Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.И. Легалов
подпись инициалы, фамилия
«__»_____2016 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – «Информатика и вычислительная техника»
код и наименование специальности

Цифровой измеритель компрессии для роторно-поршневого двигателя
тема

Пояснительная записка

Руководитель _____ доцент, к.т.н В.И. Иванов
подпись, дата должность, учёная степень инициалы, фамилия

Выпускник _____ А.В. Смирнов
подпись, дата инициалы, фамилия

Нормоконтролер _____ В.И. Иванов
подпись, дата инициалы, фамилия

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 Анализ технического задания 8
1.1 Анализ существующих систем 8
1.2 Структурная схема системы 11
2 Обзор комплектующих системы 14
2.1 Микропроцессорный модуль 15
2.2 Индикатор 23
2.3 Датчик давления 14
2.4 Источник питания 24
3 Проектирование алгоритмов работы устройства 27
4 Обработка результатов измерений 33
5 Программирование устройства 37
6 Пользовательский интерфейс 42
7 Экономическое обоснование проекта 47
ЗАКЛЮЧЕНИЕ 48
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 50
ПРИЛОЖЕНИЕ А 51
ПРИЛОЖЕНИЕ Б 60

ВВЕДЕНИЕ

Одним из объективных показателей целостности и работоспособности двигателя внутреннего сгорания является компрессия. Компрессия отображает величину давления, создаваемого в цилиндре поршневого агрегата в конце такта сжатия. Значения данного параметра зависит от условий, при которых проводятся измерения, и в первую очередь от исправного состояния двигателя. По этой причине компрессия является первопричинным диагностическим параметром, который позволяет объективно оценить исправность двигателя.

Компрессия, как и любая другая случайная измерительная величина, измеряется при одинаковых стационарных условиях. Важно чтобы на момент измерения компрессии двигатель был достаточно прогрет до рабочей температуры, дроссель полностью открытый, а свечи во всех цилиндрах — выкручены, стартер работоспособен при этом аккумуляторная батарея полностью заряжена.

Для измерения компрессии в двигателях используется специальное устройство – компрессометр. Существует множество вариаций их исполнения, но по сути они отличаются между собой лишь конструкцией и исполнением. В целом, компрессометр состоит из наконечника, вставляемого в свечное отверстие, обратного клапана на входе и манометра. Манометр с наконечником могут быть соединены шлангом или металлической трубкой. Клапан в наконечнике компрессометра необходим для того, чтобы стрелка манометра при замере фиксировалась на уровне наибольшего давления, возникшего в цилиндре. Такой компрессометр справляется с поставленной задачей при замере показателей компрессии в двигателях внутреннего сгорания, где тепловая энергия превращается в механическую с помощью кривошипно-шатунного механизма. Такие двигатели используются повсеместно и принципы измерения компрессии в диагностических целях не меняются.

Появление двигателей внутреннего сгорания дало толчок к производству автомобилей, передвигающихся на жидком виде топлива. Двигатели на протя-

жении всей истории автомобилестроения эволюционировали: появлялись различные конструкции моторов. Одной из прогрессивных конструкций двигателей стал роторно-поршневой агрегат. Разработчиком роторно-поршневого двигателя стали инженеры компании NSU – Феликс Ванкель и Вальтер Фройде. И хотя основная роль в создании роторного двигателя принадлежит именно Фройде (второй участник проекта в это время работал над конструкцией иного двигателя), в автомобильной среде силовой агрегат известен как мотор Ванкеля. Эта силовая установка была собрана и испытана в 1957 году. Первым автомобилем, на который установили роторно-поршневой двигатель, стал спорткар NSU Spider, который развивал скорость 150 км/час при мощности мотора 57 лошадиных сил. Впоследствии роторные двигатели устанавливались на автомобили Citroen (GS Birotor), Mercedes-Benz (C111), Chevrolet (Corvette), ВАЗ (21018) и так далее. Но самый массовый выпуск моделей с роторным двигателем был налажен японской компанией Mazda. Начиная с 1964 года, компания произвела несколько автомобилей с подобным типом силовой установки, а первой стала модель Cosmo Sport. Самая известная модель с роторно-поршневым двигателем, которая выпускалась этим производителем – RX.

Конструкция роторно-поршневого двигателя существенно отличается от классического поршневого двигателя. В центре ротора имеется круглое отверстие, изнутри покрытое зубцами как у шестерёнки. В это отверстие вставлен вращающийся вал меньшего диаметра, также с зубцами, что обеспечивает отсутствие проскальзывания между ним и ротором. Отношения диаметров отверстия и вала подобраны так, чтобы вершины треугольника двигались по одной и той же замкнутой кривой, которая называется «эпитрохоида», — искусство Ванкеля как инженера заключалось в том, чтобы сначала понять, что это возможно, а потом всё точно рассчитать. В итоге, поршень, имеющий форму треугольника Рело, отсекает в камере, повторяющей форму найденной Ванкелем кривой, три камеры переменного объёма и положения [8] (рисунок 1.1).

Рисунок 1.1 – Роторно-поршневой двигатель Ванкеля

Конструкция роторно-поршневого ДВС позволяет реализовать любой четырехтактный цикл без применения специального механизма газораспределения. Благодаря этому факту РПД оказывается значительно проще обычного четырёхтактного поршневого двигателя, в котором в среднем почти на тысячу деталей больше. Герметизация рабочих камер в роторно-поршневом ДВС обеспечивается радиальными и торцевыми уплотнительными пластинами, прижимаемыми к статору ленточными пружинами, а также центробежными силами и давлением газа. Ещё одна его техническая особенность — это высокая «производительность труда». За один полный оборот ротора (то есть за цикл «впрыск, сжатие, воспламенение, выхлоп»), выходной вал совершает три полных оборота. В обычном поршневом двигателе таких результатов можно добиться только используя шестицилиндровый ДВС.

Как и в поршневом двигателе внутреннего сгорания, компрессия в роторно-поршневом двигателе является одним из важнейших диагностических показателей работоспособности и фактором оценки внутреннего состояния двигателя. Кроме того, частота измерения и точность оценки компрессии РПД имеет

критичное значение по сравнению с поршневым ДВС. Связано это с тем, что предельный износ радиальных и торцевых уплотнений, благодаря которым создается герметизация рабочих камер, приводит к их выпадению из пазов и разрушению рабочей поверхности.

В связи с тем, что замер компрессии производится через отверстие для свечи зажигания, а в роторно-поршневом двигателе одна и та же свеча используется для воспламенения топливно-воздушной смести в трёх камерах сгорания поочерёдно, функционала обычного компрессометра недостаточно. Обусловлено это тем, что такие компрессометры оснащены обратным клапаном, позволяющим фиксировать максимальное значение компрессии при замере. Если использовать обычный компрессометр для измерения значений компрессии на роторно-поршневом двигателе, то возможно будет зафиксировать лишь максимальное значение одной из трёх камер сгорания, чего недостаточно для объективной оценки.

Научная новизна. В работе предлагаются модели и методы решения задачи измерения компрессии в роторно-поршневых двигателях. Проведенные расчеты и предложенные методики позволит существенно повысить эффективность работы на станциях технического обслуживания автомобиля.

Практическая значимость. Результаты дипломной работы могут быть использованы в виде промышленного автомобилестроения, в различных организациях предприятиях, различных форм собственности занимающихся ремонтом двигателей, а также в учебном процессе специализированных средних технических и высших учебных заведений.

В качестве методов исследования были использованы следующие: анализ научной литературы, анализ деятельности, систематизация и обобщение.

Использование современной элементной базы при реализации таких систем играет немаловажную роль. Исследование высокоуровневых программных и отладочных комплексов выходит за рамки одной дипломной работы. Поэтому в данной работе устройство будет построено на макетных платах начального уровня Arduino.

Конечным этапом является написание программного обеспечения обеспечивающее полнофункциональную работу устройства

Работа состоит из введения, семи глав, заключения, списка литературы, графического материала, включает в себя иллюстративный материал в виде 1 рисунка. Объем работы составляет 50 страниц.

1 Анализ технического задания

1.1 Анализ существующих систем

Для измерения компрессии в классическом поршневом двигателе используется традиционное устройство – компрессометр. Компрессометр существует как электрический, так и механический.

Основным отличием механического компрессометра от электрического – это сам манометр, который характеризуется способом отображения результатов измерения компрессии. В остальном данные устройства фактически идентичны. Компрессометр состоит манометра, клапана сброса давления, шланга высокого давления и присоединительного наконечника, у которого резьба соответствует свечному отверстию.

Рисунок 1.2 – Компрессометры для классических поршневых двигателей

Различают так же компрессометры дизельных и бензиновых поршневых двигателей. Отличаются они диапазоном измерений и свечным наконечником, так как в дизельных моторах компрессия измеряется при подключении компрессометра на место свеч накала.

В роторно-поршневых двигателях процесс измерения компрессии несколько иной, что обусловлено его конструктивом.

На сегодняшний день, приобрести готовое устройство для измерения компрессии в роторно-поршневом двигателе практически невозможно. Обусловлено

это тем, что среднестатистическому владельцу транспортного средства, оснащенного роторно-поршневым двигателем, такое устройство требуется не слишком часто, из-за чего пропадает необходимость в покупке, а производителям выгоднее изготавливать их на заказ для специализированных сервисов. Соответственно, такие владельцы производят замер компрессии на станциях технического обслуживания.

Фактически, единственным производителем устройства для измерения компрессии в РПД является компания Mazda (Япония). Но эта компания, как основной производитель транспортных средств, оснащенных РПД, производит такие компрессометры для собственных нужд. Каждый официальный дилер Mazda занимается обслуживанием проданных автомобилей, а замер компрессии является одной из важнейших диагностических процедур, соответственно, изготовление такого устройство было необходимостью. Приобрести такое устройство для личного пользования, не являясь официальным представителем Mazda, практически невозможно.

Рисунок 1.3 – Компрессометр Mazda RE COMPRESSION TESTER

Устройство представляет из себя блок с дисплеями, клавишами, проводами для подключения к источнику питания и проводами, передающими инфор-

мацию с датчика давления. На дисплеях отображаются показания оборотов двигателя в момент замера компрессии и три значения компрессии, которые соответствуют каждой из трёх рабочих камер. Клавишами устройство включается и выключается, а также приводится к готовности. В качестве источника питания служит аккумуляторная батарея транспортного средства, подключенная специальными зажимами к проводам. Датчик давления вкручивается на место одной из двух свечей зажигания.

В связи с тем, что компрессометр Mazda приобрести практически невозможно, большинство владельцев РПД используют более сложный способ для замера компрессии. Идея заключается в том, что обычный компрессометр для поршневых двигателей модифицируется путём удаления обратного клапана. Таким образом, стрелка компрессометра не фиксируется на максимальном значении, а отображает текущее значение. Помимо этого, для фиксации измеренных значений используется видеокамера. После нескольких замеров, видеоматериал просматривается в замедленном времени, что позволяет увидеть каждое значение поочередно.

Становится понятно, что второй вариант является наиболее доступным, но очень неточным.

Еще одна попытка реализации устройства для измерения компрессией было представлено компанией Twisted Rotors.

Однако данные устройства выпускаются так же малыми партиями и их стоимость предельно высока, хоть и меньше чем стоимость оригинального Mazda RE COMPRESSION TESTER. Эти тенденции обуславливают актуальность данной работы и необходимость в разработке собственного микропроцессорного устройства для измерения компрессии.

Рисунок 1.4 – Компрессометр TR-01

1.2 Структурная схема системы

Устройство должно обеспечивать возможность измерения компрессии в роторно-поршневом двигателе. Для этого требуется:

– Измерение трёх значений одновременно для каждой рабочей камеры ротора.

– Приведение значений в перерасчёте оборотов двигателя к 250 об/мин.

Приведение необходимо для сверки полученных значений с указанными заводом изготовителем. Из-за центробежной силы, которая возникает во время вращения двигателя, уплотнительные элементы двигателя более плотно прижимаются к поверхности статора, что положительно влияет на компрессию. С другой стороны, при недостаточном количестве оборотов двигателя, компрессия может оказаться ниже, чем предполагается. В связи с этим, совместно с измерением значений компрессии необходимо вычислить скорость вращения двигателя в момент замера, а после окончания замера пересчитать значения с приведением к 250 об/мин вращения вала двигателя.

Реализация каждого из узлов требует необходимого подбора элементов, характерного для выполнения функционала каждого элемента структурной схемы. Структурная схема основных узлов блока передающей аппаратуры имеет вид:

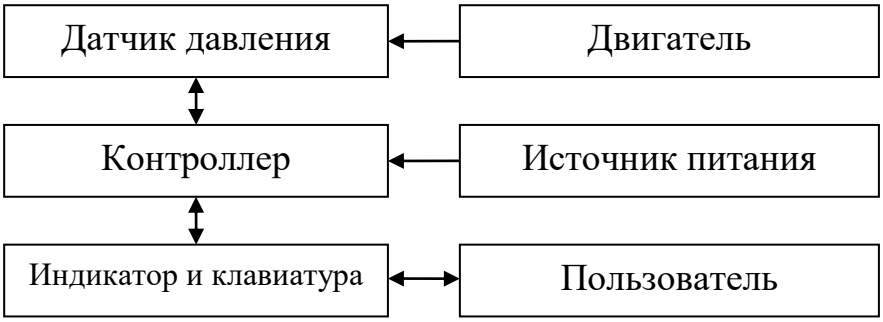

Рисунок 1.5 – Структурная схема устройства

Так как заводское максимальное значение компрессии в двигателе 13B-MSP от компании Mazda равно 8,5 кгс/см^2, потребуется датчик давления с верхним пределом значений около 10 кгс/см^2, выдающий аналоговый сигнал.

Аналоговый сигнал, полученный с датчика давления во время измерения компрессии, будет выглядеть следующим образом (Рисунок 1.6):

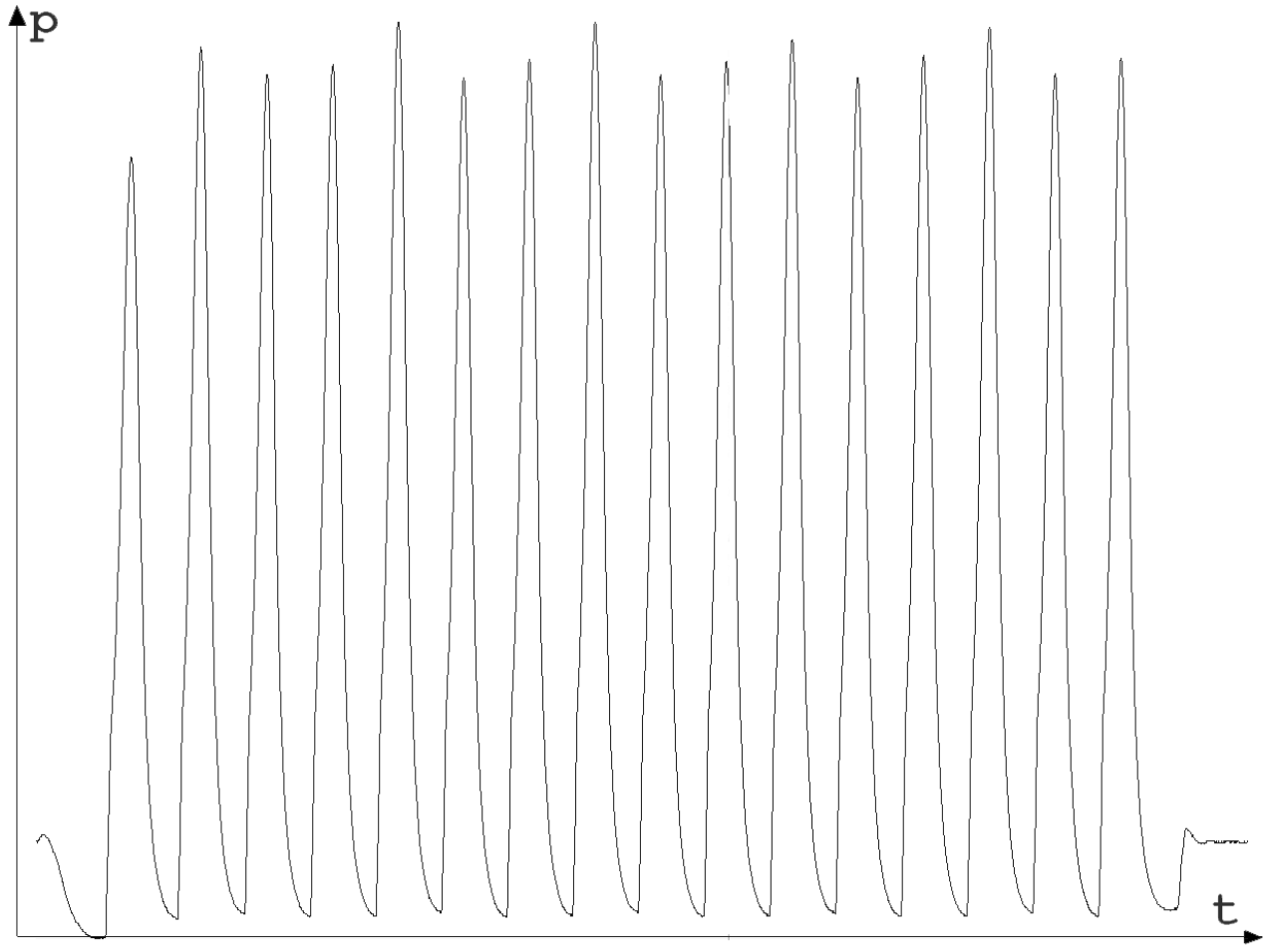

Рисунок 1.6 – Аналоговый сигнал датчика давления

Ось *t* определяет время, а ось *p* величину давления.

Для перехода к цифровой обработке сигналов необходимо определить ее роль в данной системе. Задачей цифровой обработки сигналов является своевременное преобразование аналогового сигнала датчика и кнопок в цифровые сигналы, для дальнейшей передачи и обработки. На этапе передачи сигналов микропроцессор встроенным АЦП проводит преобразование сигнала в цифровой бит после чего проводится цифровая обработка данного сигнала. Заключительным этапом является отображение данных датчиков на индикационном устройстве.

Объединить в себе устройство выполняющее ряд функций:

– Аналогово-цифрового преобразователя

– Устройства накопления аналоговых и цифровых данных

– Устройства преобразования информации для индикации

позволяет устройство на базе микропроцессора. Современные микропроцессоры обладают гибкой архитектурой, позволяющей обрабатывать всевозможные данные. Они являются неотъемлемой частью современных информационных технологий. На базе микропроцессоров строится огромное количество современных встраиваемых систем.

В связи с тем, что во время вращения двигателя стартером может происходить просадка питания бортовой сети автомобиля, предпочтительно использовать независимый источник питания. Кроме того, это избавит пользователя от подключения дополнительных проводов.

Для индикации результатов измерения потребуется ЖК-дисплей, способный отображать не менее 9 символов для отображения трёх результатов измерения с точностью до сотых. Для управления устройством потребуется клавиатура или как минимум 3 отдельные клавиши.

2 Обзор комплектующих системы

2.1 Датчик давления

Все серийно произведенные роторно-поршневые двигатели компании Mazda имеют верхний предел значения компрессии 8,5 кгс/см^2, но, путём механических доработок владельцами, достигается до 9,5 кгс/см^2. В связи с этим, для решения данной задачи потребуется датчик давления с верхним пределом значений около 10 кгс/см^2. Кроме того, в рабочей камере двигателя могут содержаться остатки горюче-смазочных материалов, а во время измерения компрессии попасть в датчик, поэтому датчик должен быть стойким к воздействию моторного масла и бензина.

Подходящим и самым доступным оказался датчик давления, представленный на рисунке 2.1:

Рисунок 2.1 – Внешний вид датчика давления

Технические характеристики:

– Напряжение питания 5В

– Диапазон измерения от 0 до 150 psi (от 0 до 10,5 кгс/см^2)

– Диапазон выходного напряжения от 0,5В при 0 кгс/см^2 до 4,5В при 10,5 кгс/см^2

– Предназначен для измерения давления воздуха, воды, масла, бензина, дизельного топлива, газа

Данный датчик полностью удовлетворяет требованиям, имеет низкую стоимость. Кроме того, имеются аналогичные датчики с такими же параметрами, но другими предельными значениями. При необходимости увеличения или уменьшения верхнего порога измерения достаточно будет заменить датчик.

2.1 Микропроцессорный модуль

Arduino представляет собой электронный конструктор и удобную платформу быстрого создания электронных интеллектуальных устройств для пользователей различной квалификации. Данная платформа имеет огромную популярность по всему земному шару благодаря удобному интерфейсу, простому языку программирования, а также благодаря открытой архитектуре и исходному программному коду. Устройства платформы программируются напрямую через порт USB без каких-либо дополнительных средств или программаторов.

Платы основе платформы Arduino могут быть приобретены в готовом виде или же быть собраны пользователем своими руками. Программное обеспечение является полностью открытым и свободно распространяемым. Исходные чертежи схем и печатных плат (файлы формата CAD) открыты и находятся в общем доступе, поэтому пользователи способны применять их на свое усмотрение.

Выбор Arduino, как основы будущего устройства, обусловлен следующими факторами:

1. Техническая совместимость с выбранным датчиком давления
2. Ценовая доступность
3. Широкая распространенность
4. Язык программирования высокого уровня
5. Отсутствие необходимости в приобретении дополнительных устройств для программирования
6. Большое количество дополнительных компонентов для дальнейшего расширения функционала

Далее представлены основные виды плат платформы Arduino:

Arduino Due (Рисунок 2.2) является новой платой на основе микроконтроллера 32bit Cortex-M3 ARM SAM3U4E.

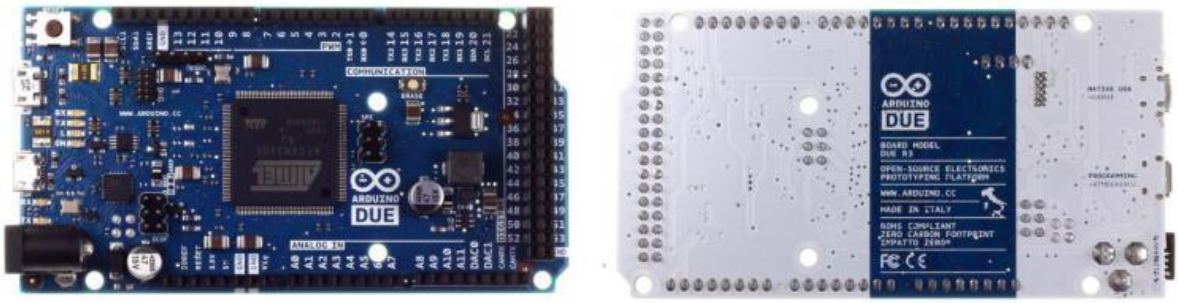

Рисунок 2.2 – Arduino Due

Arduino Leonardo (Рисунок 2.3) является последней версией платформы Arduino на микропроцессоре ATmega32u4. Отличием является разъем microUSB.

Рисунок 2.3 – Arduino Leonardo

Arduino Yun (Рисунок 2.4) – новое решение поддерживающее сеть WiFi на базе микроконтроллера ATmega32u4 и чипа беспроводной связи Atheros AR9331

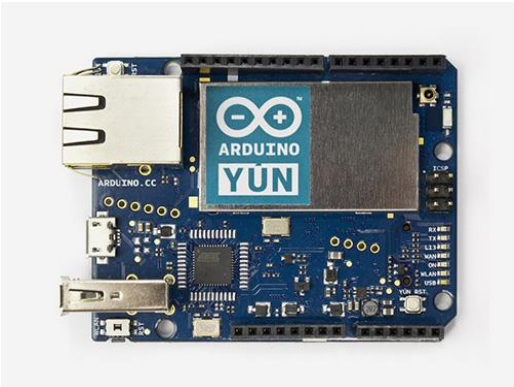

Рисунок 2.4 – Arduino Yun

Arduino Uno (Рисунок 2.5) является самой популярной версией базовой топологии Arduino USB. Arduino Uno поддерживает стандартный порт USB. Arduino Uno во многом подобна Duemilanove, но в отличии от нее содержит вспомогательный контроллер ATMega8U2 для связи с USB портом и обновленную более доступную маркировку вводов/выводов. Платформа адаптирована под различные платы расширения, например, пользовательскими платами с различными функциями.

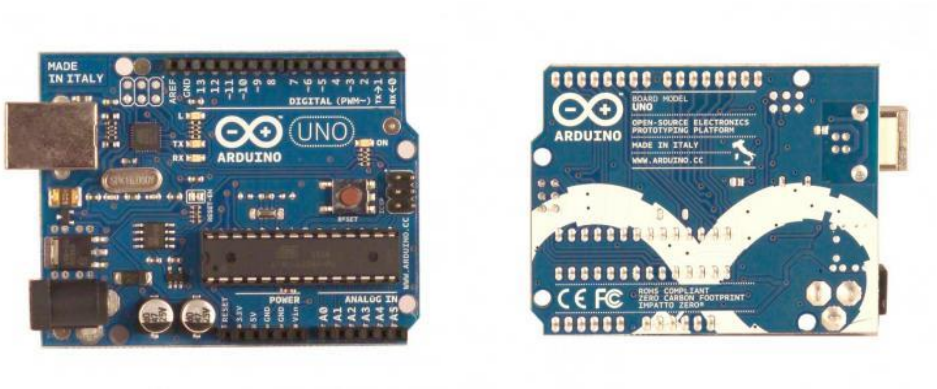

Рисунок 2.5 – Arduino Uno

Arduino Micro (Рисунок 2.6) — новое миниатюрное решение на основе ATmega32u4.

Рисунок 2.6 – Arduino Micro

Arduino Nano (Рисунок 2.7) — является компактной платформой, используемой для макетирования.

Рисунок 2.7 – Arduino Nano

Arduino Mega2560 (Рисунок 2.8) – обновленная версия платы серии Mega, которая строится на основе Atmega2560 и с применением чипа ATMega8U2 для подключение к USB порту.

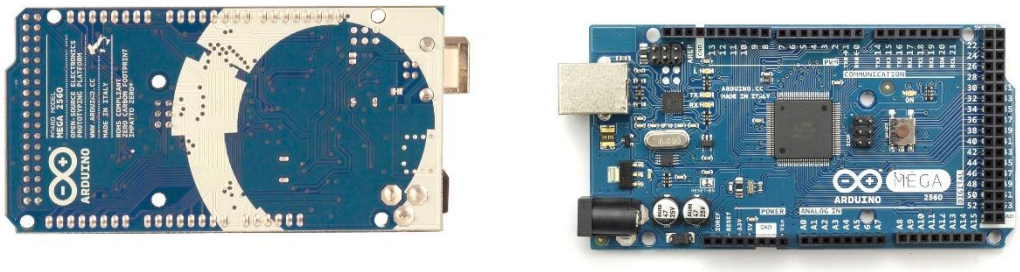

Рисунок 2.8 – Arduino Mega

Arduino Pro (Рисунок 2.9) – платформа, разработанная для опытных пользователей, может являться частью большего проекта. Она дешевле, чем Diecimila и может питаться от аккумуляторной батареи, но в тоже время требует дополнительной сборки и компонентов.

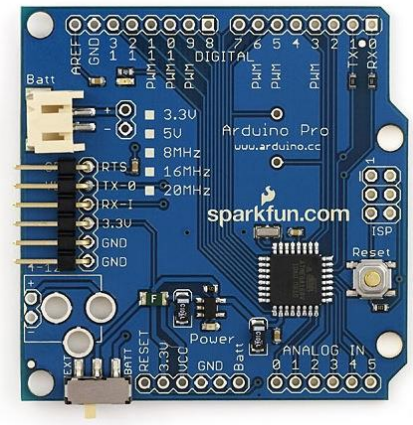

Рисунок 2.9 – Arduino Pro

Все представленные модели полностью удовлетворяют техническим требованиям, однако самой распространенной и доступной является Arduino Uno на базе микроконтроллера Atmel ATmega 328p.

В состав микропроцессорного модуля Arduino Uno входит все необходимое для удобной работы с микроконтроллером: 14 цифровых входов/выходов (из них 6 могут использоваться в качестве ШИМ-выходов), 6 аналоговых входов,

кварцевый резонатор на 16 МГц, разъем USB, разъем питания, разъем для внутрисхемного программирования (ICSP) и кнопка сброса. Для начала работы с устройством достаточно подать питание от AC/DC-адаптера или батарейки, либо подключить его к компьютеру посредством USB-кабеля.

В отличие от всех предыдущих плат Arduino, Uno в качестве преобразователя интерфейсов USB-UART использует микроконтроллер ATmega16U2 (ATmega8U2 до версии R2) вместо микросхемы FTDI.

Таблица 2.1 – Характеристики Arduino UNO

Микроконтроллер	ATmega328
Рабочее напряжение	5В
Напряжение питания (рекомендуемое)	7-12В
Напряжение питания (предельное)	6-20В
Цифровые входы/выходы	14 (из них 6 могут использоваться в качестве ШИМ-выходов)
Аналоговые входы	6
Максимальный ток одного вывода	40 мА
Максимальный выходной ток вывода 3.3V	50 мА
Flash-память	32 КБ (ATmega328)
SRAM	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактовая частота	16 МГц
Частота АЦП	100 мкс

"Uno" (в переводе с итальянского - "один") назван по случаю предстоящего выпуска Arduino 1.0. Совместно с Arduino 1.0 данные устройства будут базовыми версиями Arduino. Uno - эталонная модель платформы Arduino и является последней в серии USB-плат; для сравнения с предыдущими версиями.

В файлах проекта Arduino могут фигурировать микроконтроллеры ATmega8, 168 или 328. Например, в последних моделях используется микроконтроллер ATmega328, но на схеме может быть указан микроконтроллер ATmega8. Это не является ошибкой, поскольку все три микросхемы полностью совместимы между собой по выводам.

Arduino Uno может быть запитан от USB либо от внешнего источника питания - тип источника выбирается автоматически.

В качестве внешнего источника питания (не USB) может использоваться сетевой AC/DC-адаптер или аккумулятор/батарея. Штекер адаптера (диаметр - 2.1мм, центральный контакт - положительный) необходимо вставить в соответствующий разъем питания на плате. В случае питания от аккумулятора/батареи, ее провода необходимо подсоединить к выводам Gnd и Vin разъема POWER.

Напряжение внешнего источника питания может быть в пределах от 6 до 20 В. Однако, уменьшение напряжения питания ниже 7В приводит к уменьшению напряжения на выводе 5V, что может стать причиной нестабильной работы устройства. Использование напряжения больше 12В может приводить к перегреву стабилизатора напряжения и выходу платы из строя. С учетом этого, рекомендуется использовать источник питания с напряжением в диапазоне от 7 до 12В.

Ниже перечислены выводы питания, расположенные на плате:

VIN. Напряжение, поступающее в Arduino непосредственно от внешнего источника питания (не связано с 5В от USB или другим стабилизированным напряжением). Через этот вывод можно как подавать внешнее питание, так и потреблять ток, когда устройство запитано от внешнего адаптера.

5V. На вывод поступает напряжение 5В от стабилизатора напряжения на плате, вне независимости от того, как запитано устройство: от адаптера (7 - 12В), от USB (5В) или через вывод VIN (7 - 12В). Запитывать устройство через выводы 5V или 3V3 не рекомендуется, поскольку в этом случае не используется стабилизатор напряжения, что может привести к выходу платы из строя.

3V3. 3.3В, поступающие от стабилизатора напряжения на плате. Максимальный ток, потребляемый от этого вывода, составляет 50 мА.

GND. Выводы земли.

IOREF. Этот вывод предоставляет платам расширения информацию о рабочем напряжении микроконтроллера Arduino. В зависимости от напряжения, считанного с вывода IOREF, плата расширения может переключиться на соответствующий источник питания либо задействовать преобразователи уровней, что позволит ей работать как с 5В, так и с 3.3В-устройствами.

Память. Объем флэш-памяти ATmega328 составляет 32 КБ (из которых 0.5 КБ используются загрузчиком). Микроконтроллер также имеет 2 КБ памяти SRAM и 1 КБ EEPROM (из которой можно считывать или записывать информацию).

Входы и выходы. С использованием функций pinMode(), digitalWrite() и digitalRead() каждый из 14 цифровых выводов может работать в качестве входа или выхода. Уровень напряжения на выводах ограничен 5В. Максимальный ток, который может отдавать или потреблять один вывод, составляет 40 мА. Все выводы сопряжены с внутренними подтягивающими резисторами (по умолчанию отключенными) номиналом 20-50 кОм. Помимо этого, некоторые выводы Arduino могут выполнять дополнительные функции:

1. Последовательный интерфейс: выводы 0 (RX) и 1 (TX). Используются для получения (RX) и передачи (TX) данных по последовательному интерфейсу. Эти выводы соединены с соответствующими выводами микросхемы ATmega8U2, выполняющей роль преобразователя USB-UART.

2. Внешние прерывания: выводы 2 и 3. Могут служить источниками прерываний, возникающих при фронте, спаде или при низком уровне сигнала на этих выводах. Для получения дополнительной информации см. функцию attachInterrupt().

3. ШИМ: выводы 3, 5, 6, 9, 10 и 11. С помощью функции analogWrite() могут выводить 8-битные аналоговые значения в виде ШИМ-сигнала.

4. Интерфейс SPI: выводы 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). С применением библиотеки SPI данные выводы могут осуществлять связь по интерфейсу SPI.

5. Светодиод: Встроенный светодиод, подсоединенный к выводу 13. При отправке значения HIGH светодиод включается, при отправке LOW - выключается.

В Arduino Uno есть 6 аналоговых входов (A0 - A5), каждый из которых может представить аналоговое напряжение в виде 10-битного числа (1024 различных значения). По умолчанию, измерение напряжения осуществляется относительно диапазона от 0 до 5 В. Тем не менее, верхнюю границу этого диапазона можно изменить, используя вывод AREF и функцию analogReference(). Помимо этого, некоторые из аналоговых входов имеют дополнительные функции:

TWI: вывод A4 или SDA и вывод A5 или SCL. С использованием библиотеки Wire данные выводы могут осуществлять связь по интерфейсу TWI.

Помимо перечисленных на плате существует еще несколько выводов:

AREF. Опорное напряжение для аналоговых входов. Может задействоваться функцией analogReference().

Reset. Формирование низкого уровня (LOW) на этом выводе приведет к перезагрузке микроконтроллера. Обычно этот вывод служит для функционирования кнопки сброса на платах расширения

Arduino Uno предоставляет ряд возможностей для осуществления связи с компьютером, еще одним Arduino или другими микроконтроллерами. В ATmega328 имеется приемопередатчик UART, позволяющий осуществлять последовательную связь посредством цифровых выводов 0 (RX) и 1 (TX). Микроконтроллер ATmega16U2 на плате обеспечивает связь этого приемопередатчика с USB-портом компьютера, и при подключении к ПК позволяет Arduino определяться как виртуальный COM-порт. Прошивка микросхемы 16U2 использует стандартные драйвера USB-COM, поэтому установка внешних драйверов не требуется. В пакет программного обеспечения Arduino входит специальная программа, позволяющая считывать и отправлять на Arduino простые текстовые

данные. При передаче данных через микросхему-преобразователь USB-UART во время USB-соединения с компьютером, на плате будут мигать светодиоды RX и TX. (При последовательной передаче данных посредством выводов 0 и 1, без использования USB-преобразователя, данные светодиоды не задействуются).

В микроконтроллере ATmega328 также реализована поддержка последовательных интерфейсов I2C (TWI) и SPI. В программное обеспечение Arduino входит библиотека Wire, позволяющая упростить работу с шиной I2C. Для работы с интерфейсом SPI используется библиотека SPI.

2.3 Индикатор

Подходящим для индикации результата работы данного устройства является ЖК индикатор WH1602 (Рисунок 2.10).

Рисунок 2.10 – Внешний вид индикатора WH1602

Принцип работы данного элемента заключается в том, что разряды находятся на одной шине и за счет использования специализированных дешифраторов происходит преобразование двоичного кода в электрические сигналы, которые управляют данным индикатором.

Однако его использование несколько проблематично в связи с необходимостью дополнительных затрат на подключение к Arduino Uno большой проводной шиной данных.

Для ввода и вывода информации используется LCD keypad shield 1602. Данное устройство разработано специально для Arduino Uno и подключается к плате без дополнительных шин, использует дисплей WH1602 (16 символов по 2

строки) и имеет 5 функциональных клавиш, которых будет достаточно для управления устройством.

Рисунок 2.13 – Внешний вид LCD keypad shield 1602

2.4 Источник питания

Так как устройство является портативным, вопрос энергосбережения имеет важное значение. Кроме того, источник питания должен обеспечивать от 7 до 12 В, как рекомендовано производителем выбранной платы Arduino Uno.

Использование бортового напряжения автомобиля не рекомендовано по причине того, что во время работы стартера происходит просадка напряжения бортовой сети. Помимо этого, высокочастотные составляющие, вызванные работой системы зажигания, будут негативно сказывается на работе устройства вплоть до выхода его из строя.

В первую очередь устройство должно хранить все предварительные настройки в энергонезависимой памяти. Использование потребителей, которые используют большое количество тока недопустимо. Необходимо чтобы в течение всего цикла работы устройства обеспечивалось стабильное напряжение питающих цепей.

При этом, источник питания должен обладать минимальным весом из-за эргономических требований к устройству. Подходящим решением являются литий-полимерные аккумуляторные батареи (Li-Ion) и никель-кадмиевые (Ni-Ca) (Рисунок 2.15).

Рисунок 2.15 – Аккумуляторные источники энергии Li-Ion и Ni-Ca

Альтернативным вариантом является применение свинцово-кислотных аккумуляторов (Рисунок 2.16).

Рисунок 2.16 – Свинцово-кислотный источник энергии

Их использование недопустимо ввиду чувствительности к механическим колебаниям, больших габаритов и веса.

В данной работе в качестве источника питания применены батареи типа Крона (Рисунок 2.17).

Рисунок 2.17 – Источник питания типа «крона»

Это является самым простым, недорогим и доступным вариантом. Помимо этого, в комплекте к отладочным платам идут специализированные круглые jack разъемы 4.7мм для подключения питания к Arduino Uno.

3 Проектирование алгоритмов работы устройства

Основной функцией устройства измерения компрессии должно быть обеспечение режима работы, в котором возможно непрерывное получение данных с датчика в период вращения коленчатого вала. Реализация данной функции возможна при использовании соответствующих алгоритмов, допускающих универсальность применения устройства на роторных, двухтактных и четырехтактных двигателях внутреннего сгорания.

Входными данными, полученными с датчика, является аналоговый сигнал. При нулевом значении давления, поступающего в датчик, выходной сигнал равен 0,5В. При максимальном значении давления в 10,5 кгс/см^2 выходной сигнал равен 4,5В. С помощью встроенного АЦП получим данные, которые можно использовать для расчёта компрессии.

Для обеспечения нормальной работы устройства необходимо выполнить ряд предварительных настроек.

1. Инициализировать предварительные настройки
2. Инициализировать АЦП и датчик
3. Выбрать алгоритм измерения компрессии

Общий алгоритм работы устройства показан на рисунке 3.1. Рассмотрим более подробно условия и процессы работы алгоритма программы.

1. Инициализация устройства

1.1. Загрузка EEPROM

1.2. Проверка калибровки

1.3. Объявление прерываний

1.4. Объявление переменных и констант

1.5. Инициализация индикатора

2. Управление режимами работы устройства

2.1. Ожидание сигнала с клавиатуры

2.2. Выбор типа датчика

2.3. Выбор алгоритма измерения компрессии

2.4. Переход к измерению компрессии

3. Режим измерения компрессии

3.1. Инициализация датчика и установка нулевого значения

3.2. Чтение данных АЦП (с датчика)

3.3. Индикация показаний

3.4. Определение частоты оборотов двигателя

3.5. Переход к обработке результатов

4. Обработка результатов измерений

4.1. Определение поправки данных

4.2. Определение компрессии

4.3. Приведение оборотов к номинальным

5. Вывод результата

6. Обработка ошибок

6.1. Ошибка низких оборотов

6.2. Ошибка отсутствия или неисправности датчика

6.3. Индикация ошибки

7. Режим калибровки

7.1. Чтение значений клавиш

7.2. Запись значений клавиш в EEPROM

7.3. Индикация сообщения о необходимости перезапуска устройства

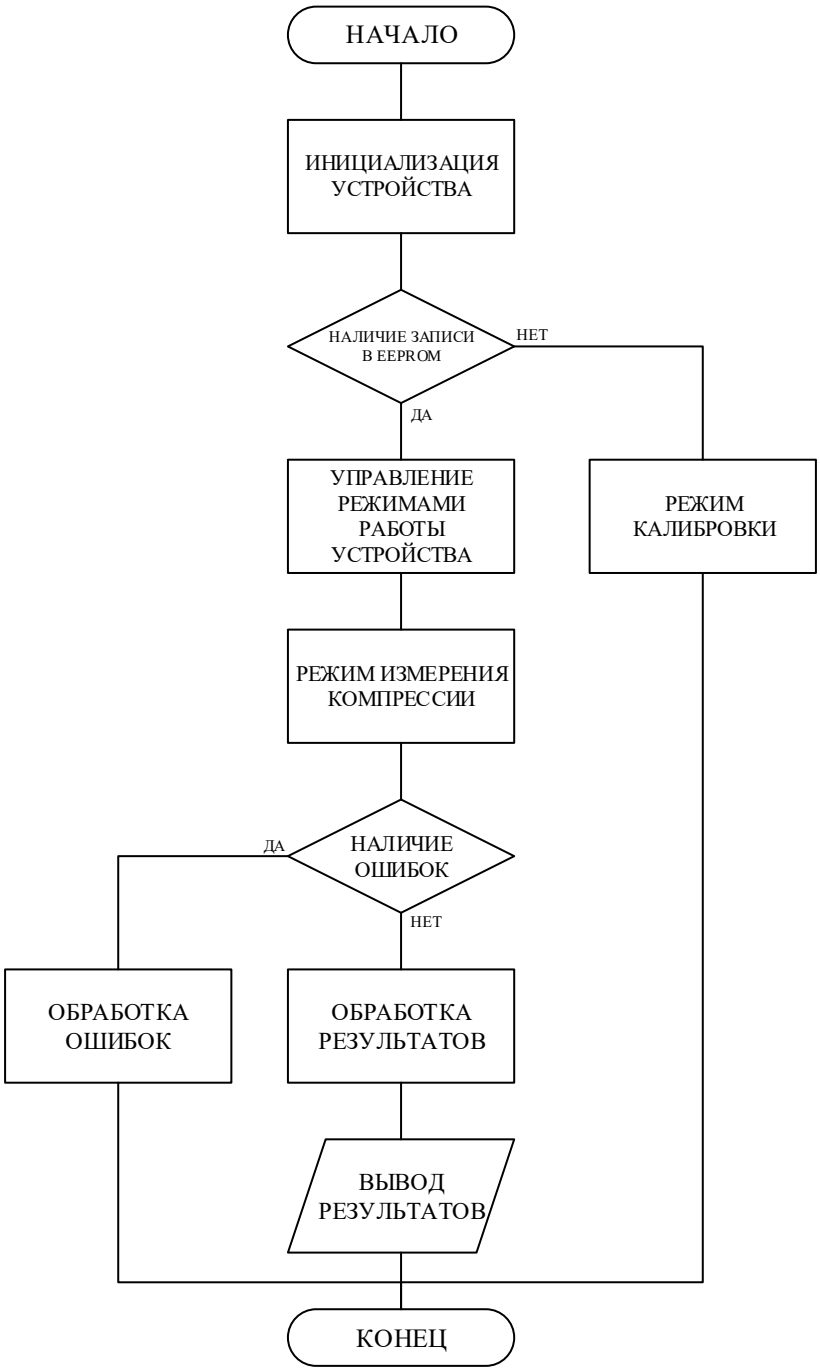

Рисунок 3.1 – Общий алгоритм работы устройства

Для корректного отображения результатов измерений необходимо выполнить установку начальных значений, приняв атмосферное значение за нулевое и предварительно проинициализировав датчик (Рисунок 3.2):

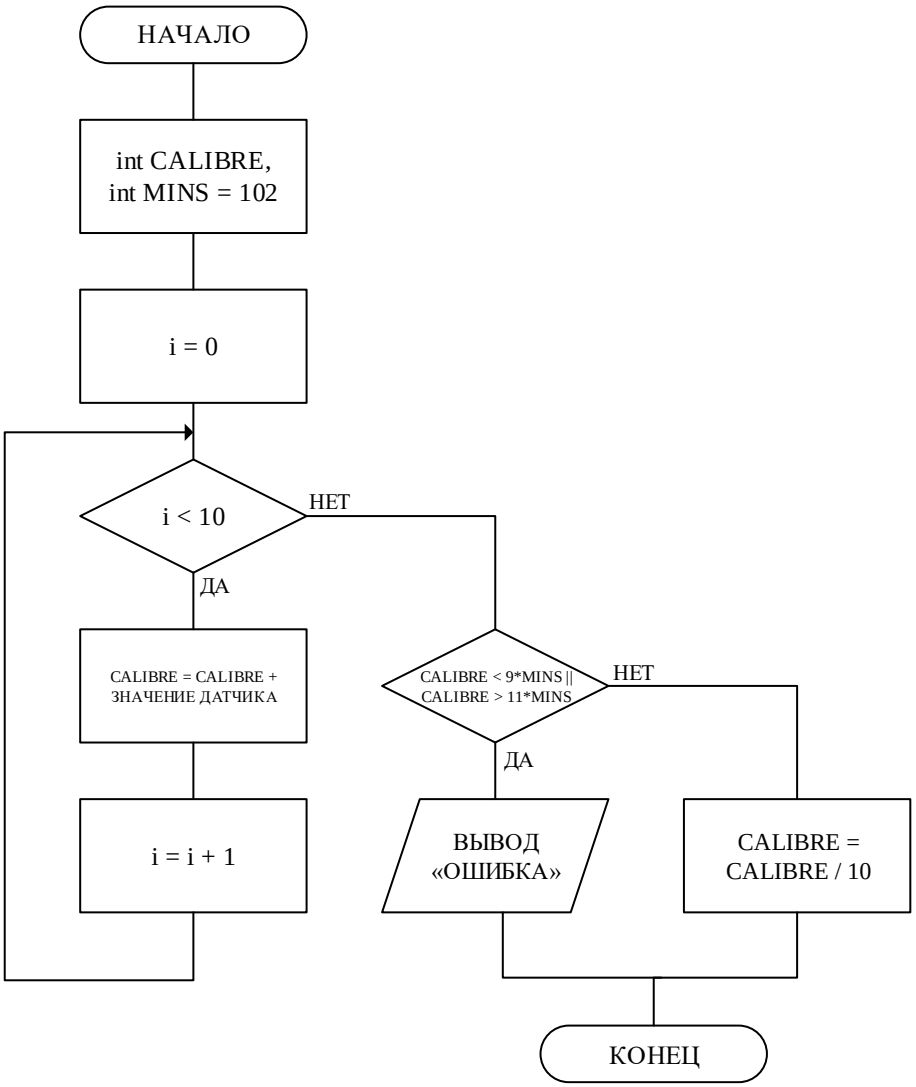

Рисунок 3.2 – Инициализация датчика и установка начальных значений

Значение переменной MINS задается константой и равно нулевому значению датчика. При использовании выбранного датчика данное значение равно 102. При обработке результатов значение переменной CALIBRE будет использоваться как поправочный коэффициент на начальное значение измерений. Численно значение переменной задается средним арифметическим атмосферным давлением.

$$v = \frac{\sum_{i=1}^{10} a_i}{10}$$

Функция получения данных с датчика реализована в виде цикла, выход из которого заканчивается при окончании времени замера. В общем виде она представлена на рисунке 3.3. В блок-схеме используются следующие значения:

1. Массив DIGICOMPR[3] служит для хранения результатов измерений
2. Указатель на массив CH указывает номер ячейки массива
3. Переменная TRESHLOW хранит числовое значение порога падения давления
4. Переменная TRESHHIGH хранит числовое значение порога роста давления
5. Переменная UP является флагом движения к верхней мертвой точке

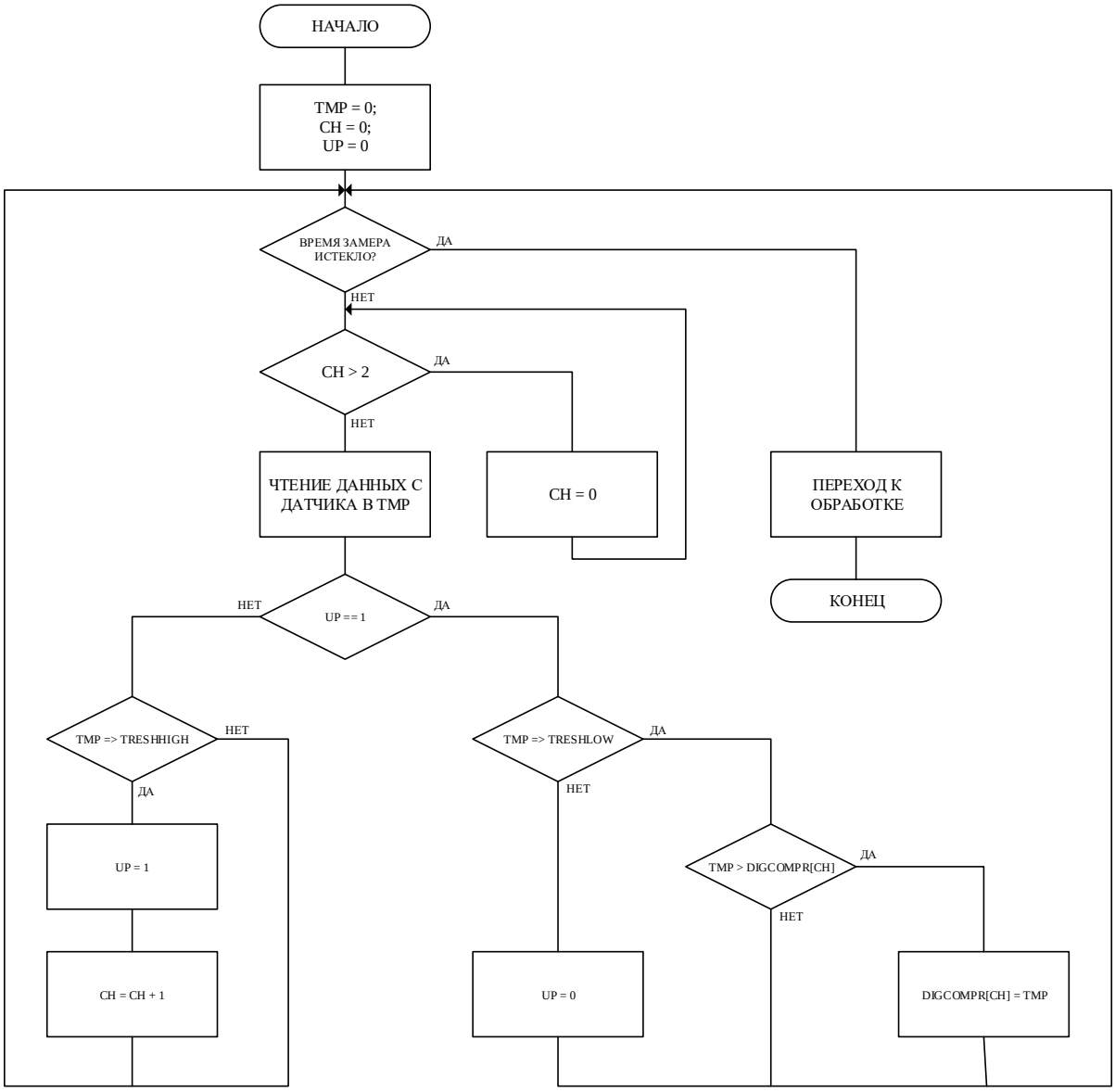

Рисунок 3.3 – Алгоритм получения данных с датчика

После завершения работы таймера и выхода из цикла в памяти хранится массив из трёх элементов. Каждый элемент является максимальным значением компрессии соответствующей камеры сгорания. Эти значения являются «сырыми», то есть требуют дополнительной обработки в виде перевода в принятые единицы измерения компрессии (килограмм-сила на сантиметр квадратный), а также приведения в соответствии с оборотами двигателя в момент измерения.

4 Обработка результатов измерений

Роторно-поршневой двигатель Mazda 13B-MSP на данный момент самый распространенный среди имеющихся роторно-поршневых двигателей, поэтому следует использовать номинальные значения, приведенные производителем в руководстве по обслуживанию двигателя: номинальное значение компрессии 8,5 кгс/см^2 и значение оборотов двигателя 250 об/мин в момент вращения двигателя стартером.

Для обработки аналогового сигнала используется встроенная функция analogRead(). Эта функция считывает значение с указанного аналогового входа. Напряжение, поданное на аналоговый вход, от 0 до 5 Вольт будет преобразовано в значение от 0 до 1023, это 1024 шага с разрешением 0.0049 Вольт. Считывание значения с аналогового входа занимает примерно 100 микросекунд (0.0001 сек), т.е. максимальная частота считывания приблизительно 10,000 раз в секунду. Фактическая частота считывания будет ниже из-за необходимости записи полученных данных в память устройства.

На рисунке 4.1 приведен график сигнала, полученного с датчика давления:

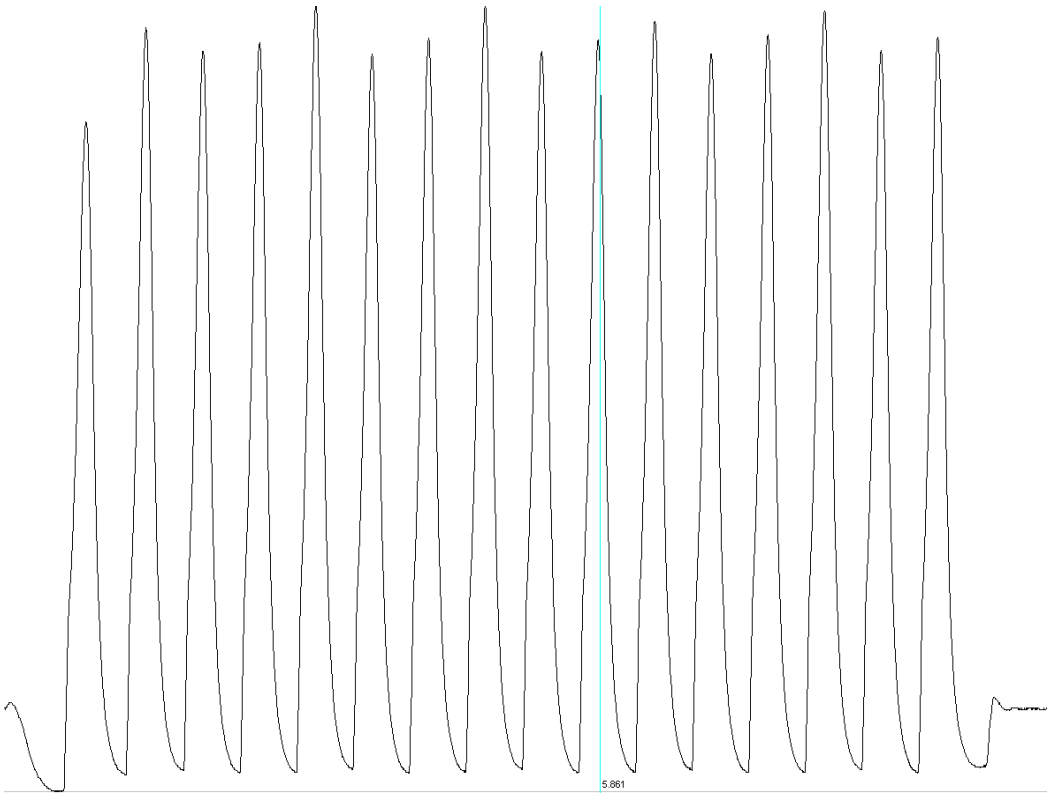

Рисунок 4.1 – Аналоговый сигнал, полученный с датчика давления

Для того, чтобы узнать фактическое время, затраченное для получения значений компрессии в двух соседних камерах сгорания, был составлен точечный график полученных с устройства значений (Рисунок 4.2):

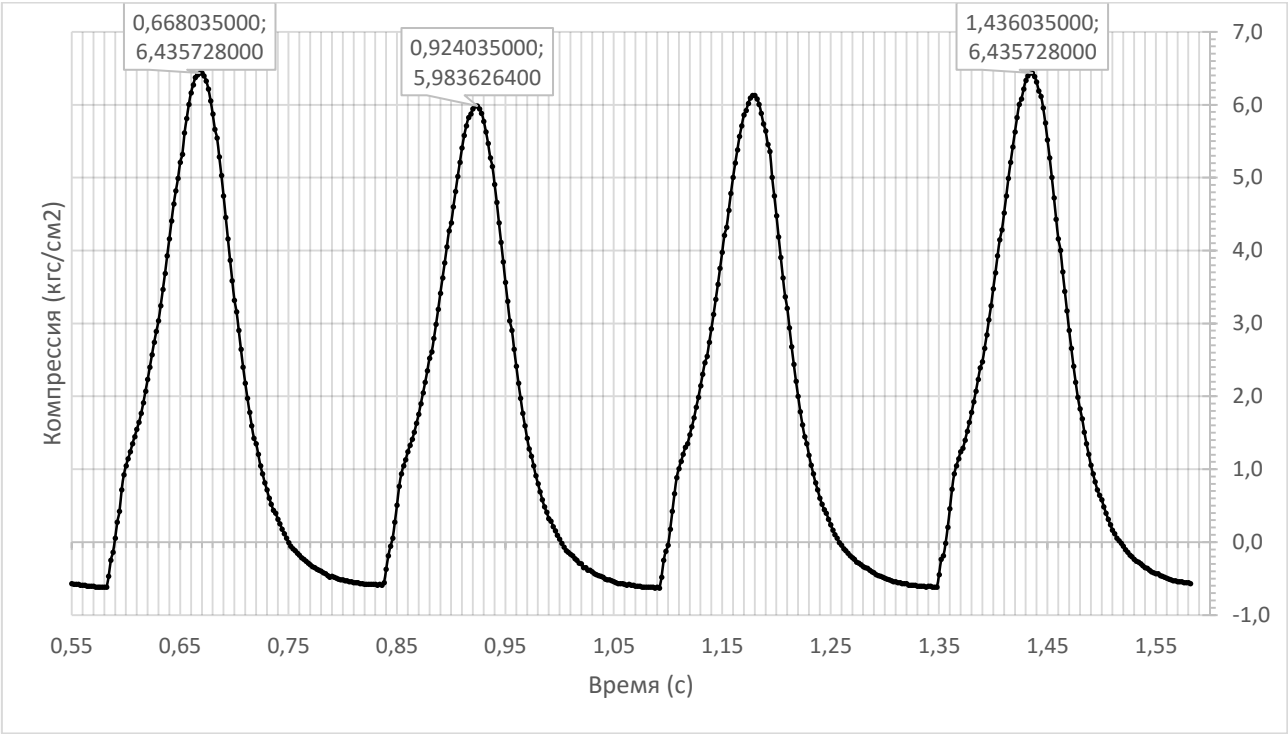

Рисунок 4.2 – Точечный график полученных значений

В данном графике взяты значения одного полного оборота двигателя. Вершины трех пиковых значений подписаны, первое число отображает время в секундах, второе число отображает значение компрессии. Время, которое двигатель тратит на совершение одного оборота – 1,44-0,67=0,77 сек (770 мс). Время перехода от одной камеры сгорания к другой – 0,92-0,68=0,25 сек (250 мс).

Тактовая частота цикла, снимающего показания с датчика давления, задана в 2 мс. Таким образом, частота опроса датчика в 125 раз больше, чем время, за которое камера сгорания завершает свой цикл. На измерение компрессии в одной камере сгорания приходится более 100 значений, что гарантирует необходимую точность для объективной оценки.

Значения номинальной компрессии актуальны только в том случае, если замер компрессии производится при вращении двигателя 250 об/мин, так как на

уплотнительные элементы ротора действует центробежная сила, способствующая увеличению прижимной силы к поверхности статора. Стартер двигателя имеет свойство терять свою мощность под воздействием износа деталей, что обуславливает необходимость дополнительной обработки значений приведением их к оборотам двигателя, равным 250 об/мин.

Вычислить обороты, при которых производился замер компрессии, возможно вычислив время, за которое двигатель совершил один полный оборот (Рисунок 4.3):

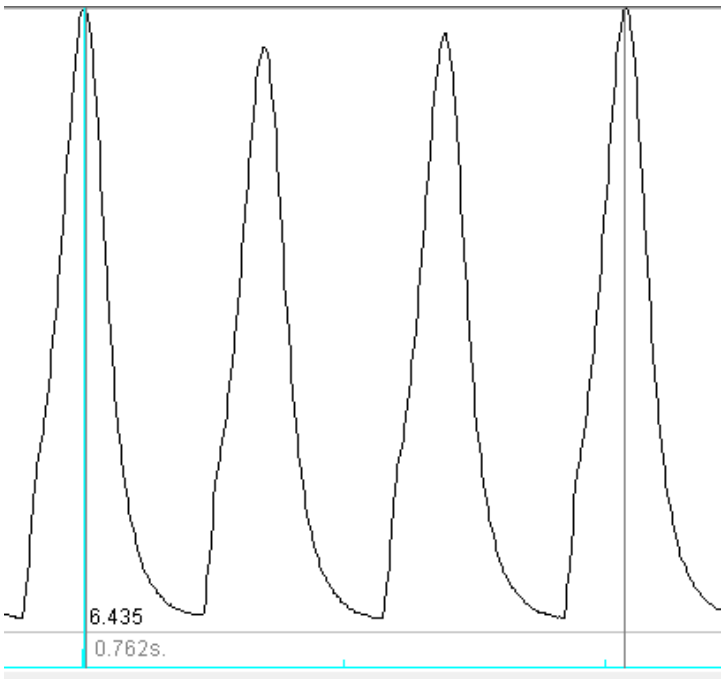

Рисунок 4.3 – Время вращения двигателя на 1 оборот

На графике видно, что при этом замере один оборот двигатель совершает за $t_1 = 0.762$ секунды. Для подсчета оборотов используется формула:

$$RPM = 180 / t_1$$

В приведенном замере двигатель вращается со скоростью 236 об/мин, что не соответствует номинальному значению. Для приведения значений к номинальной скорости вращения двигателя введем переменную CORRVAL = 1000 +

(CORRPM - RPM) * 3, где CORRPM – это эталонное число оборотов двигателя, RPM – измеренное число оборотов двигателя.

Перевод измеренных значений компрессии в единицы килограмм-сила на сантиметр квадратный осуществляется с помощью функции map(). Данная функция пропорционально переносит значение (value) из текущего диапазона значений (fromLow .. fromHigh) в новый диапазон (toLow .. toHigh), заданный параметрами. Функция выглядит следующим образом:

```
KGSCOMPR[CH] = map((long)DIGCOMPR[CH] * CORRVAL / 1000, CALIBRE, MAXS, 0, MAX_SENS);
```

Где DIGICOMPR[CH] – это измеренная компрессия, CORRVAL – это коэффициент по оборотам, CALIBRE – это измеренное атмосферное значение, MAXS – это показания датчика на аналоговом входе при максимально возможном давлении, MAX_SENS – это максимальное показание датчика давления в кгс/см^2.

5 Программирование устройства

Для написания программного кода применимы 2 наиболее распространенных языка C++ и Assembler

Assembler – Традиционно зарекомендовавший себя язык программирования. Написание приложений на данном языке сводится к написанию машинного кода: полноценная работа с ячейками памяти – адресация, чтение и запись, выполнение команд, которые сводятся к выполнению логических операций, выводу информации, работа с прерываниями. Написание программы на данном языке программирования весьма трудоемкое занятие и требует определенных знаний не только в области программирования, но и архитектуре ЭВМ.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений. Еще одной ключевой особенностью данного языка является универсальность и кроссплатформеность с возможностью быстрой адаптацией под нужный микроконтроллер.

Для написания приложения для платформы Arduino используется совместимый с C++ язык программирования. Микроконтроллер на плате программируется при помощи языка Arduino (основан на языке Wiring) и среды разработки Arduino (основана на среде Processing). Проекты устройств, основанные на Arduino, могут работать самостоятельно, либо же взаимодействовать с программным обеспечением на компьютере (напр.: Flash, Processing, MaxMSP).

Текст программы представлен в приложении А.

Для разработки приложений для встраиваемых систем Arduino Uno используется оригинальная среда разработки Arduino IDE (Рисунок 5.1). Данная среда разработки является свободно распространяемой и доступна для скачивания в сети Internet на оригинальном сайте Arduino.cc.

Рисунок 5.1 – Среда разработки Arduino IDE

Для того чтобы начать работу с кодом, необходимо создать рабочее пространство – проект (Рисунок 5.2)

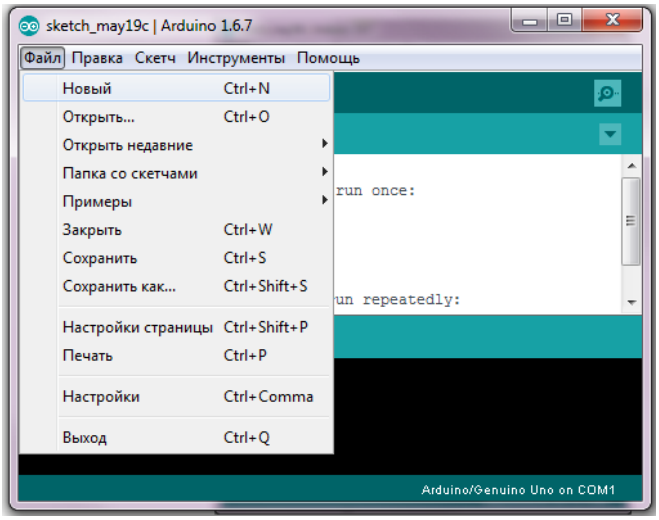

Рисунок 5.2 – Создание рабочего проекта Arduino IDE

Далее создается рабочий файл, который непосредственно является набором библиотек и исполняемых фалов, которые можно будет собрать в готовый проект и скомпилировать.

Следующим этапом является выбор типа устройства, для которого проводится разработка (Рисунок 5.3). В нашем случае это Arduino Uno, однако расширение функционала возможно и на такие устройства как Arduino Nano, Arduino FIO и другие интеллектуальные устройства с поддержкой платформы Arduino.

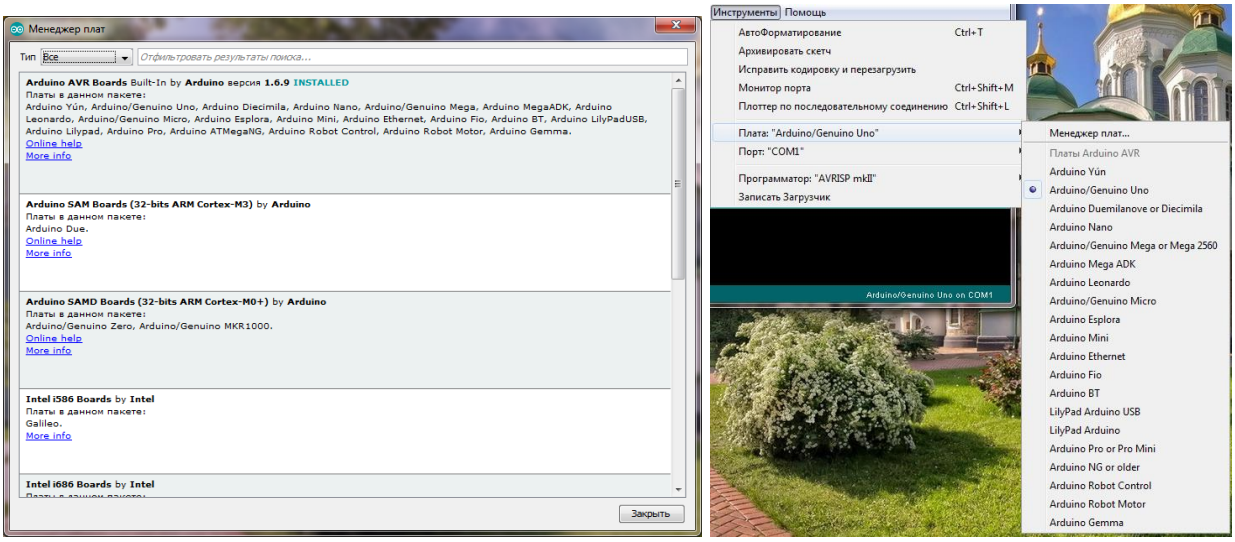

Рисунок 5.3 – Выбор устройства Arduino IDE

Конечным этапом настройки проекта является подключение пользовательских библиотек (Рисунок 5.4).

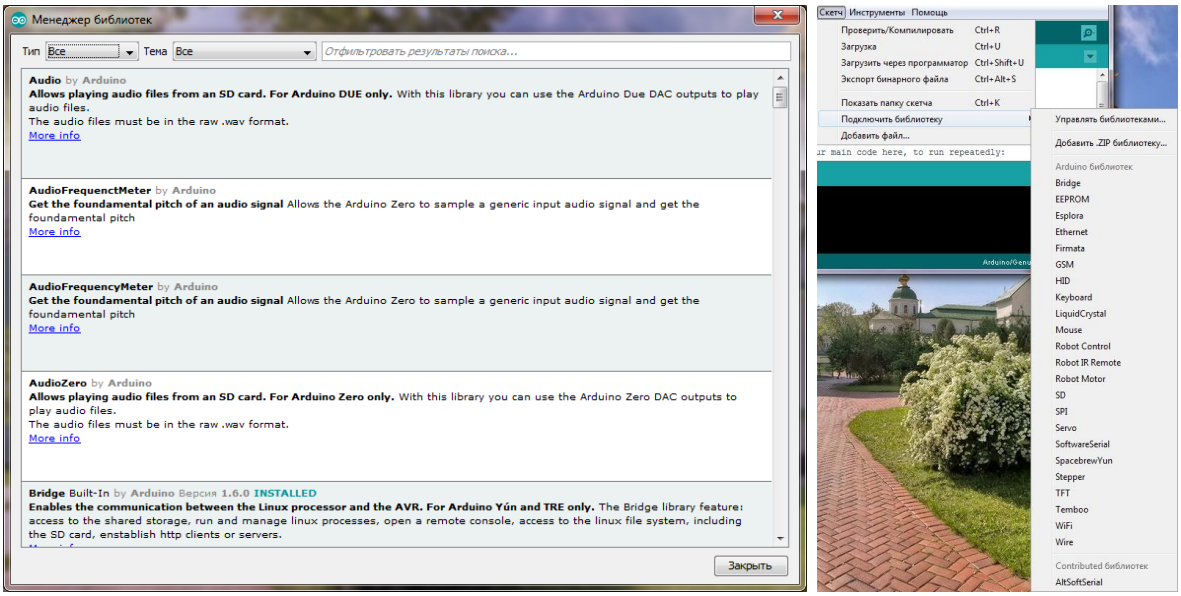

Рисунок 5.4 – Подключение пользовательских библиотек

Arduino IDE располагает встроенным редактором поставляемым с большинством необходимых функций, таких как визуальное выделение дескрипторов и автоматическое форматирование (Рисунок 5.5). Ошибки, появляющиеся в процессе работы, отображаются в нижнем выходном окне. Операции перехода по тегу ошибки предоставляет более подробную информацию об ошибках.

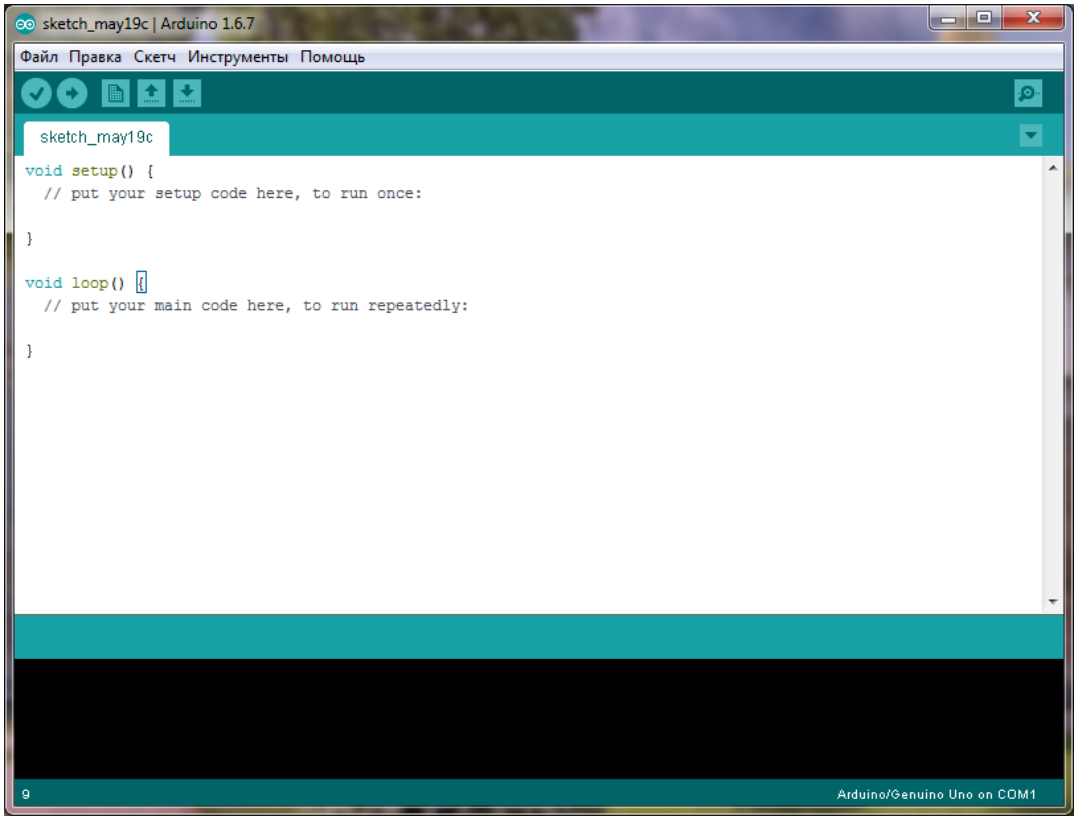

Рисунок 5.5 – Текстовый редактор проекта Arduino IDE

В то же время, при желании, разработчик может использовать не встроенный, а вызвать привычный для него редактор, типа Notepad, NetBeans, Eclipse и ряд других.

Arduino IDE позволяет через диалоговые окна установки запускать и конфигурировать компилятор и средства компоновки. Более того, непосредственно в Arduino IDE встроены три отладчика, используемые на различных этапах разработки (Рисунок 5.6). Для каждой конкретной ситуации может быть выбрано соответствующее окружение отладки.

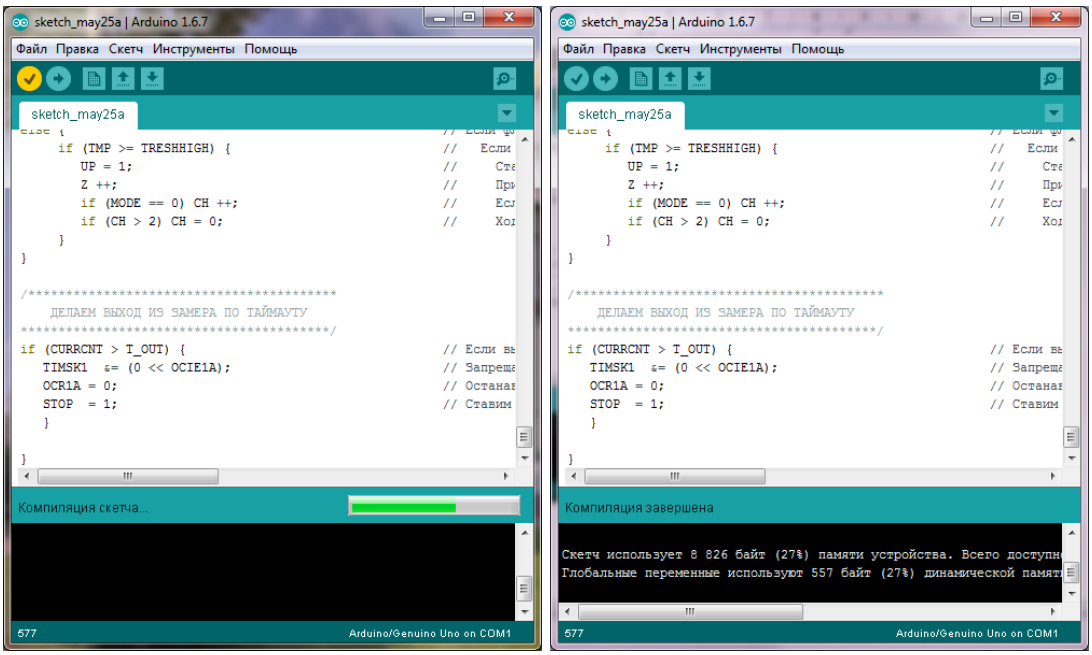

Рисунок 5.6 – Компиляция проекта Arduino IDE

Загрузка прошивки является конечным этапом разработки устройства, после чего проводится его настройка и тестирование.

6 Пользовательский интерфейс

В связи с тем, что устройство используется людьми, не владеющими знаниями в области программирования и электроники, важной задачей является создание простейшего интерфейса с интуитивно понятными действиями.

Включение устройства осуществляется при помощи тумблера, установленного на корпусе устройства.

Основной задачей устройства является измерение компрессии в роторно-поршневом двигателе. В двигателях такого типа максимально возможная компрессия колеблется в пределах 9,0 – 9,5 Атм. Для сведения погрешности измерения к минимуму предлагается использовать датчик давления с предельным значением в 10 Атм. Однако устройство предусматривает возможность измерения компрессии в классических поршневых двигателях, где максимальные значения компрессии выше. Следовательно, необходимо было предусмотреть возможность выбора нужного датчика.

После включения устройства пользователю предлагается выбрать один из двух датчиков: на 10,5 Атм или на 14 Атм (Рисунок 6.1). Выбор осуществляется при помощи клавиш ВВЕРХ и ВНИЗ.

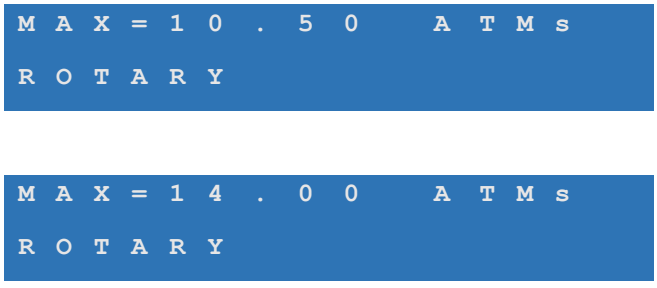

Рисунок 6.1 – Выбор датчика давления

В момент выбора датчика пользователю предлагается также указать один из трех типов измерения: для роторно-поршневого двигателя, для поршневого четырехтактного или для поршневого двухтактного (Рисунок 6.2).

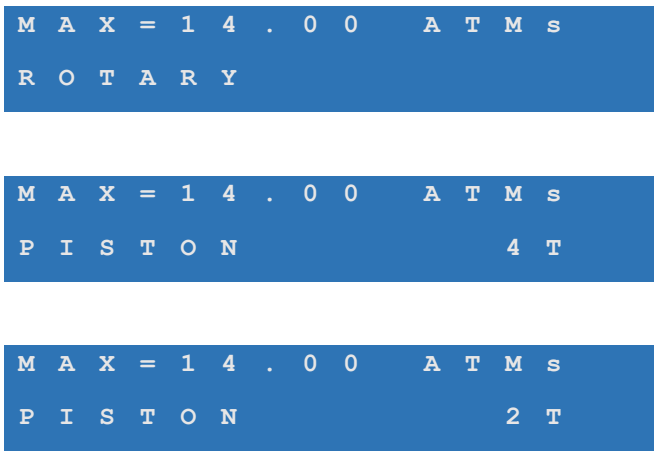

Рисунок 6.2 – Выбор типа измерения

Выбор осуществляется при помощи клавиш ВПРАВО и ВЛЕВО, подтверждается выбор при помощи клавиши ВЫБОР.

После этого, устройство готово к замеру. Датчик давления должен быть вкручен в отверстие свечи зажигания измеряемого цилиндра. Измерение начнется после нажатия на клавишу ВЫБОР. В это же время нужно начать вращать двигатель стартером до тех пор, пока значения на дисплее устройства не перестанут меняться, то есть будет достигнута максимальная компрессия для данного двигателя. Время замера ограничено временным интервалом в 30 секунд.

После окончания замера отображается информация о компрессии и оборотах двигателя в момент замера (Рисунок 6.3)

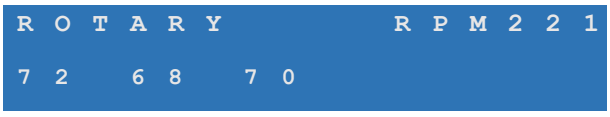

Рисунок 6.3 – Результаты измерения компрессии роторно-поршневого ДВС

В связи с тем, что заводские значения компрессии указаны при замере с частотой вращения двигателя 250 об/мин, имеется возможность автоматически пересчитать измеренные значения с поправкой на 250 об/мин. Данная функция осуществляется только в режиме измерения для роторно-поршневого двигателя при нажатии клавиши ВВЕРХ (Рисунок 6.4)

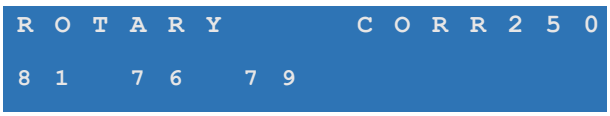

Рисунок 6.4 – Приведенные значения к частоте вращения 250 об/мин

В режиме замера компрессии для классических поршневых двигателей в момент замера будет отображаться текущее значение компрессии, а функция приведения оборотов после окончания замера недоступна ввиду отсутствия необходимости.

Предусмотрены оповещения пользователя об ошибках, связанных с отсутствием в системе датчика давления или слишком низких оборотах в момент замера. Таким образом, при отсутствии или неисправности датчика давления пользователь увидит ошибку SENSOR ERROR (Рисунок 6.5).

Рисунок 6.5 – Индикация ошибки датчика

Для повторного замера необходимо нажать клавишу СБРОС – устройство будет перезапущено.

На рисунках 6.6 и 6.7 отображен полный комплект прибора для замера компрессии в роторно-поршневом двигателе.

Рисунок 6.6 – Полный комплект измерительного прибора

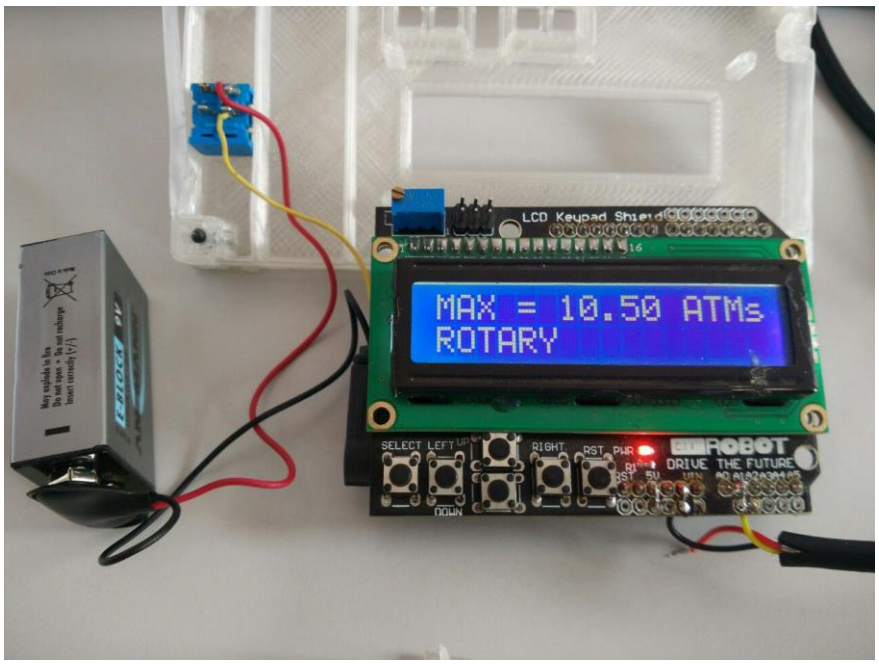

Рисунок 6.7 – Комплект измерительного прибора в разобранном состоянии

Датчик давления, адаптированный под свечное отверстие, представлен на Рисунке 6.8:

Рисунок 6.8 – Датчик измерительного прибора

Данное устройство является оконченным проектом, дальнейшая перспектива развития возможна лишь при его технической реализации и прямой отладки

всевозможных ошибок, возникших на этапе разработки. Дополнительным развитием возможно применение более гибкого программного обеспечения, однако эти все задачи выходят за рамки одной работы.

7 Экономическое обоснование проекта

Спецификация электронных компонентов и смета затрат на его приобретение и монтаж представлены в табличной форме.

Затраты на монтаж и сборку компонентов принимаются на уровне 8 – 15% от его стоимости.

Согласно законодательству, конечная цена продукта должна стоить не менее 50% его конечной стоимости. Учитывая цены конкурентов, конкурентоспособная цена на устройство равна 10000 руб. из которых 18% - это НДС. Следовательно, чистая прибыль с продажи одного устройства составит:

$$П = Ц - Ц \cdot НДС - Цу = 10000\text{-}1800\text{-}5198\text{=}3002(\text{руб})$$

Таблица 7.1 – Расчет стоимости комплектующих

Наименование	Кол-во (шт)	Цена за 1 ед (руб)	Итого (руб)
Arduino Uno	1	1450	1450
LCD keypad shield 1602	1	1840	1840
Pressure sensor 150 psi	1	680	680
Корпус	1	250	250
Проводники	1	200	200
Батарея типа «крона»	1	100	100
Итого			4520.00
Затраты на монтаж			678
Полная стоимость			5198.00

Так как на начальном этапе, потребуются значительные затраты на рекламу устройства, сертификация и прочие затраты, то потребуются дополнительные вложения, которые постепенно можно снизить.

ЗАКЛЮЧЕНИЕ

В данной работе разработана система измерения компрессии роторного двигателя при помощи датчика давления с микропроцессорным модулем.

В ходе выполнения работы был решен ряд задач, среди которых в первую очередь необходимо выделить следующие:

– анализ современных средств измерения компрессии;

– анализ современных компонентов и датчиков;

– анализ работы роторного поршневого агрегата;

– формирование задач для системы;

– выбор средств разработки системы;

– разработка программного обеспечения системы;

– проектирование структурных схем функционирования системы;

– оценка эффективности предлагаемых решений.

В результате анализа были сформированы основные функции, которые должны быть реализованы в современной интеллектуальной измерительной системе. Данное решение позволяет перейти непосредственно к физической реализации системы.

Разработанное устройство представляет собой полностью готовое изделие, выполняющее поставленные задачи. Кроме того, управление устройством не требует специальных знаний и навыков, интерфейс интуитивно понятен любому пользователю.

Для быстрой и полной адаптации с другими системами использованы универсальные кроссплатформенные средства разработки. Программное обеспечение содержит удобную архитектуру.

В заключительном разделе работы проведены оценки погрешности и натурные эксперименты.

Таким образом, в работе проведено исследование предметной области, разработана структура, алгоритмическое и программное обеспечение, спроектирована модель и проведены реальные эксперименты.

Созданная система соответствует актуальным техническим требованиям и сможет поддерживать это соответствие в течение всего жизненного цикла системы.

Предварительные расчеты показали экономическую эффективность данного проекта. Учитывая его себестоимость и сроки окупаемости можно сделать вывод, что изготовление устройства способствует получении прибыли не только за счет его продаж, но и за дополнительные услуги, которые уменьшают объем работы за счет уникальности устройства.

Численные показатели:

Полные затраты на изготовление системы составляют 5198 руб.

Ориентировочная цена на продажу 10 000 руб из которых чистая прибыль равна 3002руб.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Левитин, А.В. Алгоритмы: введение в разработку и анализ Introduction to The Design and Analysis of Algorithms. / А.В. Левитин. – Москва: «Вильямс», 2006. — С. 240-242
2. Дейт, К. Дж. Введение в системы баз данных – 6-е изд. Переработанное и дополненное / К. Дж. Дейт. – Москва: «Вильямс», 2003 – 848 с.
3. Хомоненко, А.Д. Основы современных компьютерных технологий: Учебное пособие / А.Д. Хомоненко – Санкт-Петербург: КОРОНА–Принт, 2003 – 448 с.
4. Arduino Uno [Электронный ресурс] // Amperka – Режим доступа: http://wiki.amperka.ru/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D1%8B:arduino-uno
5. Arduino Library Examples [Электронный ресурс] // Arduino – Режим доступа: https://www.arduino.cc/en/Tutorial/LibraryExamples
6. Программирование Arduino [Электронный ресурс] // Arduino – Режим доступа: http://arduino.ru/Reference
7. РПД изнутри и снаружи [Электронный ресурс] // И. Пятов – Режим доступа: http://engine.aviaport.ru/issues/11&12/page14.html
8. Изгнание дьявола из мотора Ванкеля [Электронный ресурс] // Вокруг Света – Режим доступа: http://www.vokrugsveta.ru/telegraph/technics/941/
9. Engine Workshop Manual 13B-MSP (Multi Side Port). – Hiroshima, Mazda Motor Corporation, 2008.

ПРИЛОЖЕНИЕ А

Листинг программы

```
char ver[6] = "v0.14";

int MAX_SENS1   = 105;
int MAX_SENS2   = 140;
byte TRESHOLD1  = 20;
byte TRESHOLD2  = 30;

int MINS        = 102;
int MAXS        = 922;
byte MODE       = 0;
int CORRPM      = 250;
int MINRPM      = 170;
byte Z_FIRST    = 6;
byte Z_LAST     = 9;
byte TIMER      = 2;
byte TIMEOUT    = 30;
#define SENSPIN   1
#define btnCAL    3

int  CALIBRE,
    TRESHLOW, TRESHHIGH,
    KGSCOMPR[3],
    RPM,
    CORRVAL,
    bUP, bDN, bLT, bRT, bSL;

byte i,
    ERRORCODE,
    BTN[6];

boolean START, CORR, RPMCOUNT,
       ERR, OUT,
       MAX[3];

volatile int DIGCOMPR[3],
           CURRCNT, Z_TMP, CNT[16];

volatile byte CH, Z;


volatile boolean STOP, UP;
```

```
int MAX_SENS        = MAX_SENS1;
unsigned int T_OUT  = TIMEOUT / TIMER * 1000;
unsigned int FREQ   = 2000 * TIMER;
unsigned int MAXCNT = 60000 / TIMER / MINRPM;
long DIVRPM         = 60000 / TIMER * (Z_LAST - Z_FIRST);

#include <EEPROM.h>

#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
byte lcd_key    = 0;
int adc_key_in  = 0;
#define btnNONE   0
#define btnSELECT 1
#define btnLEFT   2
#define btnUP     3
#define btnDOWN   4
#define btnRIGHT  5

int read_LCD_buttons()
{
adc_key_in = analogRead(0);
if (adc_key_in > bSL +10 )  return btnNONE;
if (adc_key_in < bRT)  return btnRIGHT;
if (adc_key_in < bUP)  return btnUP;
if (adc_key_in < bDN)  return btnDOWN;
if (adc_key_in < bLT)  return btnLEFT;
if (adc_key_in < bSL)  return btnSELECT;
return btnNONE;
}

void setup() {
  lcd.begin(16, 2);
  ERR = 1;

  lcd.setCursor(0,0);
  lcd.print(" ROTARY C-METER ");
  lcd.setCursor(0,1);
  lcd.print("     ");
  lcd.setCursor(0,1);
  lcd.print(ver);
  delay (3000);

  TCCR1A  = 0; TIMSK1 = 0;
```

```
  TCCR1B  = 0;
  TCNT1   = 0;
  OCR1A   = 0;
  TCCR1B  |= ((1 << CS11) | (1 << WGM12));

pinMode(btnCAL, INPUT);
digitalWrite(btnCAL, HIGH);

if (digitalRead(btnCAL) == 0)
  EEPROM.write(0, 0);

BTN[0] = EEPROM.read(0);
if (BTN[0] != 1) {
  ERRORCODE = 5;
  OUT = 1;
  CURRCNT = analogRead(0);
}
else {
  bSL = int(EEPROM.read(1) * 4);
  bLT = int(EEPROM.read(2) * 4);
  bUP = int(EEPROM.read(3) * 4);
  bDN = int(EEPROM.read(4) * 4);
  bRT = int(EEPROM.read(5) * 4);

}
}

void loop() {

if (ERRORCODE > 0){
  TIMSK1  &= (0 << OCIE1A);
  OCR1A = 0;

  if (ERRORCODE == 5) CALIBER();

  if (ERRORCODE == 3) {
  lcd.setCursor(0,0);
  lcd.print("LOW RPM!");
  lcd.setCursor(0,1);
  lcd.print("OR CHAMBER  DOWN");
  ERRORCODE ++;
  }

  if (ERRORCODE == 1){
  lcd.setCursor(0,0);
```

```
  lcd.print(" SENSOR  ERROR! ");
  lcd.setCursor(0,1);
  if (ERR == 0)
       lcd.print(" VALUE TOO LOW  ");
  else  lcd.print(" VALUE TOO HIGH ");
  ERRORCODE ++;
  }
  return;
}


if (START - STOP == 0 ) {
lcd_key = read_LCD_buttons();
if (lcd_key > 0) BUTTON();

}


if (START == 0 && OUT == 0) {
  lcd.setCursor(0,0);
  lcd.print("MAX =       ATMs");
  lcd.setCursor(6,0);
  lcd.print((float)MAX_SENS / 10.0);
  if (MODE == 0) {
    lcd.setCursor(0,1);
    lcd.print("ROTARY          ");
  }
  else {
    lcd.setCursor(0,1);
    lcd.print("PISTON        T");
    lcd.setCursor(14,1);
    if (MODE == 1) lcd.print("4");
    else          lcd.print("2");
  }
  OUT  = 1;
  return;
}

if (START == 1 && STOP == 0) {
  lcd.begin(16,2);
  lcd.print("WAIT     SECONDS");
  lcd.setCursor(5,0);
  lcd.print((T_OUT - CURRCNT) / (1000 / TIMER));
  lcd.setCursor(0,1);
  lcd.print(DIGCOMPR[0]);
```

```
  if (MODE == 0) {
    lcd.setCursor(6,1);
    lcd.print(DIGCOMPR[1]);
    lcd.setCursor(12,1);
    lcd.print(DIGCOMPR[2]);
   }
  delay(1000);
  return;
}

if (STOP == 1 && OUT == 0) {

  lcd.begin(16,2);

  if (RPMCOUNT == 0) {
     for (Z_TMP = 0; Z_TMP <= Z_LAST - Z_FIRST; Z_TMP ++) {
     if (CNT[Z_TMP+1] - CNT[Z_TMP] > MAXCNT && MODE == 0)
     ERRORCODE = 3;
     }
     RPM = DIVRPM / (CNT[Z_LAST - Z_FIRST] - CNT[0]);
     if (MODE == 1) RPM *= 2;
     RPMCOUNT = 1;
  }


  if (MODE == 0 && CORR == 1) {
    CORRVAL = 1000 + (CORRPM - RPM) * 3;
    lcd.setCursor(9,0);
    lcd.print("CORR");
    lcd.setCursor(13,0);
    lcd.print(CORRPM);
  }
  else {
    CORRVAL = 1000;
    lcd.setCursor(9,0);
    lcd.print("RPM=   ");
    lcd.setCursor(13,0);
    lcd.print(RPM);
  }


  for (CH = 0; CH <= 2; CH ++) {
    if (DIGCOMPR[CH] > MAXS + 3)
      MAX[CH] = 1;
```

```
            KGSCOMPR[CH] = map((long)DIGCOMPR[CH] * CORRVAL / 1000,
CALIBRE, MAXS, 0, MAX_SENS);
           }

          if (MODE == 0){
           lcd.setCursor(0,0);
           lcd.print("ROTARY");
           for (CH = 0; CH <= 2; CH ++){
            lcd.setCursor(CH * 6,1);
            if (MAX[CH] == 1 && CORR == 0)
             lcd.print("MAX!");
            else lcd.print((float)KGSCOMPR[CH] / 10.0);
           }
          }
          else {
           lcd.setCursor(0,0);
           lcd.print("PISTON");
           lcd.setCursor(0,1);
           if (MAX[0] == 1) {
            lcd.print("MAX! >");
            lcd.print((float)KGSCOMPR[0] / 10.0);
           }
           else lcd.print((float)KGSCOMPR[0] / 10.0);
          }
          OUT = 1;
         }

         }

         void BUTTON(){
          switch (lcd_key)
          {
           case btnRIGHT:
            {
             OUT  = 0;
             if (START == 0) {MODE ++;
              MODE = min(MODE, 2);
             }
            break;
            }
           case btnLEFT:
            {
             OUT  = 0;
             if (START == 0) {MODE --;
              if (MODE != 1) MODE = 0;
```

```
   }
  break;
  }
 case btnUP:
  {
   OUT  = 0;
   if (START == 0) MAX_SENS = MAX_SENS2;
   if (STOP == 1 && MODE == 0) CORR = 1;
  break;
  }
 case btnDOWN:
  {
   OUT  = 0;
   if (START == 0) MAX_SENS = MAX_SENS1;
   if (STOP == 1 && MODE == 0) CORR = 0;
  break;
  }
 case btnSELECT:
  {
   if (START == 0){
       START = 1;
       OUT  = 0;
       lcd.clear();
       for (i=0; i < 10; i++) {
          CALIBRE += analogRead(SENSPIN);
         delay(50);
       }
       if (CALIBRE > 11 * MINS ||
          CALIBRE < 9 * MINS) {
          ERRORCODE = 1;
          if (CALIBRE < 9 * MINS)
          ERR = 0;
         }
       }
   if (ERRORCODE == 0) {
      CALIBRE = analogRead(SENSPIN);
      TRESHLOW =  (long)CALIBRE * (100 + TRESHOLD1) / 100;
      TRESHHIGH = (long)CALIBRE * (100 + TRESHOLD2) / 100;
      delay(500);
      OCR1A = FREQ;
      TIMSK1  |= (1 << OCIE1A);
   }
  break;
  }
}
```

```
  delay(500);
}

void CALIBER(){
while (1 == 1){
   if (digitalRead(btnCAL) == 0) {
     lcd.setCursor(0,0);
     lcd.print("RELEASE  CALIBER");
     lcd.setCursor(0,1);
     lcd.print("    BUTTON    ");
     if (i <= 4 && OUT == 0) {
       i++;
       BTN[i] = analogRead(0) / 4 + 5;
       OUT = 1;
      }
      if (i > 4 && BTN[0] != 1) {
        BTN[0] = 1;
        for (CH = 0; CH <= 5; CH ++)
          EEPROM.write(CH, BTN[CH]);
      }
   }
   else {
   delay(100);
    if (i > 4) {
      lcd.setCursor(0,0);
      lcd.print("   REBOOT THE   ");
      lcd.setCursor(0,1);
      lcd.print("    DEVICE    ");
    }
     else {
         lcd.setCursor(0,0);
         lcd.print(" PRESS         ");
         lcd.setCursor(0,1);
         lcd.print("                  ");
         lcd.setCursor(8,0);
         if (i == 0) lcd.print("SELECT");
         if (i == 1) lcd.print("LEFT");
         if (i == 2) lcd.print("UP");
         if (i == 3) lcd.print("DOWN");
         if (i == 4) lcd.print("RIGHT");
         if (analogRead(0) < CURRCNT - 10 ||
           analogRead(0) > CURRCNT + 10) {
           lcd.setCursor(0,1);
           lcd.print(" PRESS  CALIBER ");
         }
```

```
          OUT = 0;
      }

    }
    delay(500);
  }
}


ISR(TIMER1_COMPA_vect) {
CURRCNT ++;

if (Z >= Z_FIRST && Z <= Z_LAST + 1){
    if(CNT[Z - Z_FIRST] == 0) {
     CNT[Z - Z_FIRST] = CURRCNT;
    }
}

int TMP = analogRead(SENSPIN);

if (UP == 1) {
   if (TMP >= TRESHLOW) {
      if (TMP > DIGCOMPR[CH])
         DIGCOMPR[CH] = TMP;
   }
   else UP = 0;
}

else {
     if (TMP >= TRESHHIGH) {
        UP = 1;
        Z ++;
        if (MODE == 0) CH ++;
        if (CH > 2) CH = 0;
     }
}

if (CURRCNT > T_OUT) {
   TIMSK1  &= (0 << OCIE1A);
   OCR1A = 0;
   STOP  = 1;
   }

}
```

ПРИЛОЖЕНИЕ Б

Функциональная схема