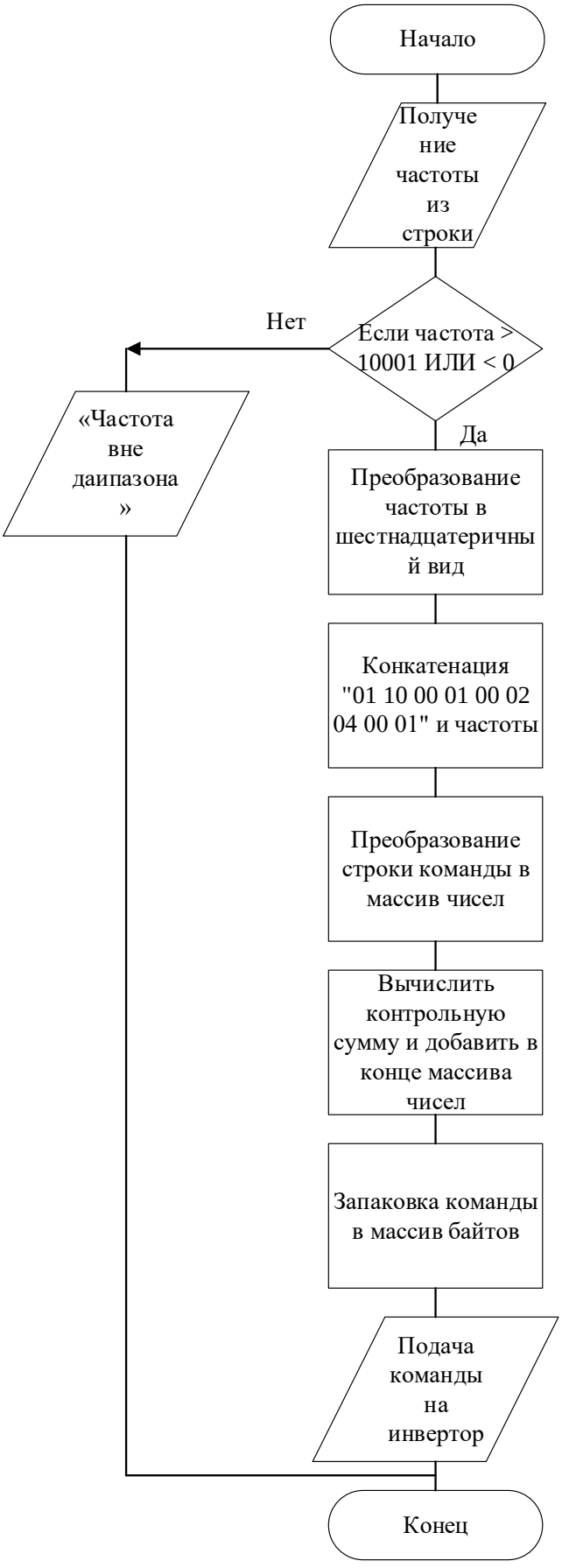


Рисунок 24 – Блок-схема подачи команды вращения двигателя с определенной частотой

3.3.2.3 Установка начального положения установки

После проведения эксперимента исследуемый объект не возвращался на исходную позицию, было принято решение реализовать данную функцию программным методом.

При запуске программы пользователю необходимо вручную выставить нулевое положение автомобильного узла и ввести в текстовое поле положение энкодера. После нажатия кнопки сравниваются значения текущего и введённого положения, если они отличаются, то на инвертор подаётся команда вращения с наименьшей скоростью для точного отслеживания текущего положения. В зависимости от разности двух величин вращение происходит в одну из сторон. При достижении заданного положения двигатель прекращает вращение. На рисунке 25 изображена блок-схема установки начального положения.

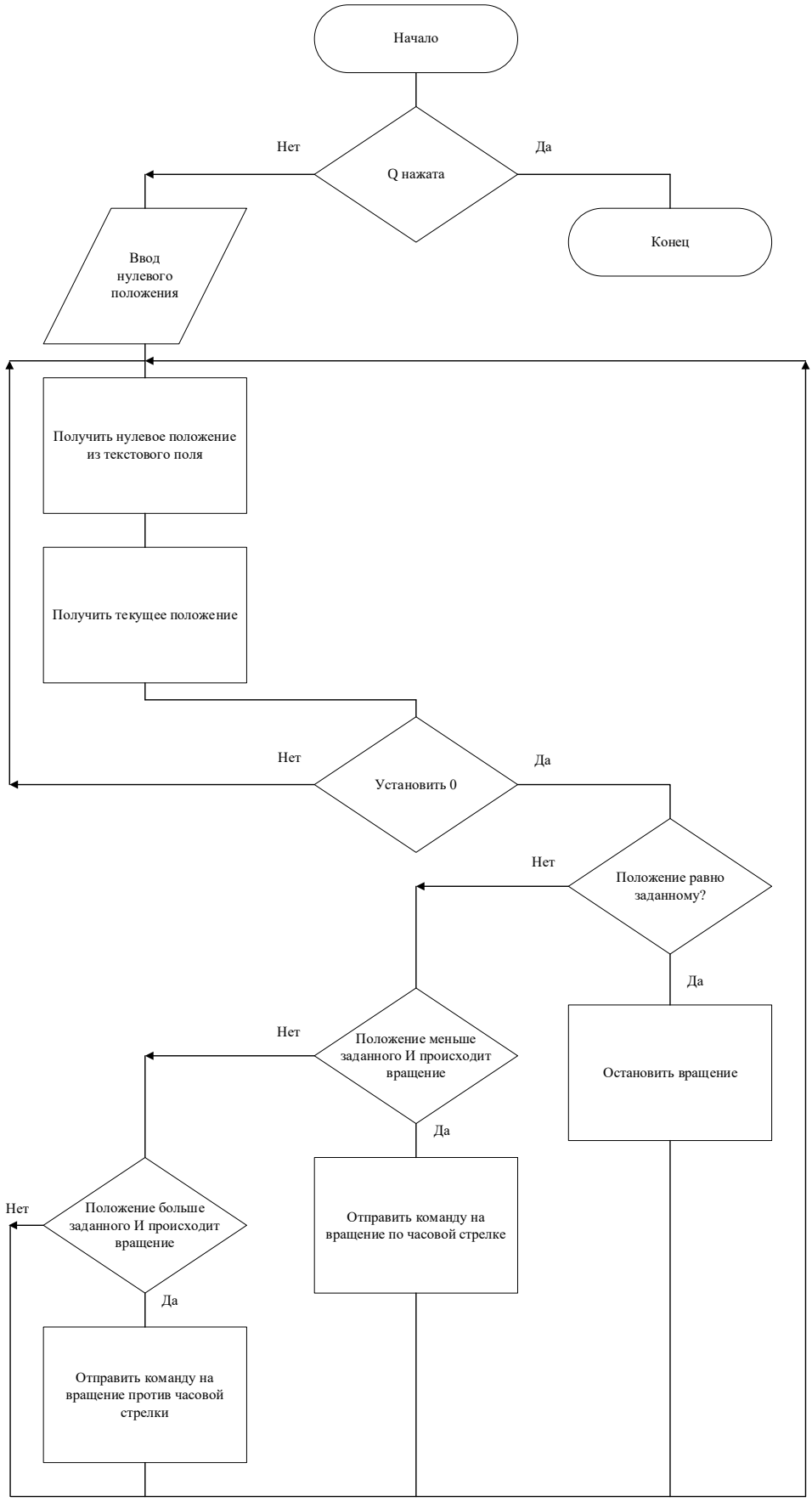


Рисунок 25 – Блок-схема установки начального положения

3.3.2.4 Перемещение штока на заданное расстояние

Чтобы физически не менять радиус кривошипа для проведения эксперимента требовалось ограничить максимальный ход, сделав его меньше 2r кривошипа. После установки нулевого положения требуется ввести желаемый ход штока. В программе исходя из указанного расстояния определяются крайняя левая и правая границы вращения. При подаче команды на проведение эксперимента из 0 положения подаётся команда вращения по часовой стрелке до крайнего левого положения, коснувшись его кривошип меняет направление своего вращения на противоположное, ожидая касания правого крайнего положения. Остановка проведения эксперимента произойдёт при нажатии кнопки «Стоп». Последовательность перемещения кривошипа изображена на рисунке 26. На рисунке 27 показана блок-схема алгоритма ограничения хода штока.

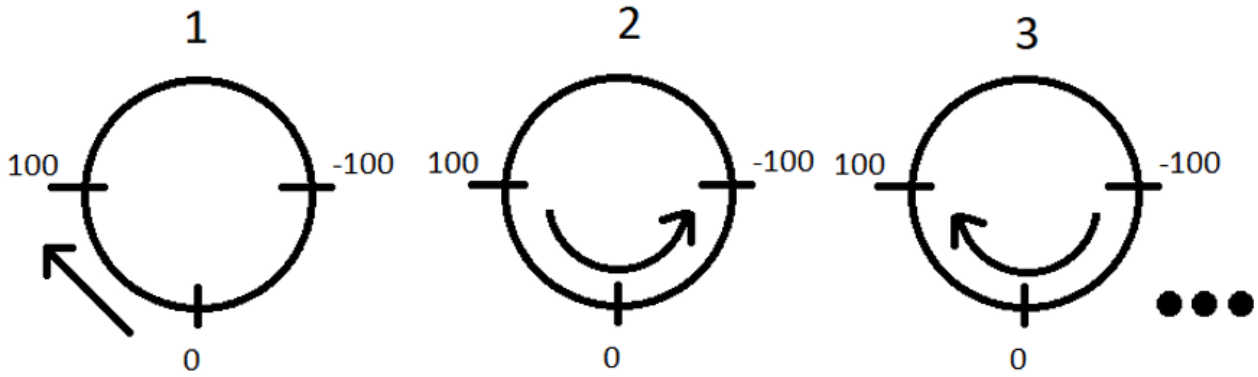


Рисунок 26 – Последовательность перемещения кривошипа

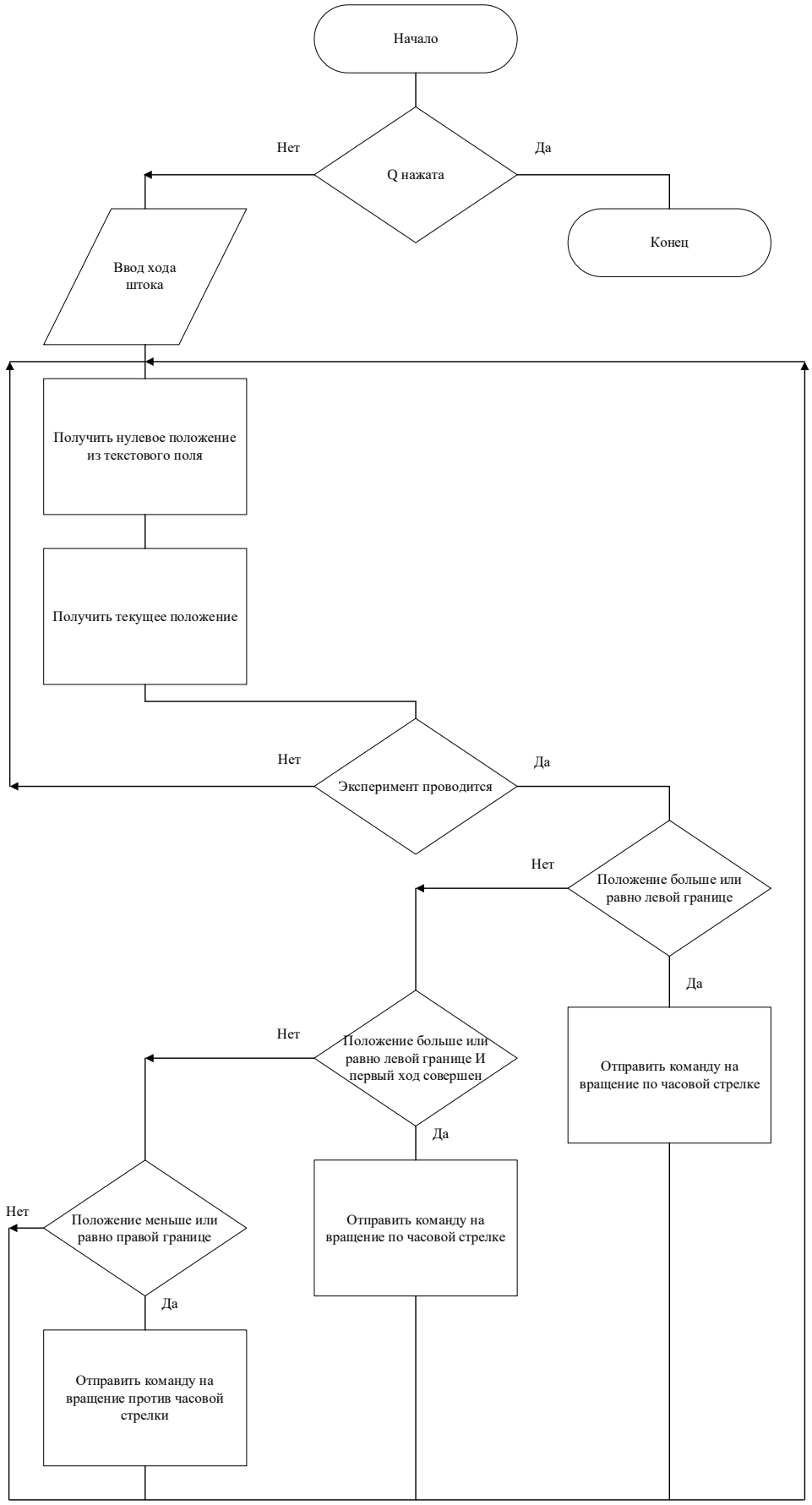


Рисунок 27 – Блок-схема ограничения хода штока

3.3.2.5 Измерение частоты вращения двигателя

Частота вращения двигателя определяется путём получения информации о положении энкодера в двух разных моментах времени, через секунду. Если системное время изменилось с четной секунды на нечетную вычисляется разность положений между двумя полученными моментами времени. Измерение производится раз в секунду с момента начала работы программы. Блок-схема измерения частоты вращения двигателя показана на рисунке 28.

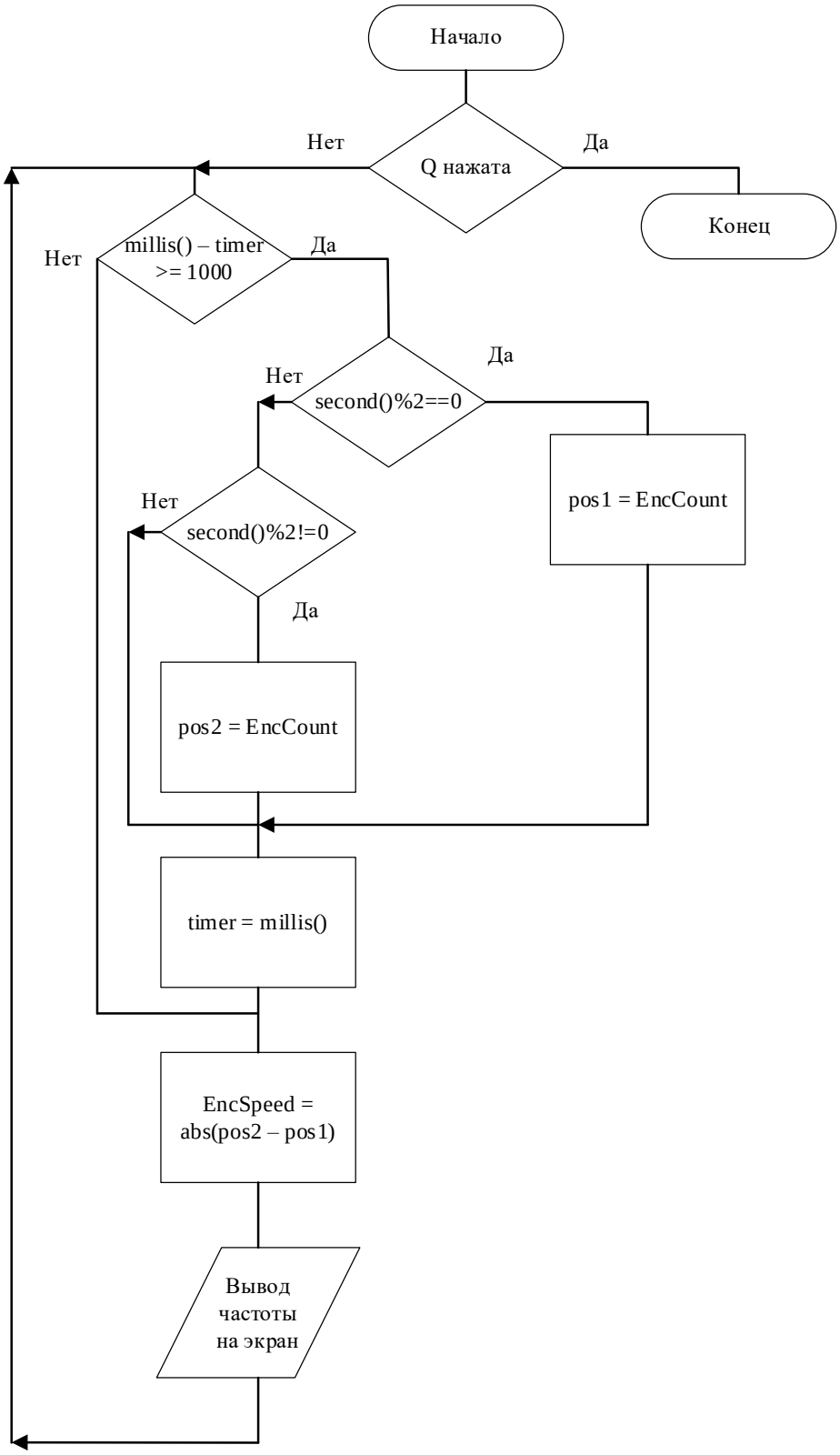


Рисунок 28 – Блок-схема измерения частоты вращения двигателя

3.4 Выводы по главе 3

В данном разделе был обоснован выбор языка и среды разработки программного обеспечения, а также:

- Описана работа части программы приема данных
- Проведена проверка работоспособности программы с микроконтроллером Arduino
- Подробно объяснено взаимодействие пользователя с графическим интерфейсом
- Описан протокол работы с промышленным оборудованием Modbus
- Рассмотрен формат отправляемых данных на инвертор
- Приведены описания и блок-схемы алгоритмов, требующиеся для взаимодействия с объектом испытаний.

4 Проведение испытаний

4.1 Измерение частоты вращения двигателя

Воспользовавшись ранее написанной функцией определения частоты вращения двигателя (в пункте 3.2.5 главы 3) и добавлением секундомера в программу, была составлена таблица 4. Было выявлено, что частота остаётся постоянной и напрямую зависит от частоты, подаваемой на инвертор.

Таблица 4 – Зависимость частоты инвертора и частоты вращения двигателя

Отображение импульсов/с	**Определение импульсов/с по секундомеру**	**Подаваемое значение на инвертор**	**Подаваемое значение в Гц**
100-108	108	100	1
280-294	282	250	2,5
570-580	570	500	5
1100-1120	1110	1000	10
2280-2290	2230	2000	20

В редакторе параметров официального программного обеспечения (рисунок 29) была найдена информация о том, что коэффициент передачи редуктора равен 17,57, а отношение эталона частоты инвертора к частоте вращения вала двигателя составляет 20. Разделив второй параметр на первый, получим 1,1383. Это даёт понимание, что 100 Гц, подаваемых на инвертор соответствуют 11383 импульсов в секунду (2,84 оборота двигателя за секунду), а полный оборот за секунду (4000 импульсов) происходит при частоте инвертора равной 35,14 Гц.

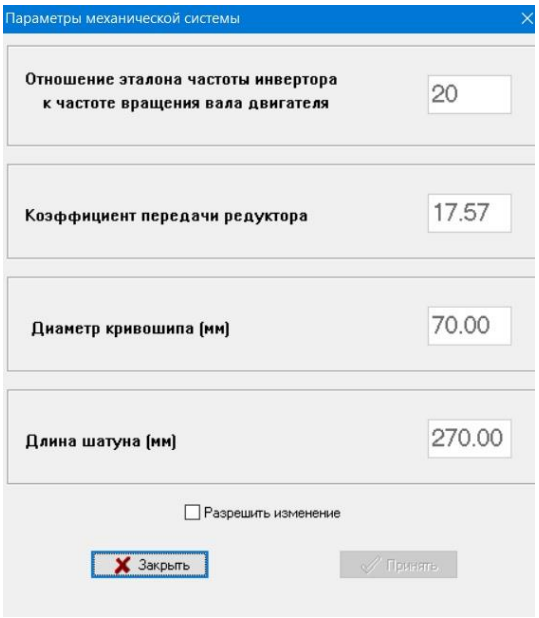


Рисунок 29 – Редактор параметров официального ПО

В результате частотных испытаний, было выявлено, что частота вращения электродвигателя равна заданной и всегда постоянна, так как инвертор будет поддерживать частоту увеличивая или уменьшая потребляемую мощность. Исходя из этого опыта стало возможным производить наблюдение за объектом испытаний при подаче заранее известной частоты.

4.2 Калибровка тензодатчика

От микроконтроллера компьютер получает числа, соответствующие показаниям тензодатчика от 0 до 16777215. В первую очередь в программе происходит преобразование числа в знаковый вид. После, на экране отображаются значения от -8388608 до 8388607. Во-вторых, следует провести корректировку тензодатчика, она происходит при отсутствии нагрузки на тензодатчик, пользователю требуется ввести число, которое установит показания примерно равными нулю (с погрешностью 200 единиц в , в поле «Корректировка тензодатчика» вводим значение равное 61400.

Далее, вкрутив шпильку с гайкой и тросом в тензодатчик массой 0,22 кг, а также грузиком массой 20,04 кг получаем общую массу 20,26 кг, в программе отображается значение 88800.

В поля калибровки тензодатчика вводим:

«Известная масса» - 20.26

«Показания при массе» - 88800

В результате на экране отображаются значения в пределах 20.24-20.28 кг.

Последним действием является проверка достоверности данных с использованием грузика другой массы.

Не меняя значений в полях ввода, извлекаем грузик массой 20,04 кг и вместо него используем грузик массой 2,04 кг. В программе отображаются верные значения в пределах 2.24-2.27 кг, если отображаемые значения не соответствуют заранее известным, то требуется повторно произвести корректировку или откалибровать тензодатчик используя грузик текущей массы, в дальнейшем проверив показания с использованием грузика другой массы.

Калибровка произведена успешно.

После выполнения калибровки тензодатчика извлекается грузик с удерживающим тросом на шпильке и устанавливается автомобильный амортизатор. По желанию пользователя дополнительно проводится корректировка тензодатчика на нагрузку амортизатора в нижней мертвой точке.

С течением времени показания тензодатчика будут отображать неверные данные, поэтому калибровку тензодатчика необходимо производить перед началом каждого эксперимента, снимая амортизатор и устанавливая грузики.

Показания одного килограмма тензодатчика примерно составляют 4383 единицам на аналого-цифровом преобразователе микроконтроллера.

На рисунке 30 представлена блок-схема алгоритма калибровки тензодатчика.

Рисунок 30 – Блок-схема алгоритма калибровки тензодатчика

4.3 Проверка производительности микроконтроллера при заданной нижней границе частоты вращения

В процессе установки нулевого положения автомобильного амортизатора микроконтроллер не успевал передать в программу актуальное положение энкодера, что вызывало пропуск желаемого положения двигателем и пользователю приходилось заново начинать запуск алгоритма. Было принято решение снизить частоту вращения двигателя при установке исходного положения до минимально возможной, чтобы микроконтроллер успел передать на компьютер текущее положение. Выяснилось, что минимальная допустимая частота вращения двигателем на инверторе была ограничена значением в 1,3 Гц, а Modbus команду с частотой вращения ниже этой инвертор не воспринимал. Программой DriveWizard Industrial [9] был изменён регистр [6], содержащий минимальную возможную частоту работы инвертора (рисунок 31), стандартное значение в регистре E1-09 1,3 Гц было изменено на 0,2 Гц.

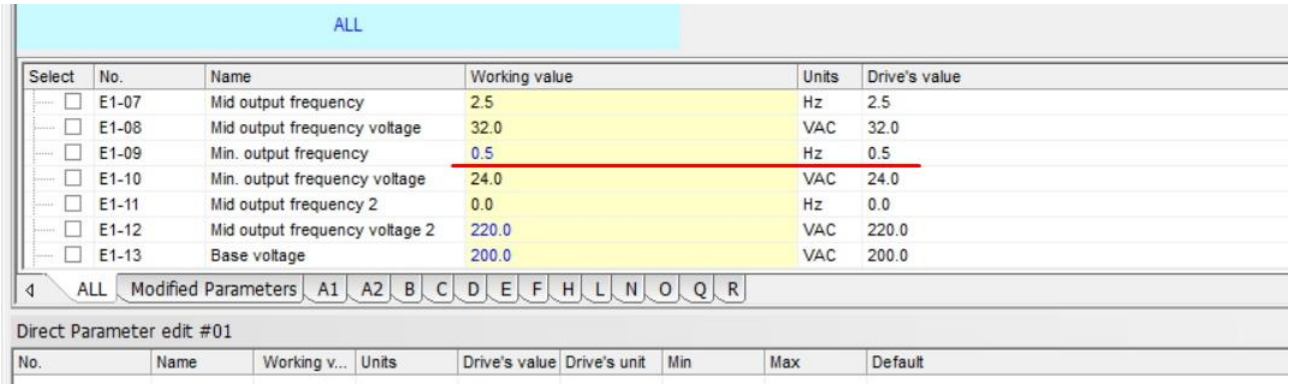


Рисунок 31 – Редактор регистров в DriveWizard Industrial

4.4 Выводы по главе 4

По результатам проведенных испытаний было установлено, что частота вращения электродвигателя равна заданной частоте инвертора, умноженной на коэффициент 1,1383, при этом в процессе испытаний частота остаётся неизменной.

В главе описан алгоритм калибровки и приведены рекомендации для более точного отображения значений, соответствующих реальным.

Была изменена минимальная частота вращения двигателя, оправляемая на инвертор. Это позволило уменьшить частоту вращения двигателя при установке нулевого положения. Установка более высокой частоты заставит алгоритм установки исходного положения работать некорректно.

ЗАКЛЮЧЕНИЕ

В результате выполнения бакалаврской работы было разработано программное обеспечение для системы управления динамическими испытаниями автомобильных узлов с применением микроконтроллерной системы для сбора данных и для подачи управляющих воздействий на инвертор.

Были сложности в понимании работы программы с постоянной отрисовкой (с главным циклом программы), так как программные задержки и циклы while замедляют работу всей программы и представляется возможным использование только условных операторов if и флагов с проверкой в разных частях программы.

Возникшие трудности в написании формул зависимости углов и перемещений кривошипно-шатунного механизма, были преодолены путём уточнения кинематических моделей компонентов объекта управления, позволяющие исключить недостатки и разработать ПО, позволяющее получать более достоверные оценки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. А.Н. Гоц Кинематика и динамика кривошипно-шатунного механизма поршневых двигателей – 2005 – 15с.
2. Processing [Электронный ресурс]: – Режим доступа: https://processing.org/reference.
3. Хабр [Электронный ресурс]: Статическая и динамическая типизация. - Режим доступа: https://habr.com/ru/post/308484/.
4. Sebastian Nilsson [Электронный ресурс]: Realtime data plotter. - Режим доступа: https://sebastiannilsson.com/project/realtime-plotter/realtime-data-plotter/.
5. Хабр [Электронный ресурс]: Как общаются машины: протокол Modbus. - Режим доступа: https://habr.com/ru/company/advantech/blog/450234/.
6. Yaskawa electric MEMOBUS/Modbus Communications - 2010– 274с.
7. Интеллект модуль [Электронный ресурс]: Краткое описание протокола MODBUS/RTU. - Режим доступа: https://intellect-module.ru/downloads/manuals/inode_35D/ModBus_RTU.pdf.
8. Lammert Bies [Электронный ресурс]: On-line CRC calculation and free library. - Режим доступа: https://www.lammertbies.nl/comm/info/crc-calculation.
9. Компоненты и системы промышленной автоматизации [Электронный ресурс]: YASKAWA DriveWizard Plus. - Режим доступа: https://www.cospa.ru/catalog/po-dlya-nastroyki-chastotnykh-preobrazovateley/drivewizard-plus/.
10. СТУ 7.5–07–2021 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности.

ПРИЛОЖЕНИЕ А
Фрагмент листинга кода программы

Project.pde

```
//импорт библиотек
import java.awt.Frame;
import java.awt.BorderLayout;
import controlP5.*; //импорт графической библиотеки
import processing.serial.*; //подключение библиотеки для работы с портами
//Создание объектов портов
Serial serial_stm32;
Serial serial_invertor;
ControlP5 cp5; //создание объекта cp5, отвечающего за интерфейс
int encoder_experiment_temp = 0;
boolean paused = false;
int metka = 0;
boolean rotate = false;
boolean rotate_temp = false;
boolean ExperimentRotate = false;
boolean ExperimentRotate_temp = false;
boolean ExperimentRotate_temp_clock = false;
boolean ExperimentRotate_temp_counterclock = false;
int PortSpeed = 9600;
String PortStmName;
String PortInvertorName;
// Переменная для отслеживания записи
boolean recording = false;
boolean labelswrite = false;
//инициализация считываемых переменных
int EncCount = 0;
float Tenz = 0;
float Current = 0;
float Voltage = 0;
float Temperature = 0;
int KorrEnc = 0;
int korr = 0;
int timer=0;
int pos1;
int pos2;
int difference = 0;
//переменные для работы с данными
int TimeInterval1 = 0;
int TimeInterval2 = 0;
```

```
int EncSpeed = 0;
int EncCut = 4000; //количество засечек на энкодере
int EncRounds = 0;
float EncRPS = 0;
float EncRPM = 0;
float EncUglSpeed = 0;
float EncAngle = 0;
float EncAngPerImp = 0.09; // 360/4000 градусы за прохождение засечки
float AmortSpeed = 0;
float DiamKrivoshipa = 0.07;
float RadiusKrivoshipa = 0.035;
float DlinaShatuna = 0.27;
JSONObject ConfigJSON; //Значения параметров сохранены в файле
PrintWriter output;
String topSketchPath = "";//помощник для сохранения исполняемого пути

void setup() {
  surface.setTitle("Программная часть"); //имя окна
  size(1280, 960); //установка разрешения экрана
  //базовая инициализация
  cp5 = new ControlP5(this); //инициализация объекта по правилам библиотеки
  cp5.setFont(createFont("Arial", 20)); //устанвока глобального шрифта
  textFont(createFont("Arial", 28)); //установка обычного шрифта
  //создание элементов графического интерфейса и установка их позиций,
размеров, именён
  cp5.addTextlabel("Encoder").setText("Энкодер").setPosition(330,
200).setColor(0);
  cp5.addTextlabel("Nagruzka").setText("kg").setPosition(385, 240).setColor(0);
  cp5.addTextlabel("Amper").setText("A").setPosition(390, 280).setColor(0);
  cp5.addTextlabel("Volt").setText("V").setPosition(390, 320).setColor(0);
  cp5.addTextlabel("Temperatura").setText("°C").setPosition(385, 360).setColor(0);
  cp5.addTextlabel("Encoder1").setText("Энкодер").setPosition(330,
460).setColor(0);
  cp5.addTextlabel("Nagruzka1").setText("kg").setPosition(385, 500).setColor(0);
  cp5.addTextlabel("Amper1").setText("A").setPosition(390, 540).setColor(0);
  cp5.addTextlabel("Volt1").setText("V").setPosition(390, 580).setColor(0);
  cp5.addTextlabel("Temperatura1").setText("°C").setPosition(385, 620).setColor(0);
  cp5.addButton("Start").setPosition(900,30).setSize(200,20).setLabel("По
часовой");
  cp5.addButton("Stop").setPosition(900,60).setSize(200,20).setLabel("Стоп");
  cp5.addButton("Reverse").setPosition(900,90).setSize(200,20).setLabel("Против
часовой");
```

```
cp5.addButton("SetToZero").setPosition(265,890).setSize(250,20).setLabel("Установить на ноль");
  Group stm32_port_select = cp5.addGroup("stm32_port_select").setPosition(5,30).setWidth(240).setBarHeight(25).setBackgroundHeight(216).setBackgroundColor(color(255,80)).setLabel("Подключение мк").setOpen(false);

cp5.addButton("RefreshStm32Port").setPosition(10,95).setSize(220,30).setLabel("Обновить порты").setGroup(stm32_port_select);

cp5.addButton("OpenStm32Port").setPosition(10,135).setSize(220,30).setLabel("Открыть порт").setGroup(stm32_port_select);

cp5.addButton("CloseStm32Port").setPosition(10,175).setSize(220,30).setLabel("Закрыть порт").setGroup(stm32_port_select);

cp5.addScrollableList("Stm32comlist").setPosition(10,10).setWidth(220).setItemHeight(25).setBarHeight(25).setLabel("Выбор порта").setGroup(stm32_port_select).close();
  Group invertor_port_select = cp5.addGroup("invertor_port_select").setPosition(260,30).setWidth(300).setBarHeight(25).setBackgroundHeight(216).setBackgroundColor(color(255,80)).setLabel("Подключение инвертора").setOpen(false);

cp5.addButton("RefreshInvertorPort").setPosition(10,95).setSize(220,30).setLabel("Обновить порты").setGroup(invertor_port_select);

cp5.addButton("OpenInvertorPort").setPosition(10,135).setSize(220,30).setLabel("Открыть порт").setGroup(invertor_port_select);

cp5.addButton("CloseInvertorPort").setPosition(10,175).setSize(220,30).setLabel("Закрыть порт").setGroup(invertor_port_select);

cp5.addScrollableList("Invertorcomlist").setPosition(10,10).setWidth(220).setItemHeight(25).setBarHeight(25).setLabel("Выбор порта").setGroup(invertor_port_select).close();

  //при запуске программы доступные порты добавляются в список
  RefreshStm32Port();
  RefreshInvertorPort();

  //файл сохранения настроек
  topSketchPath = sketchPath();
```

```
  ConfigJSON = loadJSONObject(topSketchPath+"/config.json");

  cp5.addTextlabel("Text1").setText("Запуск").setPosition(970,5).setColor(0);
  Group Tenz_Calbre =
cp5.addGroup("Tenz_Calbre").setPosition(580,30).setWidth(300).setBarHeight(25).s
etBackgroundHeight(100).setBackgroundColor(color(255,80)).setLabel("Калибровк
а тензодатчика").setOpen(false);
  cp5.addTextfield("Corr").setLabel("Корректировка тензодатчика").setPosition(-
240,
10).setColorCaptionLabel(0).setWidth(200).setAutoClear(false).setGroup(Tenz_Calb
re).setValue(getConfigString("Corr"));
  cp5.addTextfield("Massa").setLabel("Известная масса").setPosition(-240,
75).setColorCaptionLabel(0).setWidth(200).setAutoClear(false).setGroup(Tenz_Calb
re).setValue(getConfigString("Massa"));
  cp5.addTextfield("TenzMassa").setLabel("Показания при массе").setPosition(10,
75).setColorCaptionLabel(0).setWidth(200).setAutoClear(false).setGroup(Tenz_Calb
re).setValue(getConfigString("TenzMassa"));
  cp5.addTextfield("InvertorFreq").setLabel("Частота инвертора").setPosition(900,
120).setColorCaptionLabel(0).setWidth(200).setAutoClear(false).setValue(getConfig
String("InvertorFreq"));
  cp5.addTextfield("Korr1").setLabel("Корректировка энкодера").setPosition(20,
890).setColorCaptionLabel(0).setWidth(200).setAutoClear(false).setValue("0");
  cp5.addTextfield("Experiment").setLabel("Крайняя засечка").setPosition(535,
850).setColorCaptionLabel(0).setWidth(200).setAutoClear(false).setValue("0");

cp5.addButton("StartExperiment").setPosition(535,910).setSize(200,20).setLabel("C
тарт");
}

//вызов функций при нажатии кнопок
void RefreshStm32Port() {
  //Обновление доступных портов
  String list[] = Serial.list();
  cp5.get(ScrollableList.class, "Stm32comlist").addItems(list);
}

void Stm32comlist(int n) {
  PortStmName = Serial.list()[n];
}

void OpenStm32Port() {
  //инициализация порта, название, скорость
  serial_stm32 = new Serial(this, PortStmName, PortSpeed);
}
```

```
void CloseStm32Port() {
  if (serial_stm32 != null)
  {
  serial_stm32.stop();
  }
  else
  {
    println("нет подключения");
  }
}

//вызов функций при нажатии кнопок
void RefreshInvertorPort() {
  //Обновление доступных портов
  String list[] = Serial.list();
  cp5.get(ScrollableList.class, "Invertorcomlist").addItems(list);
}

void Invertorcomlist(int n) {
  PortInvertorName = Serial.list()[n];
}

void OpenInvertorPort() {
  //инициализация порта, название, скорость
  serial_invertor = new Serial(this, PortInvertorName, PortSpeed);
}

void CloseInvertorPort() {
  if (serial_invertor != null)
  {
  serial_invertor.stop();
  }
  else
  {
    println("нет подключения");
  }
}

void Start(){
  String freq = cp5.get(Textfield.class,"InvertorFreq").getText();
  int freq_int = int(freq);
  if (freq_int > 10001 | freq_int < 0)
  {
```

```
    println("Частота вне диапазона (макс 100гц)");
  }
  else
  {
    String freq_hex = hex(freq_int,4);
    String freq_hex_HI = freq_hex.substring(0,2);
    String freq_hex_LO = freq_hex.substring(2,4);
    String hx = "01 10 00 01 00 02 04 00 01" + " " + freq_hex_HI + " " +
freq_hex_LO;
    String[] list = split(hx, ' ');
    int listLength = list.length;
    int[] massive = new int[listLength];
    for (int i = 0; i < listLength; i++){
      massive[i] =  unhex(list[i]);
    }
    //вычислить контрольную сумму
    String CRC = hex(crc16st(massive),4);
    //добавить к пакету
    massive = append(massive,unhex(CRC.substring(2,4)));
    massive = append(massive,unhex(CRC.substring(0,2)));
    //запаковка команды в байты
    byte[] byteFrame = new byte[massive.length];
    for (int i = 0; i < massive.length; i++)
    {
      byteFrame[i] = (byte)massive[i];
    }
    serial_invertor.write(byteFrame);
  }
}

void Stop(){
    int[] massive1 =
{0x01,0x10,0x00,0x01,0x00,0x02,0x04,0x00,0x00,0x03,0xE8,0x32,0xDD};
    byte[] byteFrame1 = new byte[massive1.length];
    for (int i = 0; i < massive1.length; i++)
    {
      byteFrame1[i] = (byte)massive1[i];
    }
    serial_invertor.write(byteFrame1);
    ExperimentRotate = false;
  ExperimentRotate_temp = false;
}

void Reverse(){
```

```
  String freq = cp5.get(Textfield.class,"InvertorFreq").getText();
  int freq_int = int(freq);
  if (freq_int > 10001 | freq_int < 0)
  {
    println("Частота вне диапазона (макс 100гц)");
  }
  else
  {
    String freq_hex = hex(freq_int,4);
    String freq_hex_HI = freq_hex.substring(0,2);
    String freq_hex_LO = freq_hex.substring(2,4);
    String hx = "01 10 00 01 00 02 04 00 02" + " " + freq_hex_HI + " " +
freq_hex_LO;
    String[] list = split(hx, ' ');
    int listLength = list.length;
    int[] massive = new int[listLength];
    for (int i = 0; i < listLength; i++){
      massive[i] =  unhex(list[i]);
    }
    //вычислить контрольную сумму
    String CRC = hex(crc16st(massive),4);
    //добавить к пакету
    massive = append(massive,unhex(CRC.substring(2,4)));
    massive = append(massive,unhex(CRC.substring(0,2)));
    //запаковка команды в байты
    byte[] byteFrame = new byte[massive.length];
    for (int i = 0; i < massive.length; i++)
    {
      byteFrame[i] = (byte)massive[i];
    }
    serial_invertor.write(byteFrame);
  }
}

void StartExperiment(){
  ExperimentRotate = true;
  ExperimentRotate_temp = true;
  ExperimentRotate_temp_clock = true;
}

void SetToZero()
{
  rotate = true;
  rotate_temp = true;
```

```
}

byte[] SendBytesArray (int[] modbus_array)
{
    byte[] byteFrame = new byte[modbus_array.length];
     for (int i = 0; i < modbus_array.length; i++)
    {
      byteFrame[i] = (byte)modbus_array[i];
    }
    return(byteFrame);
}
//реализация паузы
public void keyPressed() {
  if ( key == 'p' || key == 'P' ) {
    paused = !paused;
    if (paused) {
      noLoop();
    }
    else {
      loop();
    }
  }
  if ( key == 'q' || key == 'Q' ) {
    exit();
    return;
  }
  if (key == 'r' || key == 'R') {
    recording = !recording;
    if (recording)
    {
    String filename = str(hour()) + " " + str(minute()) + " " + str(second()) + ".txt";
    output = createWriter(filename);
    String[] labels = new String[8];
    labels[0] = "Секунды";
    labels[1] = "мс";
    labels[2] = "Скорость";
    labels[3] = "Кол-во оборотов";
    labels[4] = "Нагрузка";
    labels[5] = "Сила тока";
    labels[6] = "Напряжение";
    labels[7] = "Температура";
    if (labelswrite == false) {
    String[] lines = new String[1];
```

```
    lines[0] = labels[0] + "\t" + labels[1] + "\t" + labels[2] + "\t" + labels[3] + "\t" +
labels[4] + "\t" + labels[5] + "\t" + labels[6] + "\t" + labels[7];
    output.println(lines[0]);
      }
    }
    else{
      output.close();
    }
  }
}

void draw() {//отрисовка
  background(153,217,234);
  cp5.setFont(createFont("Arial", 20)); //устанвока глобального шрифта
  textFont(createFont("Arial", 28)); //установка обычного шрифта
  textAlign(LEFT);
  fill(0);
  text("Нажмите \"P\" для паузы", 15, 60);
  text("Температура (°C): " + Temperature, 15, 200);
  text("   Напряжение (В): " + Voltage, 15, 260);
  text("       Сила тока (А): " + Current, 15, 320);
  text("         Нагрузка (кг): " + Tenz, 15, 380);
  text("  Положение энкодера: " + KorrEnc, 15, 440);
  if (!recording) {
    text("Нажмите \"R\" для начала записи", 800, 900);
  }
  else {
    text("Нажмите \"R\" для остановки записи", 800, 900);
  }
  //установка нуля
  if (rotate == true){
    if (KorrEnc%4000 == 0){
      int[] stop_massive =
{0x01,0x10,0x00,0x01,0x00,0x02,0x04,0x00,0x00,0x00,0x32,0xB3,0xB6};
      serial_invertor.write(SendBytesArray(stop_massive));
      print("нулевое положение установлено");
      rotate = false;
      rotate_temp = false;
    }
  else if((KorrEnc%1000 > 500) && (rotate_temp == true)){
  int[] start_massive =
{0x01,0x10,0x00,0x01,0x00,0x02,0x04,0x00,0x01,0x00,0x32,0xE2,0x76};
  serial_invertor.write(SendBytesArray(start_massive));
  rotate_temp = false;
```

```
    }
  else if((KorrEnc%1000 < 500) && (rotate_temp == true)){
  int[] reverse_massive =
{0x01,0x10,0x00,0x01,0x00,0x02,0x04,0x00,0x02,0x00,0x32,0x12,0x76};
  serial_invertor.write(SendBytesArray(reverse_massive));
  rotate_temp = false;
    }
  }
  //Проведение эксперимента
  metka = int(cp5.get(Textfield.class,"Experiment").getText());
  if (ExperimentRotate == true){
    if (KorrEnc > 62000){
      encoder_experiment_temp = KorrEnc - 64000;
    }
    else{
      encoder_experiment_temp = KorrEnc;
    }
  if((encoder_experiment_temp >= metka) &&
(ExperimentRotate_temp_counterclock == true)){
  Reverse();
  ExperimentRotate_temp_clock = true;
  ExperimentRotate_temp_counterclock = false;
  }
  else if((encoder_experiment_temp >= -metka) && (ExperimentRotate_temp ==
true)){
  Start();
  ExperimentRotate_temp_counterclock = true;
  ExperimentRotate_temp_clock = false;
  ExperimentRotate_temp = false;
  }
  else if((encoder_experiment_temp <= -metka) && (ExperimentRotate_temp_clock
== true)){
  Start();
  ExperimentRotate_temp_counterclock = true;
  ExperimentRotate_temp_clock = false;
  }
}

  //Точка становится красной при записи
  stroke(255);
  if (recording) {
    fill(255, 0, 0);
  } else {
    noFill();
```

```
  }
  ellipse(785, 890, 16, 16);
  //считывание с порта и обновление данных
  //если порт не подключен, ждём подключения
  if (serial_stm32 != null && serial_invertor != null ) {
    if (serial_stm32.available() > 0){// && (serial_invertor.available() > 0)) {
    String myString = "";
    //считывание строки, перенос - конец строки
    myString = serial_stm32.readStringUntil('\n');
    myString = myString.trim();
    //массив строк для разделения данных (по запятой)
    String[] nums = split(myString, " ");

    //преоразование массива строк в численное значение
    EncCount = int(nums[0]);
    Tenz = float(nums[1]);
    //3.3/4095
    Current = float(nums[2])*13513.5;
    Voltage = float(nums[3])*13513.5;
    Temperature = float(nums[4])/16;
  //перевод позиции в абсолютную
  korr = int(cp5.get(Textfield.class,"Korr1").getText());
  if (EncCount + korr >= 64000){
    KorrEnc = EncCount - 64000 + korr;
  }
  else{
    KorrEnc = EncCount + korr;
    if (KorrEnc < 0){
      KorrEnc = 64000 + EncCount + korr;
    }
  }
  if (Tenz > 8388607){
    Tenz = Tenz - 16777216;
  }
  //Корректировка показаний
  int Coeff =
int(cp5.get(Textfield.class,"TenzMassa").getText())/int(cp5.get(Textfield.class,"Mass
a").getText());
  Tenz = (Tenz + int(cp5.get(Textfield.class,"Corr").getText()))/Coeff;
  }
}
//Определение скорости
if (millis() - timer >= 1000) {
  if (second() % 2 == 0){
```

```
    pos1 = EncCount;
  }
  if (second() % 2 != 0){
    pos2 = EncCount;
  }
  timer = millis();
}
  difference = abs(pos2 - pos1);
  if (difference >= 32000){
  EncSpeed = 64000 - difference;
  }
  else{
    EncSpeed = abs(pos2 - pos1);
  }
  EncRounds = EncCount/EncCut;
  EncRPS = float(EncSpeed)/EncCut;
  EncRPM = 60 * EncRPS;
  EncUglSpeed = 6.28 * EncRPS;
  EncAngle = EncAngPerImp * KorrEnc;
  AmortSpeed = RadiusKrivoshipa * EncUglSpeed * (sin(radians(EncAngle))+
RadiusKrivoshipa/DlinaShatuna/2*sin(radians(2*EncAngle)));
  fill(0);
  text("Скорость: " + EncSpeed + " импульсов/с", 15, 500);
  text("Кол-во оборотов: " + EncRounds, 15, 550);
  text("Обороты в сек: " + EncRPS, 15, 600);
  text("Обороты в мин: " + EncRPM, 15, 650);
  text("Угловая скорость (рад/с): " + EncUglSpeed, 15, 700);
  text("Угол поворота: " + EncAngle, 15, 750);
  text("Скорость штока: " + AmortSpeed, 15, 800);
  text("Рекуперированная энергия (Вт) : " + Voltage*Current, 15, 850);
  if (recording) {
  output.println(second() + "\t" + millis() + "\t" + EncCount/*EncSpeed*/ + "\t" +
EncRounds + "\t" + Tenz + "\t" + Current + "\t" + Voltage + "\t" + Temperature);
  output.flush();
  }
}
//Запись названия переменных и их значений в БД
//Отслеживание событий
void controlEvent(ControlEvent theEvent) {
  if (theEvent.isAssignableFrom(Textfield.class) ||
theEvent.isAssignableFrom(Toggle.class) || theEvent.isAssignableFrom(Button.class)
|| theEvent.isGroup()) {
    String parameter = theEvent.getName();
    String value = "";
```

```
    if (theEvent.isAssignableFrom(Textfield.class))
      value = theEvent.getStringValue();
    else if (theEvent.isAssignableFrom(Toggle.class) ||
theEvent.isAssignableFrom(Button.class))
      value = theEvent.getValue()+"";
    else if (theEvent.isGroup()){
      parameter = theEvent.getController().getName();
      value = theEvent.getController().getStringValue();
    }
    ConfigJSON.setString(parameter, value);
    saveJSONObject(ConfigJSON, topSketchPath+"/config.json");
  }
}
//Функция обращения к БД
String getConfigString(String id) {
  String r = "";
  try {
    r = ConfigJSON.getString(id);
  }
  catch (Exception e) {
    r = "";
  }
  return r;
}
```

CRC_calculation.pde

```
int crc16st (int[] dt){
  int uchCRCHi = 0xFF;
  int uchCRCLo = 0xFF;
  int uIndex   = 0x0000;
  for (int i = 0; i < dt.length; i++) {
  int bt = dt[i];
  uIndex = uchCRCLo ^ bt;
  uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex];
  uchCRCHi = auchCRCLo[uIndex];
  }
  return (uchCRCHi << 8 | uchCRCLo);
}
```

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись
«20» 06 2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника
код – наименование направления

Система управления динамическими испытаниями автомобильных узлов. Программная часть
тема

Руководитель	23.06.22 подпись, дата	доцент, канд.техн.наук должность, ученая степень	В.Г. Серёдкин инициалы, фамилия
Выпускник	23.06.22 подпись, дата		Е.В. Соколов инициалы, фамилия
Нормоконтролер	2306.22 подпись, дата	доцент, канд.техн.наук должность, ученая степень	В.Г. Серёдкин инициалы, фамилия

Красноярск 2022