Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра «Электротехнические комплексы и системы»

УТВЕРЖДАЮ
Заведующий кафедрой
_____В. И. Пантелеев
«____»_____2018 г.

БАКАЛАВРСКАЯ РАБОТА

13.03.02 - Электроэнергетика и электротехника

ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ВЕНТИЛЬНЫМ ДВИГАТЕЛЕМ

Руководитель _____ к.т.н., доцент А.Н. Пахомов
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ Ю.М.Иванов
подпись, дата инициалы, фамилия

Консультант:

Генеральный директор
ООО «Вертекс» _____ Д.С. Жидков
подпись, дата инициалы, фамилия

Нормоконтролер _____ А.Н. Пахомов
подпись, дата инициалы, фамилия

Красноярск 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современной промышленности работа оборудования и дальнейшее развитие невозможно без автоматизированного электропривода.

С помощью электропривода приводятся в движения практически все механизмы станков, машин, кранов, систем вентиляции и т.д. Без электроприводов невозможно представить современное автоматизированное производство.

На данный момент в промышленности применяются различные виды электродвигателей и каждый из них используется для выполнения определённых задач, а благодаря развитию электротехники, их возможности возрастают.

Объектом моей выпускной квалификационной работы стала машина для офсетной печати Heidelberg speedmaster 52.

Работа проходила при тесном сотрудничестве с фирмой ООО «Вертекс», специализирующееся на техническом обслуживании промышленного оборудования. Фирма владеет современными средствами и технологиями в области автоматизации, что позволяет решать комплексные задачи любой сложности.

Целью бакалаврской работы является изучение и анализ программного обеспечения, используемого для программирования микроконтроллера Arduino, изучения методики программирования и проектирования микропроцессорной системы управления вентильным двигателем с использованием логического микроконтроллера ArduinoMega2560. Основными задачами микроконтроллера являются управление коммутацией двигателя, определение положения ротора с помощью датчика Холла, задание и поддержание необходимой скорости вращения для соблюдения технологического процесса.

Стенд был собран из имеющегося на производстве электрического оборудования.

1 Общее описание технологического процесса

Офсетная печать в современной полиграфии является самым простым и дешёвым способом в типографской продукции. Этикетки, упаковочная продукция, глянцевые журналы и прочие многостраничные издания выполняются именно посредством офсетной печати.

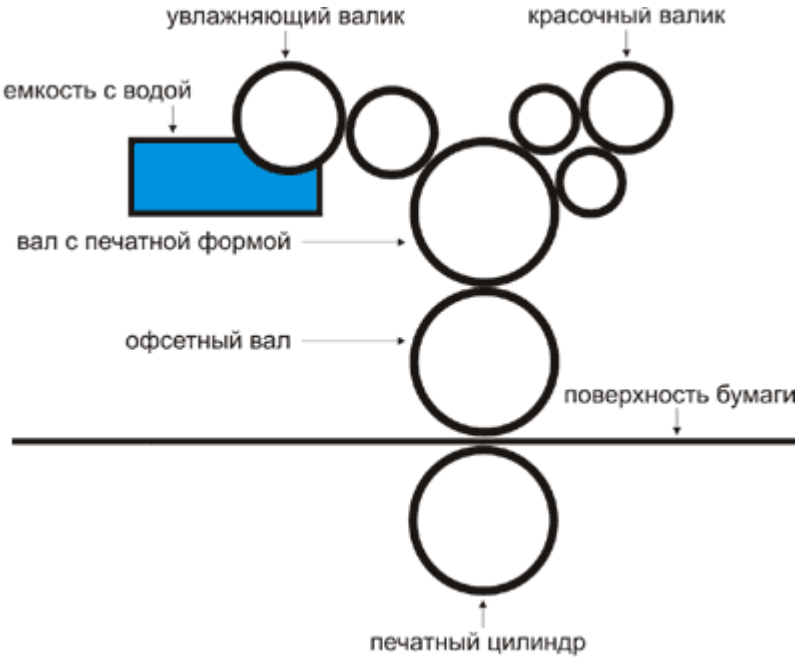


Рисунок 1.1 – Технология процесса офсетной печати

На загрузочную платформу укладывают листы форматом А3, после чего бумага проходит через два вала.

Первый называется вал с формой на который наносится изображение, соединенный с увлажняющим валиком и красящим валиком. После проявки, одни засвеченные части формы отталкивают краску и притягивают воду с увлажняющего валика, вторые наоборот притягивать краску и отталкивать воду.

При каждом повороте вал с печатной формой омывается водой, затем с красящих валиков переносится краска и таким образом формируются буквы и изображения. Затем с вала с формой изображение переносится на офсетный вал, а с него уже на бумагу.

Печатная машина имеет секции для каждой отдельной краски. Бумага поочередно проходит через каждую секцию. В случае замены краски в каждой секции включаются устройства смывки и сушки офсетного и печатного вала.

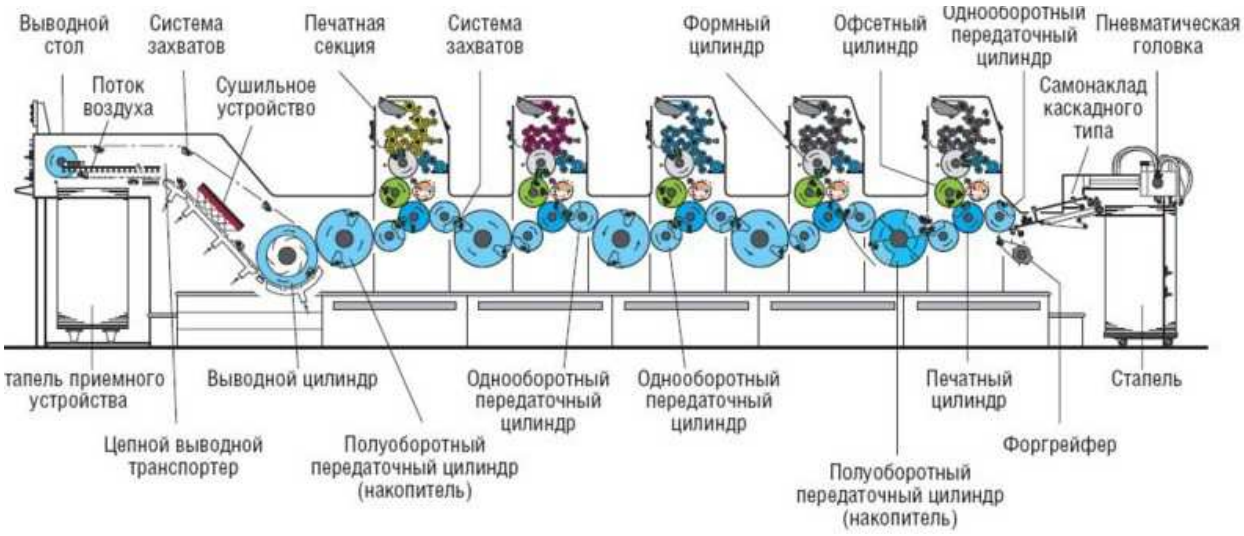


Рисунок 1.2 – Принцип работы станка офсетной печати

Затем бумага проходит через выводной цилиндр, для заключительного этапа-сушки с помощью инфракрасных сушек и обдувки потоками воздуха. Затем через выводной стол готовая продукция попадет в приёмного устройства, а оттуда уже на склад или покупателю.

2 Состав электрооборудования используемого в работе

2.1 Общие сведения о вентильном двигателе

Вентильный электродвигатель – разновидность электродвигателя переменного тока, конструкция которого подобна электродвигателю постоянного тока, но не имеющего коллекторно-щеточного узла. Вместо него используется бесконтактный полупроводниковый коммутатор. В зависимости от конструктивного исполнения существуют два вида электродвигателей:1) бесконтактные электродвигатели переменного тока, 2) бесконтактные электродвигатели постоянного тока.

Вентильные двигатели имеют ряд преимуществ, таких как высокое значение коэффициента полезного действия, что позволяет увеличить время их работы от автономного источника питания, отсутствию подвижных контактов, лучшие массогабаритные показатели по сравнению с другими электрическими двигателями аналогичных мощностей, широкий диапазон скоростей и момента.

Применяются в таких установках, как вентиляторы, системах кондиционирования, станки, авиапромышленность и т.д.

Главным недостатком вентильных электродвигателей является высокая стоимость из-за использования постоянных магнитов, в результате чего в ряде случаев выгодней использовать асинхронный двигатель с преобразователем частоты.

В данной работе используется вентильный двигатель G1G170-AB31-08, применяемый в системе вентиляции станка speedmaster 52.

Рисунок - 2.1 Электродвигатель G1G170-AB31-08

Таблица 1 – Технические характеристики электродвигателя G1G170-AB31-08

Мощность, Вт	Номинальная частота вращения, об/мин	Номинальный ток, А	КПД, %	Масса, кг	cosφ
315	5650	2,3	61,3	4,5	0,95

G1G170-AB31-08 является однофазным бесконтактным вентильным двигателем постоянного тока, имеющим постоянные магниты на роторе. Статор состоит из сердечника из электротехнической стали и медной обмотки, уложенной в пазы сердечника.

2.2 Датчик Холла

Основным элементом в управлении вентильным двигателем является элемент для реализации обратной связи—датчик положения ротора, именуемый датчиком Холла SS41. Датчик Холла—прибор, измеряющий напряженность электромагнитного поля. Принцип работы основан на эффекте Холла, суть которого заключается в перемещении и распределении в проводнике носителей заряда под воздействием магнитного поля, вызывающих появление разности потенциалов.

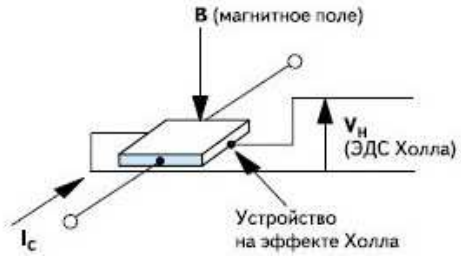


Рисунок - 2.2 Эффект Холла

Располагается рядом со статором вентильного электродвигателя для реализации обратной связи.

Рисунок 2.3-Датчик Холла SS41

Датчик Холла содержит в своей конструкции: постоянный магнит, пластиковый корпус, выводы питания, сигнала и заземления, микросхему.

2.3 Плата управления G1G170-AB31-08

Для реализации управления вентильным электродвигателем используется плата управления, выполняющая следующие задачи:

-электрическое питание отдельных элементов, таких как датчик Холла;
-преобразование переменного тока в постоянный;
-коммутация обмоток вентильного двигателя для подключения к источнику постоянного напряжения разной полярности.

Рисунок 2.4-Плата управления G1G170-AB31-08

Преобразование переменного тока в постоянный осуществляется за счет выпрямительного моста, встроенного в плату. Затем выпрямленное напряжение поступает на встроенный в плату блок питания, необходимого для подачи электроэнергией сопутствующих элементов схемы управления.

Силовые транзисторы коммутируют обмотки двигателя с помощью цифрового драйвера int100s, который управляется логическим микроконтроллером Arduino.

С помощью широтно-импульсной модуляции (ШИМ) формируется задание на скорость вращения вентильного двигателя и логический микроконтроллер осуществляет управление int100s с целью поочередного открытия полупроводниковых ключей (транзисторов).

В качестве сигнала запуска двигателя используется Simens s7-200 - контроллер, применяемый для систем автоматизации низкой и средней сложности, применяется для управления и регулирования скорости двигателя. Установлен в машине для офсетной печати Heidelberg speedmaster 52.

3 Программируемый логический контроллер

3.1 ArduinoMega2560

Первое и главное преимущество ПЛК, обусловившее их широкое распространение, заключается в том, что одно компактное электронное устройство может заменить десятки и сотни электромеханических реле. Второе преимущество в том, что функции логических контроллеров реализуются не аппаратно, а программно, что позволяет постоянно адаптировать их к работе в новых условиях с минимальными усилиями и затратами.

ПЛК отличаются от традиционных неперепрограммируемых устройств управления следующими преимуществами: они более гибки, надёжнее, имеют меньшие габариты, могут быть объединены в сети с другими устройствами и перенастраиваться по Интернету, быстрее обнаруживают ошибки, расходуют меньше электроэнергии, требуют меньше затрат на изменение своих функций и структуры и менее затратны на больших отрезках времени.

Применение ПЛК обеспечивает высокую надёжность, простое тиражирование и обслуживание систем управления, ускоряет монтаж и наладку оборудования, обеспечивает возможность быстрого обновления алгоритмов управления (в том числе и на работающем оборудовании).

Arduino Mega 2560 - это новое семейство микроконтроллеров Ардуино для решения самых разных задач автоматизации малого уровня. Это устройство на основе микроконтроллера ATmega2560. Плата имеет 54 цифровых входа/выхода, 15 из которых используются как ШИМ, 16 аналоговых входов, 4 аппаратных последовательных входа для реализации последовательных интерфейсов UART, кварцевый резонатор с частотой 16МГц, USB порт, разъём питания, разъём ISCP для программирования в устройстве по последовательному протоколу и кнопка сброса микроконтроллера. Для начала работы с устройством достаточно просто подать питание от AC/DC-адаптера или батарейки, либо подключить его к компьютеру посредством USB-кабеля. Arduino Mega совместим с большинством плат расширения, разработанных для Arduino Duemilanove и Diecimila.

Mega2560 имеет компактный размер: длиной 10,2 см и шириной 5,4 см с учетом USB разъёма и разъёма питания, выступающих из корпуса платы. Плата может работать от источника питания в диапазоне от 6 до 20 вольт. При меньшем напряжении плата работает нестабильно. При более высоком напряжении плата начинает нагреваться и может выйти из строя.

Выводы питания Arduino Mega 2560 перечислены ниже:

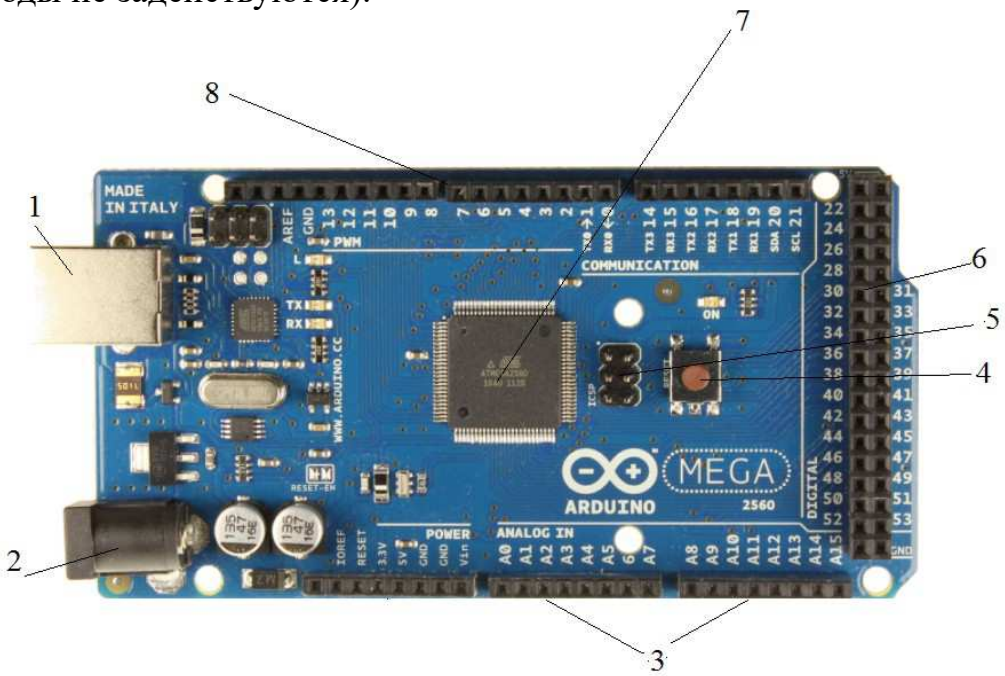
-VIN. Напряжение, поступающее в Arduino непосредственно от внешнего источника питания (при отсутствии подачи напряжения на USB порт или

иного сточника). Через этот вывод можно как подавать внешнее питание, так и потреблять ток, когда устройство запитано от внешнего адаптера.
-5V. На этот 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 Ардуино. 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.
Arduino Mega 2560 имеет 225 Кб флэш-память, которая используется для хранения программного кода, 8Кб из которых используются для загрузки и 4 Кб энергонезависимой памяти служащей для работы с библиотекой EEPROM.

Каждый из 54 цифровых пинов на Arduino Mega может работать в режиме входа или выхода, используя функции pinMode, digitalWrite и digitalRead. Выходы работают на 5 В. Каждый пин может отдать или принять максимум 40 мА и имеет внутренний подтягивающий резистор 20-50 кОм (отключен по умолчанию). Плюс к этому, некоторые выводы имеют специальные функции:
- Последовательный интерфейс Serial: 0 (RX) и 1 (TX); Serial 1: 19 (RX) и 18 (TX); Serial 2: 17 (RX) и 16 (TX); Serial 3: 15 (RX) и 14 (TX). Данные выводы используются для получения (RX) и передачи (TX) данных по последовательному интерфейсу. 0 и 1 подключены к выводам микросхемы ATmega16U2, выполняющей роль преобразователя USB-to- ATmega16U2.
-ШИМ выводы 2-13 и 44-46 с помощью функции analogWrite могут выводить 8-битные аналоговые значения в виде ШИМ -сигнала.
- Интерфейс SPI: выводы 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Выводы осуществляют передачу связи по интерфейсу SPI, используя библиотеку SPI. Пины SPI могут быть выведены на ISCP, совместимый с Arduino Uno, Duemilanove и Diecimila.
-LED 13. Это встроенный в плату светодиод, который включен в 13 вывод. При подаче сигнала HIGH, светодиод загорается, при подаче сигнала LOW-выключается. Служит сигналом об успешной загрузки программы в плату.
-Выводы 20(SDA) и 21(SCL) позволяют осуществлять передачу связи по интерфейсу TWI.

В Mega2560 имеются 16 аналоговых входов, каждый из которых можно представить в аналоговое напряжение в виде 10-битного числа.
-Вывод AREF опорного напряжения для аналоговых входов
-Вывод RESET подаёт низкий уровень сигнала для перезагрузки микроконтроллера.

Arduino Mega 2560 предоставляет ряд возможностей для осуществления связи с компьютером, еще одним Ардуино или другими микроконтроллерами. В ATmega2560 есть четыре аппаратных приёмо-передатчика UART для реализации последовательных интерфейсов (с логическим уровнем TTL 5В). Микроконтроллер ATmega16U2 обеспечивает связь одного из приемопередатчиков с USB-портом компьютера, и при подключении к ПК позволяет Ардуино определяться как виртуальный COM-порт. В пакет программного обеспечения Ардуино входит специальная программа SerialMonitor, позволяющая считывать и отправлять на Ардуино простые текстовые данные. При передаче данных через микросхему ATmega8U2/ATmega16U2 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).



1. UCB разъём;
2. внешнее питание;
3. 16 аналоговых входов;
4 .кнопка перезагрузки;5-ISCP;
5 .54 цифровых входа/выхода;
6 .ATmega2560;
7 . входы ШИМ

Рисунок 3.1- Программированный микроконтроллер ArduinoMega2560

Библиотека SoftwareSerial позволяет работать с подключением по последовательной связи на любых цифровых выводах Mega2560.

В микроконтроллере ATmega2560 также реализована аппаратная поддержка последовательных интерфейсов TWI и SPI. В программное обеспечение Ардуино входит библиотека Wire, позволяющая упростить работу с шиной TWI. Для работы с интерфейсом SPI используется библиотека SPI.

Для работы с Arduino Mega, необходимо использовать программное обеспечение Arduino IDE.

Микроконтроллер ATmega2560 на плате Arduino Mega поставляется с прошитым загрузчиком, который позволяет загружать новый код в микроконтроллер без использования внешнего аппаратного программатора. Загрузчик использует оригинальный протокол ST. Также есть возможность не использовать загрузчик и программировать микроконтроллер через выводы блока ISCP, используя Arduino ISP или аналогичный.

Исходный код прошивки ATmega16U2 доступен для скачивания в репозитории Arduino. ATmega16U2/8U2 загружается, используя загрузчик DFU, для активации которого необходимо:

-На платах версии R1: замкнуть перемычку на обратной стороне платы (возле изображения карты Италии), после перезагружаем 8U2.

-На платах версий R2 и выше для упрощения перехода в режим DFU присутствует резистор, подтягивающий к земле линию HWB микроконтроллера 8U2/16U2. После перехода в DFU-режим для загрузки новой прошивки можно использовать программное обеспечение Atmel's FLIP (для Windows).

В данной дипломной работе микроконтроллер ArduinoMega2560 применялся для проведения тестовых испытаний, однако вследствии больших габаритов самой платы в готовой установке не применялась. Вместо ArduinoMega2560 использовался микроконтроллер Arduino Nano, описание которого приведино в следующей главе.

3.2 Arduino Nano

Платформа Nano, построенная на микроконтроллере ATmega328 (Arduino Nano 3.0), имеет небольшие размеры и может использоваться в лабораторных работах и местах ограниченного пространства. Она имеет схожую с Arduino Duemilanove функциональность, однако отличается сборкой. Отличие заключается в отсутствии силового разъема постоянного тока и работе через кабель Mini-B USB. Nano разработана и продается компанией Gravitech.

Arduino Nano может получать питание через подключение Mini-B USB, или от нерегулируемого 6-20 В (вывод 30), или регулируемого 5 В (вывод 27), внешнего источника питания. Автоматически выбирается источник с самым высоким напряжением.

Микросхема FTDI FT232RL получает питание, только если сама платформа запитана от USB. Таким образом при работе от внешнего источника (не USB), будет отсутствовать напряжение 3.3 В, генерируемое микросхемой FTDI, при этом светодиоды RX и TX мигаю только при наличие сигнала высокого уровня на выводах 0 и 1. Микроконтроллер ATmega328, в свою очередь, имеет 32 кБ флеш-памяти. ATmega328 имеет 2 кБ ОЗУ и 1 Кб EEPROM.

Каждый из 14 цифровых выводов Nano, используя функции pinMode(), digitalWrite(), и digitalRead(), может настраиваться как вход или выход. Выводы работают при напряжении 5 В. Каждый вывод имеет нагрузочный резистор (стандартно отключен) 20-50 кОм и может пропускать до 40 мА. Некоторые выводы имеют особые функции:

-Последовательная шина: 0 (RX) и 1 (TX). Выводы используются для получения (RX) и передачи (TX) данных TTL. Данные выводы подключены к соответствующим выводам микросхемы последовательной шины FTDI USB-to-TTL.

-Внешнее прерывание: 2 и 3. Данные выводы могут быть сконфигурированы на вызов прерывания либо на младшем значении, либо на переднем или заднем фронте, или при изменении значения. Подробная информация находится в описании функции attachInterrupt().

-ШИМ: 3, 5, 6, 9, 10, и 11. Любой из выводов обеспечивает ШИМ с разрешением 8 бит при помощи функции analogWrite().

-SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Посредством данных выводов осуществляется связь SPI, которая, хотя и поддерживается аппаратной частью, не включена в язык Arduino.

-LED: 13. Встроенный светодиод, подключенный к цифровому выводу 13. Если значение на выводе имеет высокий потенциал, то светодиод горит.

На платформе Nano установлены 8 аналоговых входов, каждый разрешением 10 бит (т.е. может принимать 1024 различных значения). Стандартно выводы имеют диапазон измерения до 5 В относительно земли, тем не менее имеется возможность изменить верхний предел посредством функции analogReference(). Некоторые выводы имеют дополнительные функции:

-I2C: A4 (SDA) и A5 (SCL). Посредством выводов осуществляется связь I2C (TWI). Для создания используется библиотека Wire (информация на сайте Wiring).

Дополнительная пара выводов платформы:

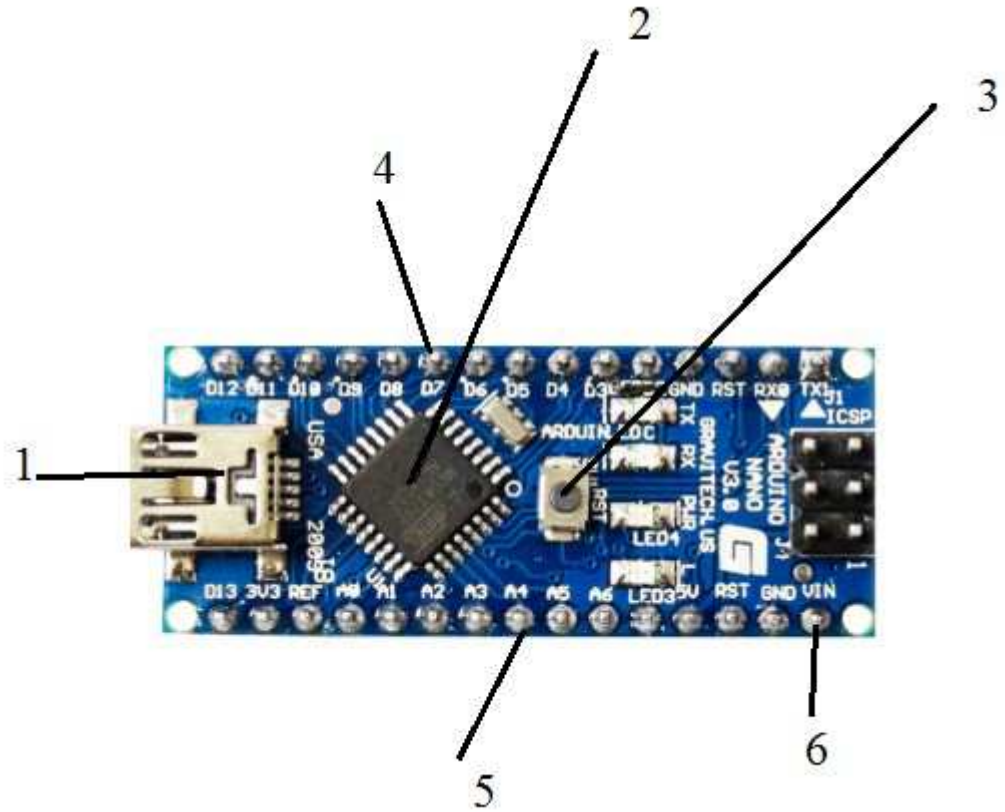
-AREF. Опорное напряжение для аналоговых входов. Используется с функцией analogReference().

-Reset. Низкий уровень сигнала на выводе перезагружает микроконтроллер. Обычно применяется для подключения кнопки перезагрузки на плате расширения, закрывающей доступ к кнопке на самой плате Arduino.

На платформе Arduino Nano установлено несколько устройств для осуществления связи с компьютером, другими устройствами Arduino или

микроконтроллерами. ATmega328 поддерживают последовательный интерфейс UART TTL (5 В), осуществляемый выводами 0 (RX) и 1 (TX). Установленная на плате микросхема FTDI FT232RL направляет данный интерфейс через USB, а драйверы FTDI (включены в программу Arduino) предоставляют виртуальный COM порт программе на компьютере. Мониторинг последовательной шины (Serial Monitor) программы Arduino позволяет посылать и получать текстовые данные при подключении к платформе. Светодиоды RX и TX на платформе будут мигать при передаче данных через микросхему FTDI или USB подключение (но не при использовании последовательной передачи через выводы 0 и 1).

Библиотекой SoftwareSerial возможно создать последовательную передачу данных через любой из цифровых выводов Nano.



1. микро-UCB разъём;
2. ATmega328;
3 .кнопка перезагрузки;
4 .14 цифровых входа/выхода, 6 из которых могут использоваться как выход ШИМ;
5. 8 аналоговых входов;
6. внешние питание с вывода VIN;

Рисунок 3.2- Программированный микроконтроллер Arduino Nano

ATmega328 поддерживают интерфейсы I2C (TWI) и SPI. В Arduino включена библиотека Wire для удобства использования шины I2C. Более подробная информация находится в документации.

Nano разработана таким образом, чтобы перед записью нового кода перезагрузка осуществлялась самой программой, а не нажатием кнопки на платформе. Одна из линий FT232RL, управляющих потоком данных (DTR), подключена к выводу перезагрузки микроконтроллеров ATmega168 или ATmega328 через конденсатор 100 нФ. Активация данной линии, т.е. подача сигнала низкого уровня, перезагружает микроконтроллер. Программа Arduino, используя данную функцию, загружает код одним нажатием кнопки Upload в самой среде программирования. Подача сигнала низкого уровня по линии DTR скоординирована с началом записи кода, что сокращает таймаут загрузчика.

Функция имеет еще одно применение. Перезагрузка Nano происходит каждый раз при подключении к программе Arduino на компьютере с ОС Mac X или Linux (через USB). Следующие полсекунды после перезагрузки работает загрузчик. Во время программирования происходит задержка нескольких первых байтов кода во избежание получения платформой некорректных данных (всех, кроме кода новой программы). Если производится разовая отладка скетча, записанного в платформу, или ввод каких-либо других данных при первом запуске, необходимо убедиться, что программа на компьютере ожидает в течение секунды перед передачей данных.

4 Принципиальная электрическая схема

Принципиальная схема платы управления электродвигателем приведена на листе №1 графической части. Для коммутации обмотки электродвигателя используются драйвера int100s. Драйвер int100s-это полумостовой драйвер, обеспечивающий управление силовыми ключами - полевыми транзисторами T1-T4 (МОП-транзисторы). Каждый драйвер управляет своей парой ключей.

Драйвер int100s обеспечивает простой экономичный интерфейс между низковольтной логикой управления Arduino выходы (9 и 10) и высоковольтными нагрузками. Предназначен для выпрямленных напряжений 110В или 220В.

Встроенная логика защиты предотвращает одновременное включение обоих ключей и замыкания питания высокого напряжения. Импульсный внутренний сдвиг уровня обеспечивает повышенную помехозащищенность. Номинальное напряжение питания драйверов 15В.

Функциональное описание контактов драйвера int100s показанных на схеме:

-1 контакт: Подаёт питание 15В на драйвер;

-2 контакт: Вход логического уровня с активным низким уровнем HS IN управляет выходным высоковольтным сигналом;

-3 контакт: Вход логического уровня с активным высоким уровнем LS IN управляет выходным низковольтным сигналом;

-4,5 контакт: Используются в качестве аналоговой контрольной точки для схемы;

-7 контакт: Является опорной точкой для питания низковольтной цепи и подключен к источнику низковольтного питания полевого транзистора;

-8 контакт: Выходной сигнал драйвера, управляющий транзистором T2(T4);

-11 контакт: Выходной сигнал драйвера, управляющий транзистором T1(T3);

-12,13,14 контакт: Является опорной точкой для питания высоковольтной цепи и подключен к источнику высоковольтного питания полевого транзистора;

-15 контакт: Подаёт питание на управляющую логику.

Сигнал LS IN напрямую управляет полевым транзистором T2(T4). Сигнал HS IN управляет транзистором T1(T3) с помощью сдвига высокого уровня напряжения, взаимодействующего с приводом. Драйвер будет игнорировать сигналы, которые будут командовать одновременным открытием T1 и T2 (соответственно T3 и T4), защищая от замыкания.

Локальное шунтирование для драйвера со стороны низкого потенциала обеспечиваются конденсатором C1 (C4). Смещение нагрузки для драйвера со стороны высокого потенциала обеспечивается диодом D1(D4) и конденсатором C2(C3). Скорость нарастания и эффект паразитных колебаний регулируются резисторами R2 и R3 (R8 и R7 для второго драйвера

соответственно). Изоляция высоковольтной шины от драйвера обеспечивается резисторами R1 и R4(R5 и R6) и диодами D2 и D3(D5 и D6). Входы контактов спроектированы так, чтобы быть совместимыми с 5В-уровнями сигналов.

Входы HS IN и LS IN изолированы друг от друга внутренними высоковольтными переключателями.

Максимальная частота работы ограничена рассеиванием мощности из-за переключения высокого напряжения, заряда затвора и мощности смещения. . На рисунке 4.1 показана максимальная частота переключения в зависимости от входного напряжения и заряда затвора. Для более высоких температур окружающей среды частота переключения должна линейно снижаться.

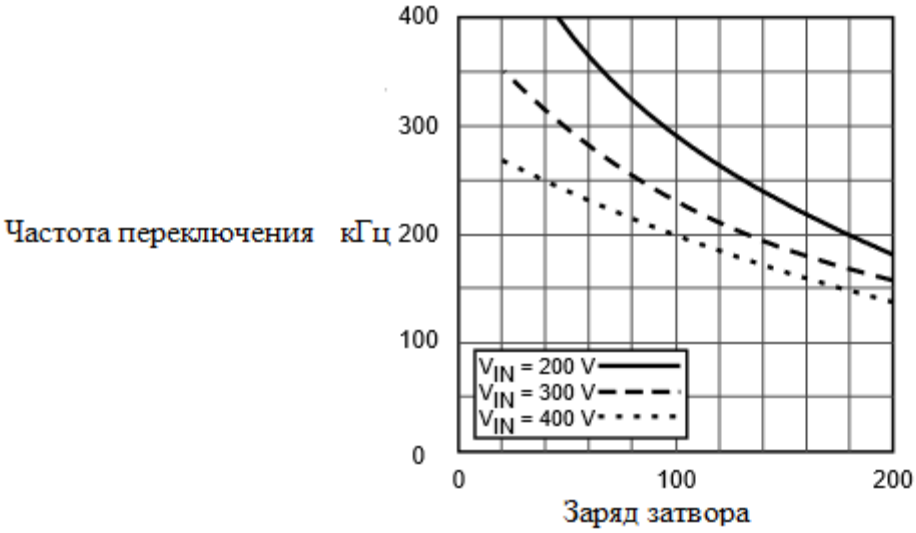


Рисунок 4.1- Зависимость заряда затвора от частоты переключения

Конденсатор нагрузки C2(C3) должен иметь достаточно большой заряд, чтобы обеспечить ток смещения в течении длительного времени работы драйвера без значительного спада напряжения.

Питание электрической платы приведена на листе №1 графической части и представляет собой бестрансформаторный источник питания с балластным конденсатором C5. Конденсатор C5 - балластный, используется для ограничения сетевого повышенного напряжения и бросков тока при подключении. Выпрямленное напряжение поступает с мостовой схемы выпрямления VD1 со сглаживающим конденсатором C9. Переменное сетевое напряжение поступает на дроссель L1 для обеспечения постоянного по величине тока. Для защиты схемы от пульсаций напряжения сети и ограничения тока применяются варистор RU1 и предохранитель F1. Конденсаторы C6,C7,C8 образуют фильтр сглаживающий пульсации. В качестве защиты от большого значения входного тока в схему включен

термистор TH1. Для стабилизации постоянного напряжения используется стабилитрон VD3.

Принципиальная электрическая схема блока питания драйверов представлена на листе №1 графической части. Для питания драйверов используется в качестве преобразователя напряжения микросхема TOP210PFI относящаяся к семейству микросхем TOPSwitch для импульсных источников питания. Для реализации всех функций, необходимых для автономной системы управления используются: высокочастотный силовой полевой транзистор с управляемым драйвером затвора, схема отключения высокого напряжения, защитная схема от помех, усилитель помех для обратной связи. TOP210PFI является интегральной схемой, что позволяет снизить общую стоимость, количество компонентов, вес и габариты, что позволяет повысить эффективность и надежность автономной системы питания.

Схема микросхемы показана на рисунке 4.2. Функциональное назначение выводов:

-1 вход: Вход управления, присоединенный к входному источнику;

-8 вход: Выходное соединение для возврата высокого напряжения;

-4 вход: Усилитель ошибок и выходного тока обратной связи для управления рабочим циклом и обеспечения контроля отклонения. Также используется для подключения шины питания и автоматического перезапуска/ компенсации конденсатора;

-5 вход: Обеспечивает внутренний ток смещения при запуске через внутренний источник тока с высоким напряжения.

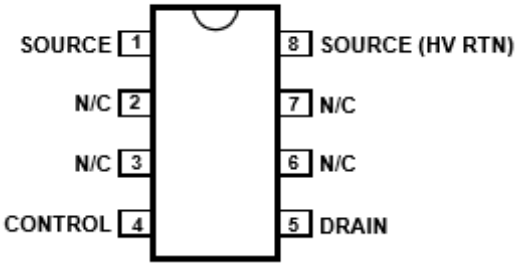


Рисунок 4.2- Принципиальная схема микросхемы TOP210PFI

Использование интегральной схемы позволяет значительно снижать токи смещения, устраняются внешние силовые резисторы, используемые для измерения тока и/или подачи начального тока смещения пуска. Во время нормальной работы внутренний цикл работы полевого транзистора линейно уменьшается с увеличением тока на входе 4, как показано на рисунке 4.3. Для обеспечения необходимых функций управления, защиты и смещения, входы 5 и 4 выполняю несколько функций.

Вход 4 используется как источник питания и смещения для схемы. Для подачи тока затвора полевого транзистора используется внешний конденсатор, связанный между входами 4 и 1. Емкость подобранна таким

образом, чтобы обеспечить время для автоматического перезапуска и компенсации контура управления.

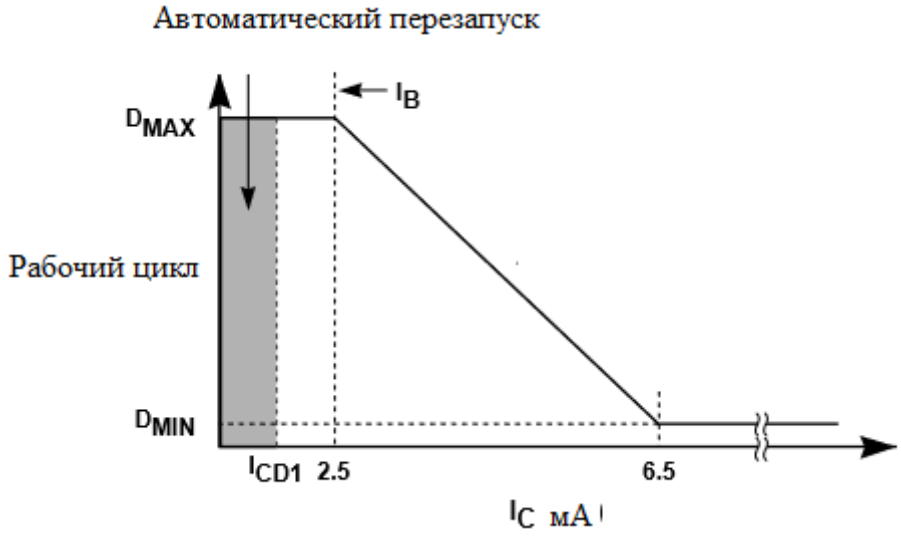


Рисунок 4.3 - Зависимость рабочего цикла от тока входа 4

Напряжение питания регулируется в обоих режимах работы. Гистерезисное регулирование используется для начального запуска. Регулирование шунта используется для отделения сигнала ошибки рабочего цикла от тока питания цепи управления. Во время пуска ток питания подается от источника постоянного тока с высоким напряжением, подключенного между контактами 5 и 4. Источник тока обеспечивает достаточный ток для питания схемы управления, а также заряжает общую внешнюю емкость конденсатора.

При первом запуске, при достижении напряжения питания допустимого уровня, высоковольтный источник питания отключается, а выходной полевой транзистор и ШИМ модулятор активируются. Во время нормальной работы (при регулировании выходного напряжения) ток управления с обратной связью подает напряжение питания. Ток обратной связи входа 4, превышающего требуемое значение постоянного тока питания, поддерживается постоянное напряжение. Динамическое сопротивления входа 4 вместе с внешним сопротивлением и емкостью определяет компенсацию контура управления системы питания.

При разряжении внешней емкости входа 4, выходной полевой транзистор отключается, а схема переходит в режим ожидания. Высоковольтный источник тока включается и снова заряжает внешнюю емкость. Ток зарядки индицируется с отрицательной полярностью, а ток разряда отображается с положительной полярностью. Гистерезисный регулятор повторного перезапуска удерживает напряжение питания в допустимом диапазоне 4,7В до 5,7В путем включения и выключения высоковольтного источника тока. В схеме автоматического перезапуска имеется счетчик, который предотвращает включение выходного полевого транзистора до тех пор, пока не истекут восемь циклов разрядки. Счетчик эффективно ограничивает рассеивание мощности TOPSwitch за счет

снижения коэффициента автоматического перезапуска до 5%. Автоматический перезапуск продолжает цикл до тех пор, пока не будет достигнуто регулирование выходного напряжения.

ШИМ модулятор реализует управление напряжением за счет возбуждения выходного полевого транзистора с рабочим циклом, обратно пропорциональным току, поступающему в контакт 4. Сигнал ошибки фильтруется RC-цепью с типичной угловой частотой 7 кГц для уменьшения эффекта шума. Отфильтрованный сигнал ошибки сравнивается с сигналом внутреннего пилообразного сигнала генератора для генерации формы сигнала рабочего цикла. По мере увеличения управляющего тока рабочий цикл уменьшается. Сигнал синхронизации от генератора устанавливает защелку, которая включает выходной транзистор. ШИМ модулятор сбрасывает защелку, отключая выходной полевой транзистор. Максимальный рабочий цикл устанавливается симметрией внутреннего генератора. Модулятор имеет минимальное время включения, чтобы поддерживать ток потребления TOPSwitch независимо от сигнала ошибки. Минимальный ток должен быть включен в контакт 4 до того, как рабочий цикл начнет меняться.

Встроенный драйвер предназначен для включения выходного полевого транзистора с контролируемой скоростью.

Усилитель ошибок используется для усиления сигнала ошибок с целью обеспечения обратной связи. Коэффициент усиления усилителя ошибки задается динамическим сопротивлением выхода 4. Выход 4 подключает внешние сигналы схемы к уровню напряжения питания. Ток входа 4, превышающий ток питания протекает как сигнал ошибки.

Для ограничения пикового тока цикла используется выходное сопротивление полевого транзистора в качестве резистора. Компаратор ограничения тока сравнивает выходное напряжение источника транзистора с пороговым напряжением. Высокий ток стока приводит к тому, что напряжение превышает пороговое напряжение и отключает выходной полевой транзистор до начала следующего тактового цикла. Пороговое напряжение тока предельного компаратора компенсируется температурой, чтобы свести к минимуму изменение предела пикового тока из-за связанных с температурой изменений выходного полевого транзистора.

В течение короткого времени после включения полевого транзистора блокируется компаратор ограничения тока. Время установлено так, чтобы ток, возникающий из-за емкостей первичной стороны, и времени обратного восстановления выпрямителя вторичной стороны, не приведет к преждевременному прекращению импульса переключения.

Схема, показанная на листе №1, создает источник питания 15 В, 8 Вт, который работает от входного напряжения от 170 В до 264 В постоянного тока. Выходное напряжение 15 В определяется напряжением шунтирующим входом регулятора TOPSwitch 4-й вход, падениями напряжения на диодах D9 и D10 и коэффициентом трансформации между выходными обмотками

трансформатора T1. R9 и C13 служат в качестве фильтра для обмотки смещения.

Для получения постоянного (не пульсирующего) напряжения, применяемого к первичной обмотке T1, используется мостовая схема выпрямления VD1 вместе с конденсаторами C9 и C10. Другая сторона первичной обмотки трансформатора управляется встроенным высоковольтным полевым транзистором в TOP210. D8 и VD2 уменьшают всплеск напряжения, вызванный индуктивностью утечки трансформатора, до безопасного значения и уменьшают количество шумов. Напряжение на вторичной обмотке выпрямляется и фильтруется с помощью D9, C11, L2 и C12 для создания выходного напряжения 15 В. R10 обеспечивает предварительную нагрузку на выход 15 В для улучшения регулирования нагрузки. Обмотка смещения выпрямляется и фильтруется D10, R9 и C13 для создания напряжения смещения в TOP210. Конденсатор C14 ослабляет токи, вызванные импульсами высокого напряжения на стороне первичной обмотки. Дроссель L1 и конденсатор C5 ослабляют дифференциальные токи эмиссии, вызванные фундаментальными гармониками трапецеидальной формы первичного тока. Конденсатор C13 фильтрует внутренние импульсы тока зарядки затвора транзистора на выходе 4, определяет частоту перезапуска и вместе с резистором R9 компенсирует контур управления.

Для коммутации обмотки двигателя используется логический микроконтроллер ArduinoNano, который управляет драйверами int100s. Внешний сигнал на включение двигателя, обозначенный как ШИМ, поступает через токоограничивающий резистор R11 с внешнего генератора, используемого вместо программируемого контроллера Simens s7-200. Arduino принимает сигнал генератора через высокоскоростной оптрон с транзисторным выходом VU1 на порт 8, где R12 подтягивающий резистор. Напряжение на базу и коллектор транзистора оптрона подаётся от микроконтроллера.

Датчик Холла, обеспечивающий обратную связь, получает питание от микроконтроллера и подает сигнал о положении ротора двигателя на вход 2.

Входы 9 и 10 подают ШИМ сигнал на драйверы, сформированный Arduino. Каждый вход микроконтроллера управляет одним драйвером. Одновременная подача сигнала с одного порта микроконтроллера на 2-й и 3-й вход драйвера int100s обеспечивается перемычкой, соединяющий оба входа драйвера (рисунок 4.4).

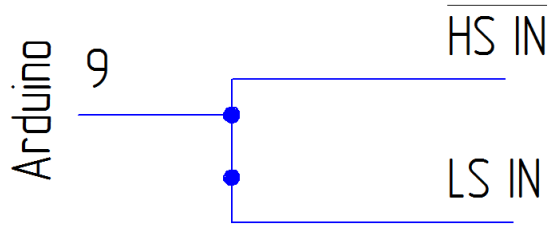


Рисунок 4.4- Пример соединения входов драйвера и 9-го входа Arduino перемычкой

5 Программирование системы управления

5.1 Описание среды программирования Arduino IDE

Для создания разработок на базе Arduino, необходимо программное обеспечение для написания и загрузки программ в микроконтроллер. Данные функции выполняются с помощью Arduino IDE. Среда разработки Arduino состоит из встроенного текстового редактора программного кода, панели инструментов, окна вывода текста. Для загрузки программ необходимо подключить среду разработки к аппаратной части микроконтроллера.

Любая программа, написанная в среде программирования Arduino IDE, называется скетч. Скетч пишется в текстовом редакторе, имеющим инструменты, позволяющие производить над ним любые действия, такие как вставка/удаление, замена/поиск текста. Во время сохранения или компиляции скетча в области сообщений появляются пояснения, либо сообщения об ошибке, в случае неправильного написания программного кода.

Язык программирования Arduino стандартный C++, имеющим особенности, облегчающими написание программ:

-Файлы программ перед компиляцией обрабатываются препроцессором Arduino;

-Программист обязан написать две обязательные функции: setup() вызываемая при старте и loop() повторяющаяся в бесконечном цикле;

-В текст программы не обязательно записывать заголовочные файлы при использовании стандартных библиотек;

-Отсутствие предварительных настроек компилятора.

Загрузка скетча происходит через встроенный загрузчик (Bootloader), являющийся программой, которая позволяет загружать программный код без использования дополнительных аппаратных средств и может работать через интерфейсы RS-232, USB и Ethernet.

Загрузчик активен в течении нескольких секунд при загрузке любого скетча в микроконтроллер. О работе загрузчика сигнализирует светодиод, встроенный в плату микроконтроллера.

Для создания решения задачи управления вентильным электродвигателем в Arduino IDE необходимо выполнить следующие этапы настройки:

- Создание самого проекта;
- Конфигурирование оборудования;
- Подключение устройств к сети;
- Программирование микроконтроллера;
- Настройка визуализации;
- Загрузка данных конфигурации;
- Диагностика работы различных функций оборудования.

Интерфейс программы обеспечивает простую навигацию по задачам и данным проекта (рисунок 5.1):

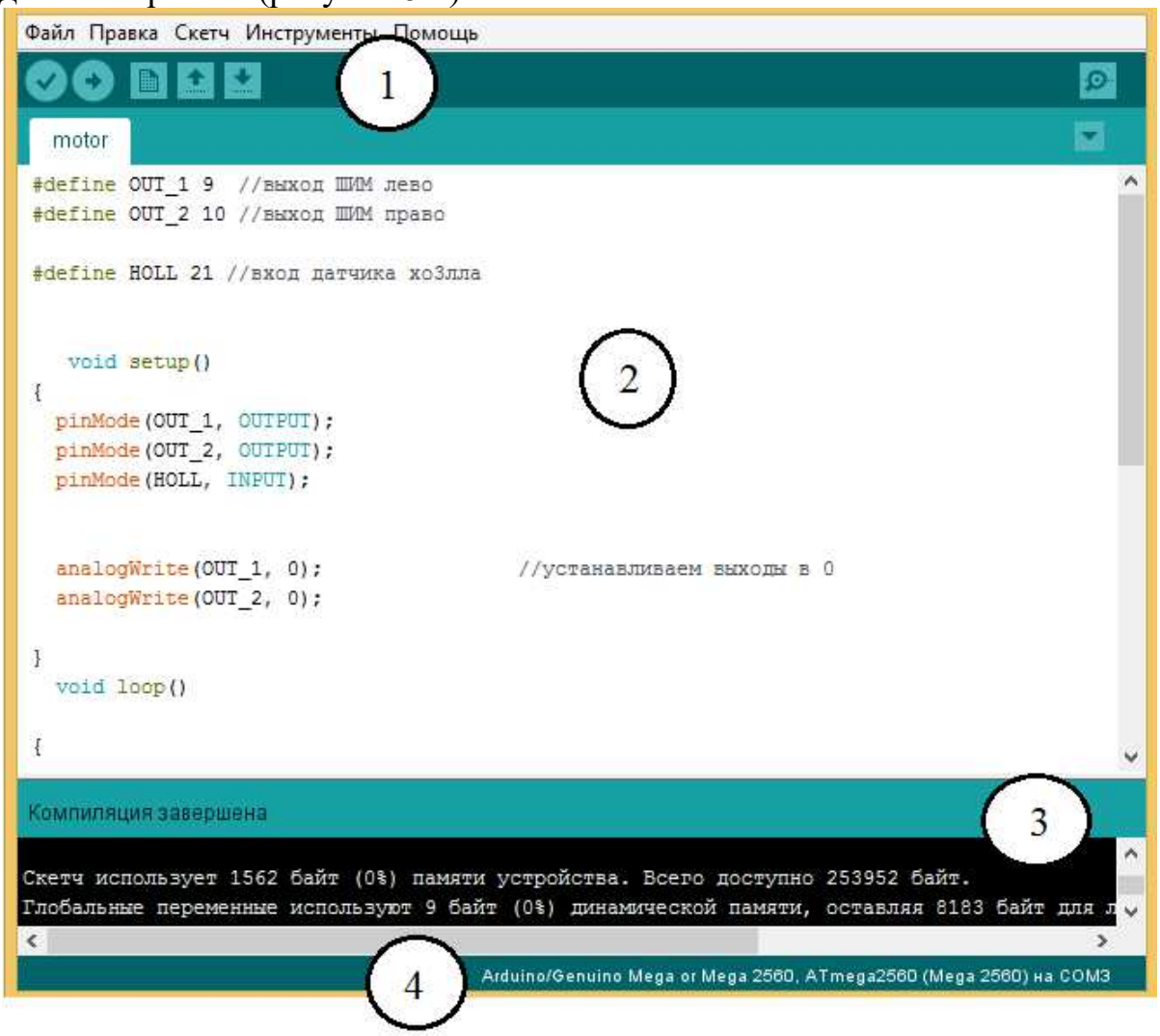


Рисунок 5.1-Структура интерфейса создания скетча

1) Кнопки панелей инструментов, позволяющие проверить, загрузить, создать программу, открыть или сохранить скетч, открыть мониторинг порта;

2) Текстовый редактор, предназначенный для написания скетча;

3) Окно вывода текста (консоль);

4) Область сообщений с информацией о выбранном текущем микроконтроллере и порта вывода.

Arduino IDE также содержит множество предусмотренных библиотек, добавляющие функциональность скетчам при работе с аппаратной частью или обработке данных. Для использования библиотеки необходимо выбрать меню «Скетч > Библиотеки». Одна или несколько директив «#include» будут размещены в начале кода скетча с последующей компиляцией библиотек и вместе со скетчем. Загрузка библиотек требует дополнительного места в памяти Arduino. Неиспользуемые библиотеки можно удалить из скетча убрав директиву «#include». Также присутствует возможность добавление библиотек со сторонних ресурсов и создание собственных.

Выбор аппаратной платформы влияет на скорость передачи данных, скорость ЦП, параметры при компиляции и загрузке скетчей.

Возможность мониторинга порта позволяет отображать посылаемые данные в платформу Arduino (плата USB). Для отправки данных необходимо ввести текст и выбрать меню Инструменты>Мониторинг порта. Затем выбирается скорость передачи данных, соответствующая значению Serial. Набор функций Serial служит для связи устройства Arduino с компьютером или другими устройствами, поддерживающими последовательный интерфейс обмена данными. Для обмена данными Serial используют цифровые порты ввод/вывода 0 (RX) и 1 (TX), а также USB порт. Важно учитывать, что если вы используете функции Serial, то нельзя одновременно с этим использовать порты 0 и 1 для других целей.

Плата Arduino Mega имеет три дополнительных последовательных порта: Serial1 на портах 19 (RX) и 18 (TX), Serial2 на портах на портах 17 (RX) и 16 (TX), Serial3 на портах на портах 15 (RX) и 14 (TX). Чтобы использовать эти порты для связи с компьютером понадобится дополнительные адаптеры USB-to-serial, т.к. они не подключены к встроенному адаптеру платы Mega. Для связи с внешним устройством через последовательный интерфейс необходимо соединить TX порт устройства с RX портом внешнего устройства и RX порт вашего устройства с портом TX внешнего и соедините "землю" на устройствах.

5.2 Описание основных используемых функций Arduino IDE

Язык программирования устройств Arduino основан на C++. Язык Arduino делится на три раздела: операторы, данные и функции.

Пользовательская программа может состоять из нескольких функций, каждая из которых необходима для решения задачи управления.

Директива «#define» изображённая на рисунке 5.2 с помощью которой задаются имена константам перед компиляцией программы. Определенные этой директивой константы не занимают программной памяти, компилятор заменяет все обращения к ним их значениями на этапе компиляции, соответственно они служат исключительно для удобства программиста и улучшения читаемости текста программы.

Недостаток этой функции заключается в том, что при записи имени константы, заданной этой функцией «#define», записать в имени другой переменной, то значение будет изменено.

```
#define knopka 52
#define in1 9
#define in2 10
#define holla 21
```

Рисунок 5.2- Запись констант «#define» в программе скетча

По рисунку 5.2 первой записывается имя входа, удобное для программиста, вторым записывается (присваивается) номер входа микроконтроллера.

Логический тип данных, изображенный на рисунке 5.3, записывается «bool». Принимает значение «true» и «false», занимает в память один байт. «false» определяется в логическом выражении как 0. «true» в логическом выражение принимает значение 1, а также значения -1,-2,-200.

```
bool gtv1; // состояние выхода 255
bool gtv2; // состояние выхода 0
bool count1I = 0; // приращение к мощности
```

Рисунок 5.2- Запись данных «bool» в программе скетча

По рисунку 5.2 count1I - имя переменной типа «bool», 0 - значение, присваиваемое переменной.

Тип данных «int» (рисунок 5.3) используется для хранения чисел в диапазоне от -32768 до 32767 , занимает 2 байта памяти. Arduino компилятор сам заботиться о размещение в памяти и представление отрицательных чисел, поэтому арифметические действия над целыми числами не требуют дополнительных действий.

```
int count1P = 0; // мощность на драйверы
```

Рисунок 5.3- Запись данных «int» в программе скетча

При арифметических операциях переменная типа «int» достигает своего максимального значения, "перескакивает" на минимальное значение и наоборот.

Тип данных записываемые как «unsigned long» используется для хранения положительных целых чисел в диапазоне от 0 до 4294967295 и занимает 32 бита (4 байта) в памяти.

```
unsigned long bounseInputD2P = 0UL;
bool bounseInputD4S = 0;
bool bounseInputD4O = 0;
```

Рисунок 5.3- Запись данных «unsigned long» в программе скетча

Квалификатор «volatile» записывается перед типом переменной и является указателем для компилятора. Он указывает компилятору не загружать переменную из запоминающего регистра – временной ячейки

памяти, в которой хранятся переменные программы и производятся операции с ними. При определенных условиях значения переменных, хранящихся в регистрах, могут оказаться неточными. При объявлении переменной как «volatile», мы задаем микроконтроллеру задание, что значение переменной может быть изменено в любой момент чем-либо за пределами программы, а значит микроконтроллер должен перезагрузить значение переменной каждый раз при её использовании.

При управлении электродвигателем в данной работе переменной «volatile» объявлен сигнал датчика Холла, с целью исключения возможности пропуска сигналов, выполняющих роль обратной связи.

Функция «setup()», вызываемая при старте скетча. Инициализирует переменные, определяет режим работы выводов контроллера, запускает используемые библиотеки. Запускается один раз после каждой подачи питания на микроконтроллер.

Функция «loop()» повторяющаяся в бесконечном цикле. Инициализирует и устанавливает первоначальные значения переменных, производит вычисления в программе и реагирует на них.

Функция «pinMode» используется для инициализации режима работы выводов платы микроконтроллера. Запись pin инициализирует вход микроконтроллера, который будет использоваться в работе. Mode устанавливает режим работы входы или выход.

```
pinMode(52, INPUT);
pinMode(in1, OUTPUT);/
pinMode(in2, OUTPUT);/
pinMode(holla, INPUT);
```

Рисунок 5.4- Функция «pinMode()» в программе скетча

При записи функции pinMode(52, INPUT) первая переменная записывается как номер входа микроконтроллера или присвоенное ему имя директивой «#define», затем режим работы.

Данная функция необходима, так как все выводы платформы Arduino могут работать как входы и выходы. Все выводы находятся в высокоимпедансном состоянии, то есть порт вывода даёт малую нагрузку на схему, в которую подключена Arduino. Для перевода состояния вывода из одного состояние в другое необходимо малое значение тока. Если порт микроконтроллера никуда не подключен, то значения на нём будут принимать случайные величины из-за наличие электрических помех и ёмкостных связей между портами.

Режим работы «INPUT» устанавливает выбранный порт как вход. Режим работы «OUTPUT» устанавливает порт как вход и будет находится в низкоимпедансном состоянии, пропуская через себя максимально допустимый ток 40мА для платы.

Функция «digitalRead()» записывается по такому же принципу, как и функция «pinMode()», она считывает значение с заданного функцией «pinMode()» входа- «HIGH» или «LOW» цифрового порта.

```
bool sensor = digitalRead(holla);// считываем значение с входа
```

Рисунок 5.5- Функция «digitalRead()»

На рисунке 5.5 запись читается следующим образом: с помощью функции digitalRead()» считывается значение со входа 21, имеющим имя holla. Для упрощения программного кода присваиваем этой записи имя sensor.

Значения входа/выхода цифрового порта «HIGH» может обозначать несколько разное в зависимости от режима порта как «INPUT» или «OUTPUT». Когда порт вход/выхода установлен в режим «INPUT» с помощью функции «pinMode()», и считывается функцией digitalRead()», микроконтроллер отдаст значение «HIGH» напряжение 3В или выше на указанном порту.

Также порт может быть установлен как«INPUT» функцией «pinMode()», и затем установлен в «HIGH» значение функцией «digitalWrite()». Это подключит к порту внутренний подтягивающий резистор 20кОм, что позволит получать постоянное значение «HIGH» при чтение этого порта, если только значение не будет приведено к «LOW» внешней цепью подключенной к этому порту.

Когда порт вход/выхода сконфигурирован как «OUTPUT» функцией «pinMode()», и установлено значение «HIGH» функцией «digitalWrite()», на порту будет постоянное напряжение 5В.

Значение «LOW» также разное для режима «INPUT» и «OUTPUT». Когда порт сконфигурирован как «INPUT», и считывается функцией «digitalRead()», микроконтроллер вернет «LOW» если напряжение на данном порту меньше или равно 2В.

Если же порт установлен в «OUTPUT» и «LOW», то напряжение на выходе порта будет 0 вольт.

Оператор «if...else» осуществляет контроль и проверки над процессом выполнения кода программы.

```
if (pinFiveInput < 500)
{
  // действие A
}
else
{
  // действие B
}
```

Рисунок 5.6- Пример использования оператора «if...else»

На рисунке 5.6 продемонстрирован простой пример использования данного оператора. В скобках записывается условие, которое необходимо проверять и выполнять соответствующие действие А. В случае иного условия оператор «else» выполняет проверку, отличную от условия «if», что позволяет проверять несколько взаимоисключающие проверки. Каждая проверка позволяет переходить к следующему за ней оператору не раньше, чем получит логический результат ИСТИНА. Когда проверка с результатом ИСТИНА найдена, запускается вложенная в нее блок операторов, и затем программа игнорирует все следующие строки в конструкции «if...else». Если ни одна из проверок не получила результат ИСТИНА, по умолчанию выполняется блок операторов в «else», если последний присутствует, и устанавливается действие по умолчанию.

Функция «analogWrite()» выводит аналоговую величину ШИМ. В данной работе используется для управления скоростью электродвигателя. Например analogWrite(9, 255), где 9 это номер порта вывода Arduino (вместо цифры может быть записано присвоенное данному выводу имя, написанное программистом), а число 255 это ширина импульса ШИМ. Диапазон скважности от 0 до 255.

ШИМ на выходе представляет собой прямоугольную волну с заданной шириной импульса, вплоть до вызова нового импульса. Частота ШИМ сигнала для Arduino составляет примерно 500Гц. Для вызова этой функции не используется задающая функция «pinMode()». На используемой в данной работе ArduinoMega2560 ШИМ поддерживает порты с 2 по 13.

Оператор «switch» действует как альтернатива оператору «if», позволяя создавать альтернативный код, выполняемый при разных условиях, как на рисунке 5.7

```
switch (var) {
    case 1:
      //выполняется, когда var равно 1
      break;
    case 2:
      //выполняется когда  var равно 2
      break;
    default:


  }
```

Рисунок 5.7- Пример использования оператора «switch»

Оператор «switch» сравнивает значение переменной var со значением, определяемым оператором «case». Когда условие «case» заданное значение переменной выполняется, выполняется программный код.

Оператор «break» выводит выполнение программного кода из оператора «case». Без «break» оператор «switch» будет продолжать выполнение программы, игнорируя значение переменной оператора «case».

Функция «millis()» используется для работы со временем. Она возвращает количество миллисекунд с момента начала выполнения работы программы. Это количество сбрасывается на ноль, в следствие переполнения значения, приблизительно через 50 дней.

5.3 Разработка программы управления электродвигателем

Далее будут подробно описаны этапы создания проекта по управлению вентильным двигателем. Данные и код программы, разработанные в процессе создания проекта, написаны по порядку.

Номера портов ввода/вывода, записанные далее, использовались при работе с микроконтроллером ArduinoMega2560 для отладки программы, функциональность которых совпадает с соответствующими входами ArduinoNano.

Первый шаг заключается в инициализации портов вывода микроконтроллера Arduino. Для более удобного ориентирования в коде программы с помощью директивы «#define»каждому задействованному выводу будут присвоены имена, более явно выражающие их выполняемые функции.

Таблица 2- Таблица директивы «#define» для программы скетча управления двигателем

Имя вывода	Номер вывода	Описание
Knopka2	52	Цифровой вывод, подающий сигнал на включение двигателя.
in1	9	ШИМ вывод, подающий сигнал управления с заданной шириной импульса на цифровой драйвер int100s
in2	10	ШИМ вывод, подающий сигнал управления с заданной шириной импульса на цифровой драйвер int100s
holla	21	Сигнал датчика Холла, подающий импульсы на контроллер для переключения между выходами ШИМ 9 и 10.

```
#define knopka 52
#define in1 9
#define in2 10
#define holla 21
```

Рисунок 5.8 - Запись инициализации в программе входов/выходов микроконтроллера директивой «#define»

Вывод 52 цифровой вывод, настраиваемый как входы с помощью функций «pinMode()» и «digitalRead()», работает как сигнал на включение электродвигателя.

ШИМ выводы 9 и 10 подают на драйвера сигнал управления с соответствующей шириной импульса для управления скоростью двигателя с помощью функции «analogWrite()».

Датчик Холла подключен к выводу 21, имеющим конфигурацию для выполнения задачи автоматического переключения между ШИМ входами 9 и 10. Это используется для того, что бы не пропускать короткий импульс сигнала с датчика Холла. Таким образом, для получения сигнала переключения, приостанавливается основная последовательность команд. Для управления всеми переменными, присвоим каждому свой тип данных, функциональное значение которых описано в главе 5.2.

```
int gtv2;
bool gtv3 = 0;
bool gtv4 = 0;
bool gen3I = 0;
bool gen3O = 0;
unsigned long gen3P = 0UL;
bool gen1I = 0;
bool gen1O = 0;
unsigned long gen1P = 0UL;
bool count2I = 0;
int count2P = 0;
bool swi2;
```

Рисунок 5.9- Запись типов данных в виде кода в программе

Таблица 3- Таблица пояснения типов данных микроконтроллера.

Имя	Тип данных	Присваиваемое значение
gtv2	int	-
gtv3	bool	0
gtv4	bool	0
gen3I	bool	0
gen30	bool	0
gen3P	unsigned long	0UL
gen1I	bool	0
gen10	bool	0
gen1P	unsigned long	0UL
count2I	bool	0
count2P	int	0
swi2	bool	-

Большинству переменных присваиваются нулевые значения. Алгоритм работы части кода на рисунке 5.10 демонстрирует отслеживание входного сигнала включения двигателя. Переменная gtv3 считывают наличие входного

сигнала, то есть отслеживают изменение входного сигнала включения. Например, если есть сигнал на цифровой вход 52, то нужно увеличивать ширину импульса, за это отвечает переменная gtv3, и значение переменной становится gtv3=1соответственно увеличивать скорость двигателя, если gtv3=0 скорость уменьшается. Когда сигнал на вход 52 микроконтроллера не поступает, нужно уменьшать величину импульса. Тип данных swi2 используется для переключения между переменными. Переменная gtv4 изначально имеет состояние gtv4=0, и отвечает за величину ширины импульса, до которой должен досчитать счётчик. Если gtv4=1, начинает выполняться условие на рисунке 5.15. Переменная принимает значение gtv4=1 только тогда, когда есть сигнал на включение со входа 52.

```
gtv3 = !(!( (digitalRead (52))));
if(gtv4)  /
{swi2=0;}
else
{swi2=gtv3;} /
```

Рисунок 5.10 - Код программы, отслеживания сигнала включения

Переменная gt2, представленная типом данных «int», хранит значение ширины импульса, до значения указанного в программе, которое нужно подавать на выход.

Как показано на рисунке 5.10 приравнивание типов данных swi2 и gtv3 необходимо для запуска «программного генератора». «Генератора» - это таймер.

С помощью переменных gen3I и gen1I устанавливается рабочее состояние кода программы, отвечающего за генерацию ширины импульса в сторону увеличения и уменьшения соответственно.

```
if (swi2)
{ if (! gen3I)
{ gen3I = 1;
gen3O = 1;
gen3P = millis(); } }
else {
  gen3I = 0 ;
  gen3O= 0;}
 if (gen3I)
 {  if ( isTimer ( gen3P , 5 ))
 { gen3P = millis();
 gen3O = ! gen3O;}}
```

Рисунок 5.11- Фрагмент кода программы запуска «генератора»

Если есть сигнал на вход 52, значит gtv3=1, следовательно swi2=1 (из условия на рисунке 5.10). Команда gen3P = millis() выполняет роль запоминания текущего момента. Переменная gen3I отвечает за состояние, то есть если работает gen3I =1, если не работает то gen3I=0. Запись gen3I=1 означает перевод генератора в рабочее состояние, gen3O = 1 установка выхода «генератора» в 1.

Если «генератор» в рабочем состоянии (запись if (gen3I)), проверяем с помощью функции if (isTimer (gen3P , 5)), наступило ли время переключения - в данном примере 5 миллисекунд, это период переключения. Если время наступило, запоминаем текущее состояние gen3P = millis(). Записью gen3O = ! gen3O инвертируем выход. Также 5 миллисекунд это время одного импульса. «Генератор» отвечает за прирост импульса за каждый период, чтобы посчитать за какое время выходной ШИМ сигнал вырастит с 0 до установленного значения, например до максимального значения 255, необходимо 5 миллисекунд умножить на 255, что получается 1255 миллисекунд или 1,2 секунды. То есть за 1,2 секунды ШИМ изменится от 0 до 255.

Запись кода на рисунке 5.12 представляет собой функцию определения конца периода времени с учётом переполнения аппаратного счетчика (аппаратный счётчик микросекунд считает до 4294967295 а потом сбрасывается в 0).

```
bool isTimer(unsigned long startTime, unsigned long period )
  {
  unsigned long currentTime;
currentTime = millis();
if (currentTime>= startTime) {return (currentTime>=(startTime + period));}
else {return (currentTime >=(4294967295-startTime+period));}
  }
```

Рисунок 5.12- Код определения конца периода времени

На рисунке 5.13 показан алгоритм уменьшения ширины импульса в случае отсутствия сигнала на цифровом входе микроконтроллера. Если нет сигнала включения и текущее значение ширины импульса, записанное в тип данных gtv2 больше или равен единице if (((!(gtv3)) && ((gtv2) >= (1)), значит активируется «генератора» на уменьшение, аналогично как показано на рисунке 5.11.

Для того, что бы каждый «генератор» начинал свою процедуру изменения ширины импульса, необходимо знать, какой из них сейчас работает. За это отвечает код программы на рисунке 5.14. Логически не может работать одновременно два «генератора», каждый должен работать в соответствии наличия входного сигнала. Проверяем какой генератор активен if (((gen3O) ^ (gen1O))), где «^»логическое исключающее «ИЛИ». Если

активна переменная (gtv3=1), содержащая состояние сигнала, то есть сигнал включения поступает на цифровой вход 52, то текущее значение ширины импульса (count2P) увеличить на 1 (count2P = count2P+1), что соответствует написанному коду if (gtv3) {count2P = count2P+1 на рисунке 5.14. Если gtv3=0, то есть сигнала включения не поступает на вход микроконтроллера, уменьшить значение ширины импульса на 1 {count2P = count2P-1;}. Также как и переменные gen3I и gen1I отвечают за режим работы «генератора», ту же функцию выполняет переменная count2I. Если не работает ни один из «генераторов», значит изменять значение ширины импульса не нужно.

```
if (( (!(gtv3)) && ((gtv2) >= (1)) ))
{ if (! gen1I) {
  gen1I = 1;
  gen1O = 1;
  gen1P = millis(); }
} else {
  gen1I = 0 ;
gen1O= 0;}
 if (gen1I) {
  if ( isTimer ( gen1P , 5 ))
 { gen1P = millis();
 gen1O = ! gen1O;}}
```

Рисунок 5.13 - Фрагмент кода на уменьшение ширины импульса

```
if (( (gen3O) ^ (gen1O) ))
   {
   if (! count2I)
      {
        if (gtv3) {count2P = count2P+1;}
        else
        {count2P = count2P-1;};
       count2I = 1;/
      }
   }
else
   {
   count2I=0;
   }
```

Рисунок 5.14 - Фрагмент кода, отвечающего за величину ширины импульса выходного сигнала с микроконтроллера

Счетчик показанный на рисунке 5.12 будет считать импульсы, подаваемые с «генератора» и подавать их на выход непрерывно. Чтобы остановить подачу импульсов на выход в пределах ШИМ сигнала, необходимо отключить «генераторы» и остановить счётчик. Для этого записываются условия (рисунок 5.15)

```
gtv4 =  count2P  >=  255;
gtv2 = count2P;
```

Рисунок 5.15 - Условие остановки нарастания выходного ШИМ сигнала

Когда счётчик начинает считать от 0 до бесконечного значения и когда счёт дойдёт до 255, переменная count2P ловит это значение и подаётся это на переменную gtv2, значение которой меняется с 1 на 0. Это означает остановку «генератора» и счётчика. Таким образом достигается плавный разгон электродвигателя до установленного значения.

Данный алгоритм работы программы представлен в виде блок схемы на листе №2 графической части, для лучшего понимания алгоритма работы программы.

Блок на рисунке 5.16 представляет собой сигнал включения двигателя подаваемый на вход микроконтроллера. Вход показан инвертированным для того, что бы при отсутствии сигнала двигатель не запускался.

Рисунок 5.16 - Блок включения/выключения двигателя

Переменные блок схемы именуемые TT и L> представляют собой переменные состояния gtv3 и gtv4. Блок Switch содержит переменную swi2, работающим следующим образом: если на вход S подана логическая единица то на выход Q подаётся сигнал со входа 0, если на S подан логический ноль, то сигнал на выход подаётся ноль. Блок G-SM представляет собой «генератор», с соответствующими ему переменными. Счётчик CTDU считает импульсы с генератора с 0 до 255.

При подаче сигнала на включения двигателя, мы активируем переменную gtv3 блока TT, которая по умолчанию принимает значение 0. Значение gtv3 становится равным 1. С блока swi2 подаётся 1, что в свою очередь включает блок генератора G-SM и с него, через блок исключающего ИЛИ XOR, и логическая единица, поданная на счётчик CTDU, включает его.

Одновременно с этим сигнал с TT блока подаётся на инвертируемый вход блока And, и на выходе этого блока мы получаем логический ноль. Этот логический ноль подаётся на второй блок генератора. Так как на вход генератора подаётся ноль, он остаётся не активным.

Когда счётчик просчитает до необходимого значения ширины импульса, например 255, будет подан логический ноль на блок L>. Когда

логический ноль с блока L> будет установлен на входе блока Switch, то на выходе этого блока будет подан логический ноль, который остановит генератор, который в свою очередь остановит счётчик. Отсчитанное значение ШИМ, хранимое переменной count2P, с блока с именем «ШИМ на выход», будет подано на выход микроконтроллера.

Когда сигнал не поступает с блока включения/выключения двигателя, подаётся логический ноль на блок TT. Блок And с инвертированным входом принимает с TT логический ноль и с выхода блока And поступает логическая единица, которая активирует блок генератора G-SM, который через блок XOR запускает счётчик. Чтобы счётчик CTDU начал считать обратно к его входу UD подаётся сигнал с TT. Если подан логическая единица, счётчик считает на увеличение, если логический ноль, счётчик считает на уменьшение.

Для ограничения диапазона работы счётчика используется блок сравнения I1>=I2.

Данный код программы позволяет плавно изменять значение выдаваемого с микроконтроллера ШИМ сигнала, с целью плавного разгона двигателя. Данный код не влияет на установившийся режим работы двигателя, а только в момент включения или выключения.

Для коммутации обмотки двигателя и движения ротора в нужном направлении, необходим алгоритм (рисунок 5.18) переключения между драйверами int100s с помощью датчика Холла.

```
bool sensor = digitalRead(holla);
 switch (sensor)
 {
 case 0 :
  analogWrite(in1, 0);
  analogWrite(in2, count2P);
break;

case 1:
  analogWrite(in2, 0);
  analogWrite(in1, count2P);
  break;
```

Рисунок 5.18 - Фрагмент кода коммутации двигателя

Работа с оператором «switch», указанном на рисунке 5.18, описана в главе 5.2. Сигнал с датчика Холла задаётся значением переменной sensor. Если сигнал с датчик Холла ноль, ШИМ сигнал подаётся на первый драйвер через 10 выход микроконтроллера с именем in2, а ШИМ сигнал через 9 выход с именем in1 не подаётся. Если сигнал с датчика Холла равен 1, очередность включения драйверов меняется.

6 Устройство и принцип работы микроконтроллера

6.1 Принцип работы микроконтроллера

Микроконтроллеры серии Arduino имеют основу AVR-семейство восьмибитных микроконтроллеров фирмы Atmel. Главная функция ядра AVR - гарантировать правильное программное выполнение. Контроллер должен выполнять операции, такие как обращение к памяти, выполнение вычислений, управление внешними устройствами, и управление прерываниями.

Для того, чтобы максимизировать выполнение всех функций, AVR пользуется Гарвардской архитектурой (рисунок 6.1) - имеющие разделение программы и данных в разных адресных пространствах. Инструкции в программной памяти выполняются с единственной горизонтальной конвейерной обработкой. Пока одна инструкция выполняется, следующая инструкция предварительно извлекается из программной памяти. Это концепция предоставляет возможность инструкциям выполняться в каждом цикле. Программная память - это встроенная перепрограммируемая флэш-память..

Файл регистра быстрого доступа содержит 32 8-битных регистра, создающих записи с одним временем доступа к циклу. Это позволяет выполнять однотактное арифметическое логическое устройство (АЛУ). В типичной операции АЛУ из файла регистра выводятся два операнда (аргументы операции, данные, обрабатываемые командой), выполняется операция, и результат сохраняется в файле регистра - за один цикл. Шесть из 32 регистров могут использоваться в качестве трех 16-битных указателей для адресации данных - с целью эффективного вычисления. Программная память - это встроенная перепрограммируемая флэш-память. Эти добавленные функциональные регистры представляют собой 16-битные X-, Y- и Z-регистры.

АЛУ поддерживает арифметику и логические действия между регистрами или между константой и регистром. Действия одной операции могут также выполняться в АЛУ. После арифметической операции, статус регистра обновляется, чтобы отразить информацию о результате действия.

Ход выполнения программы обеспечивается условным и безусловным переходами и командами обращения, способных непосредственно обратиться к целому адресному пространству. Большинство инструкций AVR имеют единственный формат 16-битного слова. Каждый адрес программной памяти содержит 16 - или 32-битную инструкцию.

Программное пространство флэш-памяти разделено на две секции, секции программы загрузки и секции прикладной программы. Обе секции имеют выделенные биты блокировки для защиты записи и чтения/записи. Инструкция SPM, которая записывается в секцию прикладной программы, должна находиться в разделе программы загрузки. SPM - аббревиатура, означающая программное обеспечение для управления проектами.

В течение прерываний и вызовов подпрограммы, счетчик команд сохраняется в стеке. Стек эффективно распределяется в общей статической памяти (SRAM), следовательно, размер стека ограничен только общим размером статической памяти и использованием её использованием. SRAM - статическая паять с произвольным доступом, полупроводниковая оперативная память.

Все пользовательские программы должны инициализировать УС в подпрограмме «Сброс» (до выполнения подпрограмм или прерываний). Указатель стека (УС) доступен для чтения / записи в пространстве ввода / вывода. С помощью SRAM можно легко получить доступ через пять различных режимов адресации, поддерживаемых в архитектуре AVR. Пространство памяти в архитектуре AVR представляют собой линейные и регулярные карты памяти.

Модуль гибкого прерывания имеет свои регистры управления в пространстве ввода/вывода с дополнительным битом разрешения глобального прерывания в регистре состояния. Все прерывания имеют отдельный вектор прерывания в таблице векторов прерываний. Прерывания имеют приоритет в соответствии с их позицией вектора прерывания. Чем ниже адрес вектора прерываний, тем выше приоритет. Объем памяти ввода / вывода содержит 64 адреса для периферийных функций AVR, таких как регистры управления, SPI и другие функции ввода-вывода. Доступ к памяти ввода / вывода можно получить напрямую или в качестве местоположения пространства данных, следующего за файлом регистра. Кроме того, это устройство имеет расширенное пространство ввода / вывода в SRAM, где могут использоваться только инструкции ST / STS / STD и LD / LDS / LDD.

Высокопроизводительный АЛУ AVR работает в прямом соединении со всеми 32 рабочими регистрами общего назначения. В течении одного цикла, выполняются арифметические операции между регистрами общего назначения или между регистрами и непосредственного выполнения. Операции АЛУ делятся на три главных категории: арифметические, логические, и битовые. Некоторые реализации архитектуры также обеспечивают мощный множитель, поддерживающий как знаковое/бесзнаковое умножение, так и дробный формат.

Регистр состояния содержит информацию о результате самой последней выполняемой арифметической команды. Эта информация может быть использована для изменения хода выполнения программы для того, чтобы выполнять условные действия. Регистр состояния обновляется в конце операции АЛУ. Это во многих случаях устранит необходимость в использования специальных команд сравнения, что приведет к более быстрому и компактному коду. Регистр состояния не сохраняется автоматически при вводе подпрограммы прерывания и восстанавливается при возврате из прерывания. Это должно быть обработано программным обеспечением.

Регистровый файл оптимизирован под AVR. Для достижения требуемого быстродействия и гибкости, используются методы , поддерживаемые регистровым файлом:

- Один 8-битный выходной операнд и один 8-битный входной результат;
- Два 8-битных выходных операнда и один 8-битный входной результат;
- Два 8-битных выходных операнда и один 16-битный входной результат;
- Один 16-битный выходной операнд и один 16-битный входной результат.

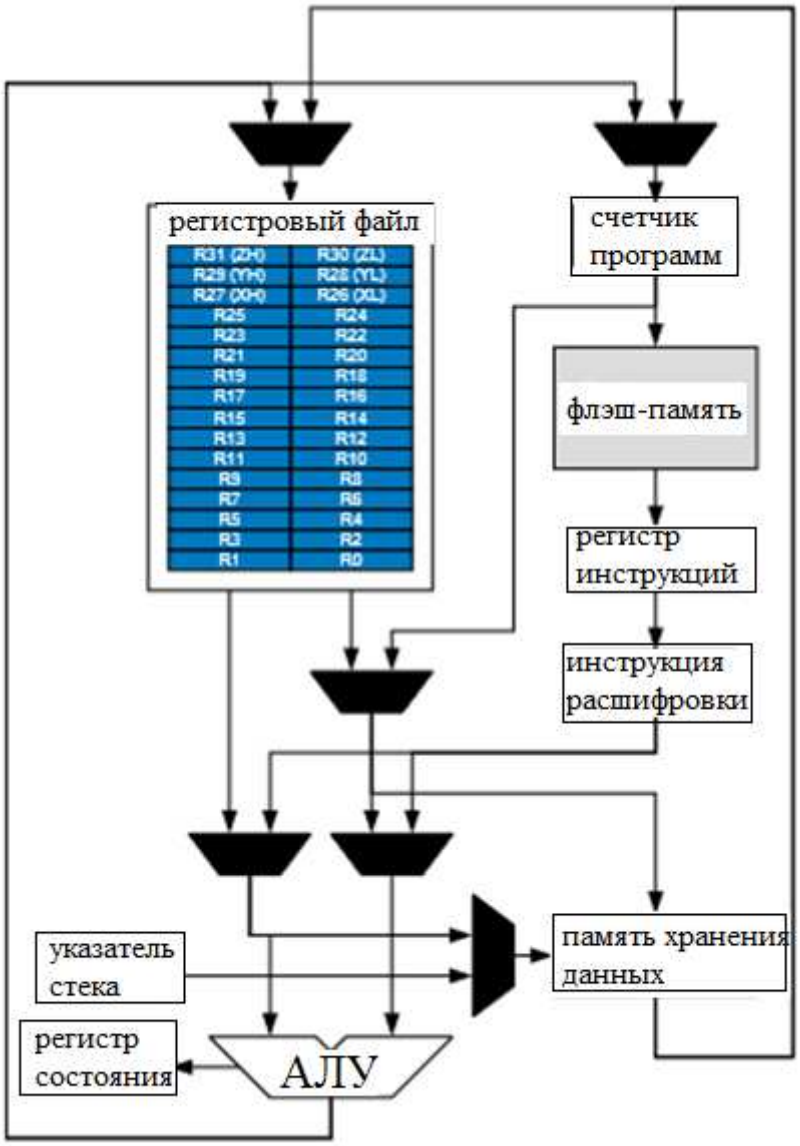


Рисунок 6.1 - Функциональная схема архитектуры AVR

Большинство из инструкций, работающих с регистровым файлом, имеют прямой доступ ко всем записям, и большинство из них являются единственными инструкциями цикла. Каждой записи также присвоен адрес памяти данных, отображающий их непосредственно в первые 32 места пользовательского пространства данных. И хотя физически они не располагаются в SRAM, эта организация памяти обеспечивает большую гибкость в доступе к регистрам, и X-, Y-, и Z-указатели регистра могут

использоваться, чтобы индексировать любой регистр в файле. X-, Y-, и Z-регистры, представляющие собой 16-битовые адресные указатели для косвенной адресации пространства данных.

Стек главным образом используется для хранения временных данных, для хранения местных переменных и для хранения адресов возврата после прерываний и вызовов подпрограммы. Реализация стек осуществляется от более высоких к более низким местам памяти. Указатель стека указывает на область SRAM данных, в которой расположены подпрограммы и стеки прерываний. Команда Stack PUSH уменьшит указатель стека. Стек в SRAM данных должен быть определен программой перед выполнением любых вызовов подпрограмм или разрешенных прерываний. Начальное значение указателя стека должно быть равно последнему адресу внутренней SRAM и должен быть установлен в положение, указанное выше начала SRAM.

Процессор AVR управляется тактовой частотой процессора clkCPU, непосредственно генерируемой из выбранного источника синхронизации для чипа. Нет внутреннего разделения таймера. На приведенном ниже рисунке 6.2 показаны выборки и инструкции параллельных команд, разрешенные архитектурой Гарварда и концепцией регистрового файла с быстрым доступом. Это базовая концепция конвейерной обработки для получения до 1 MIPS на МГц (MIPS - единица вычислительной скорости, эквивалентная миллиону команд в секунду).

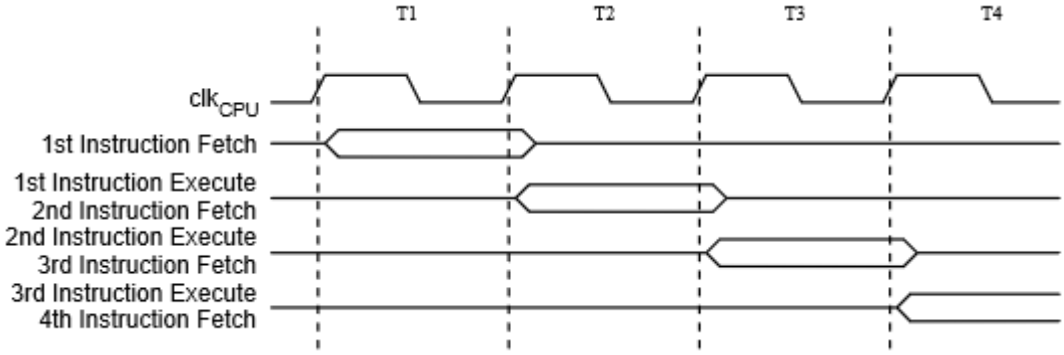


Рисунок 6.2 - Инструкции параллельных команд

На следующем рисунке 6.3 показана концепция внутреннего таймера для регистрового файла. За один такт выполняется операция АЛУ с использованием двух регистровых операндов, результат сохраняется в реестре назначения.

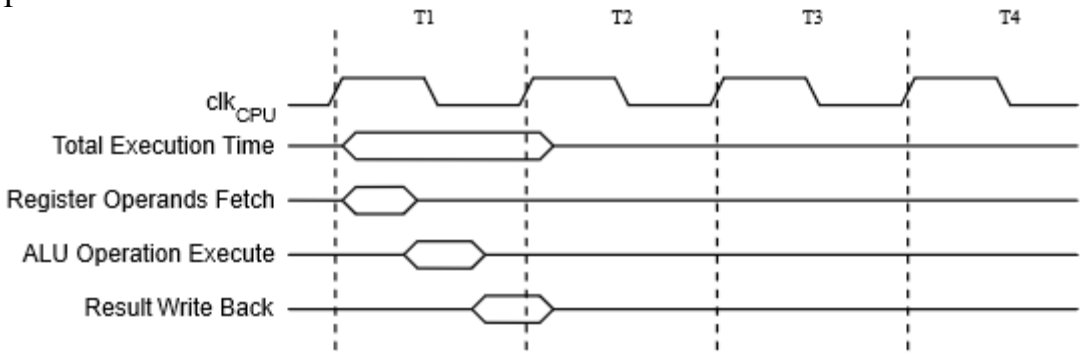


Рисунок 6.3 - Один цикл выполнения команды АЛУ

6.2 Сброс и обработка прерываний

AVR предоставляет несколько разных источников прерываний. Эти прерывания и вектора прерываний имеют отдельный программный вектор в программной памяти. Все прерывания назначаются индивидуальными битами включения, которые должны быть записаны логически вместе с битом разрешения глобального прерывания в регистре состояния, чтобы включить прерывание. В зависимости от значения счетчика программ прерывания могут быть автоматически отключены, если запрограммированы биты блокировки загрузки BLB02 или BLB12. Эта функция улучшает безопасность программного обеспечения.

Самые низкие адреса в программной памяти по умолчанию определяются как сброс и вектора прерываний. Они определяют уровни приоритета: чем ниже адрес, тем выше уровень приоритета.

Сброс имеет наивысший приоритет, а следующий - INT0 - запрос внешнего прерывания 0. Вектора прерываний могут быть перенесены в начало загрузочного сектора, установив бит IVSEL в регистр управления/контроля MCU. Вектор сброса также можно перенести в начало загрузочного сектора, с помощью бита BOOTRST.

Когда происходит прерывание, 1-бит глобального прерывания очищается, и все прерывания отключены. Пользовательское программное обеспечение может записывать логический код в 1-бит, чтобы включить вложенные прерывания. Все разрешенные прерывания могут останавливать текущую процедуру прерывания. 1-бит автоматически устанавливается, когда выполняется команда выход из прерывания - RETI.

Существуют в основном два типа прерываний: первый тип запускается событием, которое устанавливает флаг прерывания. Для этих прерываний программный счетчик векторизован к фактическому вектору прерываний для выполнения процедуры обработки прерываний, а аппаратное обеспечение очищает соответствующий флаг прерывания. Флаги прерываний также можно очистить, записав логический бит в позицию бит - флага, подлежащую очистке. Если условие прерывания происходит, когда бит разрешения прерывания очищается, флаг прерывания будет установлен и сохранен до тех пор, пока прерывание не будет включено, или флаг не будет очищен программным обеспечением. Аналогично, если выполняется одно или несколько условий прерывания, когда бит глобального разрешения прерываний очищается, соответствующий флаг прерывания будет установлен и сохранен до тех пор, пока бит глобального разрешения прерывания не будет установлен, и затем будет выполняться по порядку приоритета.

Второй тип прерываний будет срабатывать до тех пор, пока присутствует условие прерывания. Этим прерываниям не обязательно иметь флаги прерывания. Если условие прерывания исчезает до того, как прерывание включено, прерывание не будет запущено. Когда AVR выходит из прерывания, он всегда возвращается к основной программе и выполняет еще одну команду до того, как будет обслуживаться какое-либо ожидающее прерывание.

Регистр состояния не сохраняется автоматически при вводе подпрограммы прерывания и не восстанавливается при возврате из процедуры прерывания. Это должно быть обработано программным обеспечением. При использовании команда CLI для отключения прерываний прерывания будут немедленно отключены. Команда CLI - относится к классу инструкций ввода/вывода, она сбрасывает флаги прерывания. После команды CLI все последующие прерывания не будут выполнятся, даже если оно происходит одновременно с инструкцией CLI.

Ответ на выполнение прерывания для всех разрешенных прерываний AVR равен четырем тактам. После четырех тактовых циклов выполняется программный адрес вектора для действительной процедуры обработки прерываний. Во время этого четырех тактового цикла счетчик программ помещается в стек. Обычно вектор представляет собой переход к процедуре прерывания, и этот переход занимает три тактовых цикла. Если прерывание происходит во время выполнения команды с несколькими циклами, эта команда завершается до того, как прерывание будет обслуживаться. Если прерывание происходит, когда регистр контроля MCU находится в спящем режиме, время отклика на выполнение прерывания увеличивается на четыре тактовых цикла. Это увеличение происходит в дополнение к времени запуска из выбранного спящего режима. Возврат из процедуры обработки прерываний занимает четыре тактовых цикла. Во время этих четырех тактовых циклов счетчик программ (два байта) извлекается из стека, указатель стека увеличивается на два, а 1-бит глобального прерывания в устанавливается SREG - регистр состояния.

6.3 Память AVR

Архитектура AVR имеет два основных сектора памяти: память данных Data Memory и программную память Program Memory. Все сектора памяти являются линейными и регулярными.

ATmega328 содержит 32-килобайтную встроенную перезагружаемую флэш-память для хранения программ. Поскольку все инструкции AVR имеют разрешение 16 или 32 бит, флэш-память имеет размерность 16Kx16. Для безопасности программного обеспечения пространство флэш-памяти разделено на две секции, секции программы загрузки и секции прикладной программы. Флэш-память имеет выдержку не менее 10 000 циклов записи / стирания. Счетчик программ ATmega328 имеет разрешение 14 бит, таким

образом обращаясь к ячейкам памяти 16K. Работа секции программы загрузки и программа блокировки загрузки связанны одними битами для защиты программного обеспечения.

6.4 Системные часы

На рисунке 6.4 показаны основные системы часов в устройстве и их распределение. Все часы не должны быть активны в один момент времени. Чтобы уменьшить потребление энергии, часы для не используемых модулей могут быть остановлены с использованием разных режимов сна. Все тактовые выходы блока управления часов AVR работают с одинаковой частотой.

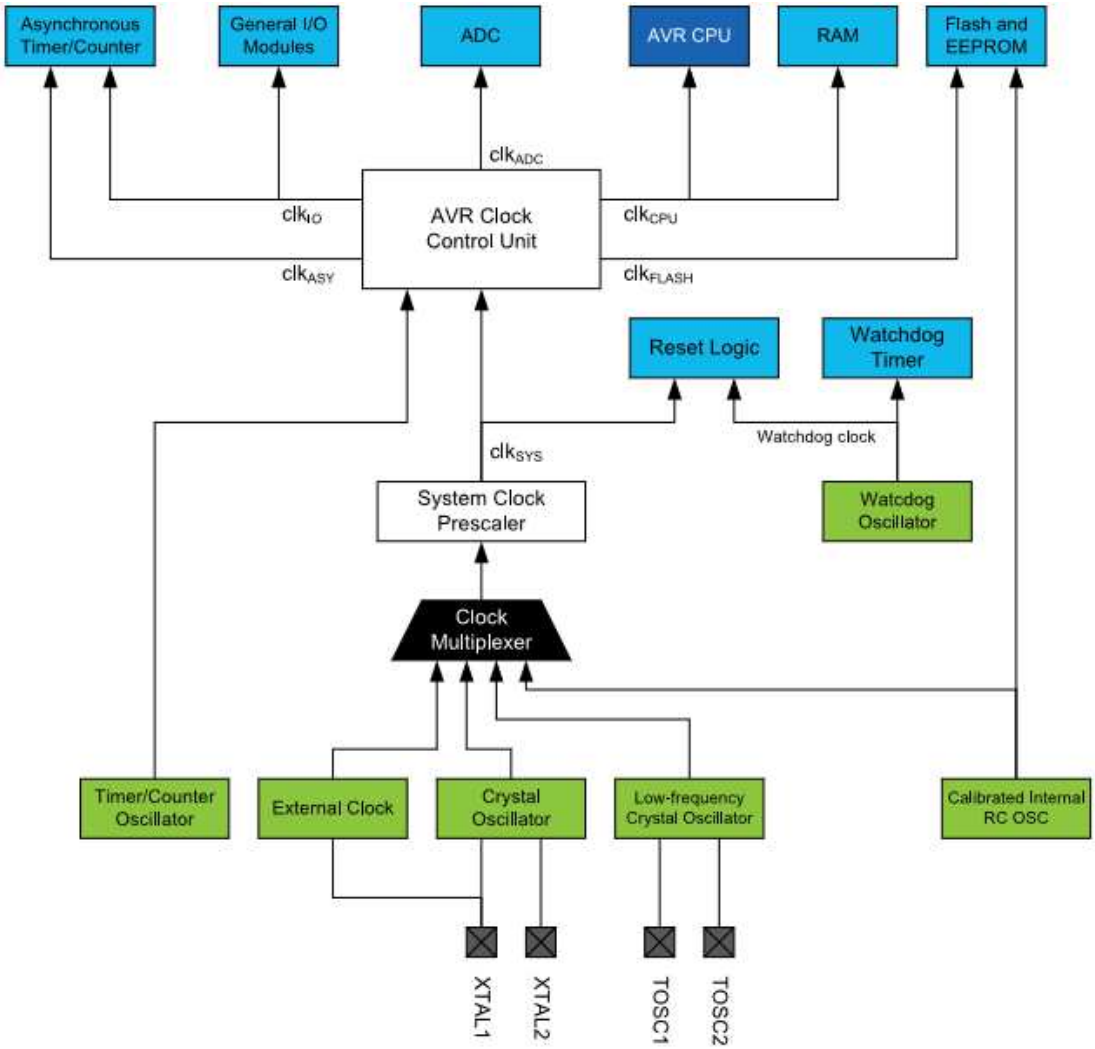


Рисунок 6.4 - Распределение часов

Центральный процессор направлен в части системы, связанные с работой ядра AVR. Примерами таких модулей являются файл регистра общего назначения, регистр состояния и память данных, содержащая

указатель стека. Прекращение тактовых импульсов центрального процессора блокирует ядро от выполнения общих операций и вычислений.

Часы ввода/вывода clkI/O используются большинством модулей ввода/вывода, таких как таймер/счетчики, SPI и USART. Часы ввода-вывода также используются модулем внешнего прерывания, но обнаружение состояния запуска в модуле USI выполняется асинхронно, когда clkI/O останавливается. Если для пробуждения от отключения питания используется прерывание с уровнем, требуемый уровень должен быть достаточно длительным, чтобы регистр контроля MCU завершил пробуждение, чтобы вызвать прерывание уровня. Если уровень исчезнет до истечения времени запуска, MCU все равно включится, но прерывание не будет сгенерировано.

Асинхронный таймер - clkASY позволяют синхронизировать асинхронные таймеры/счетчики непосредственно с внешнего тактового генератора или внешнего тактового кристалла с частотой 32 кГц. Специальный тактовый домен позволяет использовать этот таймер/счетчик в качестве счетчика в реальном времени, даже когда устройство находится в спящем режиме.

Часы АЦП снабжены выделенной тактовой областью. Это позволяет заблокировать часы центрального процессора, чтобы уменьшить шум, создаваемый цифровой схемой. Это дает более точные результаты преобразования АЦП.

Устройство имеет внутренний RC-генератор с частотой 8,0 МГц и с запрограммированным предохранителем CKDIV8, что приводит к системным часам с частотой 1,0 МГц. Время запуска устанавливается на максимум, и период тайм-аута включен: CKSEL = 0010, SUT = 10, CKDIV8 = 0. Этот параметр по умолчанию гарантирует, что все пользователи смогут настроить желаемый источник синхронизации с помощью любого доступного интерфейса программирования.

Любой источник синхронизации нуждается в достаточном напряжении питания, чтобы начать генерирование колебаний и генерирование минимального количества циклов, прежде чем его можно считать стабильным. Для обеспечения достаточного напряжении питания устройство выдает внутренний сброс с задержкой тайм-аута (tTOUT) после сброса устройства всеми другими источниками сброса. Задержка (tTOUT) синхронизируется с генератором сторожевого таймера, а количество циклов задержки устанавливается битами SUTx и CKSELx. Частота генератора сторожевого таймера зависит от напряжения.

Основная цель задержки - сохранить устройство в режиме сброса, пока оно не будет снабжено минимальным напряжением питания. Задержка не будет контролировать фактическое напряжение, поэтому требуется выбрать задержку дольше, чем время нарастания напряжением питания. Если это невозможно, используется внутренняя или внешняя схема ограничения доступа электроэнергии - Brown-Out Detection. Цепь BOD обеспечит достаточное напряжение питания до сброса, и можно отключить задержку

тайм-аута. Отключение задержки тайм-аута без использования схемы ограничения доступа электроэнергии не рекомендуется.

Генератор должен выдавать колебания в течение минимального количества циклов, чтобы считать его стабильным для работы. Внутренний счетчик пульсаций контролирует выход генератора и сохраняет внутренний сброс активным для заданного количества тактовых циклов. Затем сброс отпускается, и устройство начинает выполнение. Рекомендуемое время запуска генератора зависит от типа тактового сигнала и варьируется от 6 циклов для внешних тактовых импульсов до 32K циклов для низкочастотного кристалла.

Последовательность запуска для часов включает в себя как временную задержку, так и время запуска, когда устройство запускается с момента сброса. При запуске из режима энергосбережения или отключения питания предполагается, что напряжение питания находится на достаточном уровне и включается только время запуска.

Выводы XTAL1 и XTAL2 являются входными и выходными сигналами инвертирующего усилителя, который может быть сконфигурирован для использования в качестве генератора, как показано на рисунке ниже. Можно использовать либо кварцевый кристалл, либо керамический резонатор.

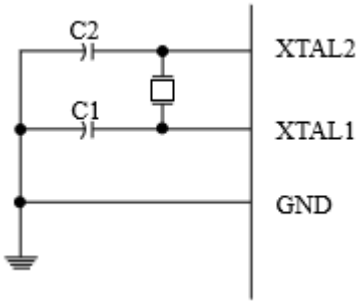


Рисунок 6.5 - Схема кристального генератора

C1 и C2 всегда должны быть равны для обоих кристаллов и резонаторов. Оптимальное значение конденсаторов зависит от используемого кристалла или резонатора, количества паразитной емкости и электромагнитного шума окружающей среды. Для керамических резонаторов следует использовать значения конденсатора, указанные производителем.

Таймер/счетчик генератора использует тот же кварцевый генератор для низкочастотного генератора и генератора таймера/счетчика. На этом устройстве контакты генератора таймера/счетчика (TOSC1 и TOSC2) совместно используются с XTAL1 и XTAL2. При использовании генератора таймера/ счетчика частота системных часов должна быть в четыре раза больше частоты генератора. В связи с этим и совместным использованием контактов генератор таймера/счетчика может использоваться только тогда, когда в качестве источника системных часов выбран калиброванный внутренний RC-генератор. Применение внешнего источника синхронизации к

TOSC1 может быть выполнено, если бит входа «Включить внешние часы» в регистре асинхронного состояния (ASSR.EXCLK) записывается в «1».

Устройство может выводить системные часы на вывод CLKO. Чтобы включить выход, необходимо запрограммировать бит CKOUT. Этот режим подходит, когда часы микросхемы используются для управления другими цепями в системе. Часы также будут выводиться во время сброса, и нормальная работа вывода будет отменена при программировании. Любой источник синхронизации, включая внутренний RC-генератор, можно выбрать, когда часы выводятся на CLKO.

Устройство имеет предварительный делитель системных часов, а системные часы можно разделить, настроив регистр Prescale Clock (CLKPR). Эта функция может использоваться для уменьшения тактовой частоты системы и энергопотребления, когда требование к мощности обработки низкое. Это можно использовать со всеми часами источника, и это повлияет на тактовую частоту процессора и всех синхронных периферийных устройств. При переключении между настройками предварительного делителя система Prescaler System Clock обеспечивает отсутствие сбоев в системе часов. Это также гарантирует, что никакая промежуточная частота не будет больше, чем частота, соответствующая предыдущей настройке, ни тактовая частота, соответствующая новой настройке. Счетчик пульсаций, который реализует предделитель, работает на частоте нераздельных часов, что может быть быстрее, чем частота процессора. Следовательно, невозможно определить состояние предделителя - даже если бы оно было читаемым, точное предсказание времени, которое требуется для переключения с одного такта на другое, невозможно точно предсказать. В этот момент создаются два активных интервала времени. Время T1 - это предыдущий период времени, а T2 - период, соответствующий новой настройке предварительного делителя. Чтобы избежать непреднамеренных изменений тактовой частоты, необходимо выполнить специальную процедуру записи, чтобы изменить бит CLKPS:

1. Записать бит Prescaler Change Enable (CLKPCE) в значение «1» и все остальные бит CLKPR в ноль.
2. В течение четырех циклов написать нужное значение CLKPS [3: 0] при записи нуля в CLKPCE.

Прерывания должны быть отключены при изменении настроек предделителя, чтобы убедиться, что процедура записи не прерывается.

6.5 Система контроля и сброса (SCRT)

Во время сброса все регистры ввода/вывода устанавливаются на их начальные значения, и программа запускает выполнение из вектора сброса. Инструкция, размещенная в векторе сброса, должна быть инструкцией абсолютного перехода (JMP) для процедуры обработки сброса. Если программа никогда не включает источник прерывания, векторы прерываний

не используются, и в этих местах может быть размещен обычный программный код. Это также имеет место, если вектор сброса находится в секции приложения, в то время как векторы прерываний находятся в разделе загрузки или наоборот. Порты ввода-вывода AVR немедленно возвращаются в исходное состояние, когда источник сброса активируется. Для этого не требуется запуск любого источника синхронизации. После того, как все источники сброса неактивны, вызывается счетчик задержки, растягивая внутренний сброс. Это позволяет достичь стабильного уровня мощности до начала нормальной работы. Период тайм-аута счетчика задержки определяется пользователем через биты SUT и CKSEL.

6.6 Сторожевой таймер

Если сторожевой таймер не используется в программе, модуль должен быть отключен. Если сторожевой таймер включен, он будет включен во всех режимах сна и, следовательно, всегда будет потреблять электроэнергию. В режимах более глубокого сна это значительно повлияет на общее потребление тока.

Устройство имеет расширенный сторожевой таймер (Wachdog Prescaler). Wachdog Prescaler - это циклы подсчета таймера отдельного встроенного генератора 128 кГц. Он дает прерывание или системный сброс, когда счетчик достигает заданного значения тайм-аута. В нормальном режиме работы требуется, чтобы система использовала инструкцию сброса сторожевого таймера (Watchdog Timer Reset) для перезапуска счетчика до достижения значения тайм-аута. Если система не перезапускает счетчик, выдается прерывание или сброс системы.

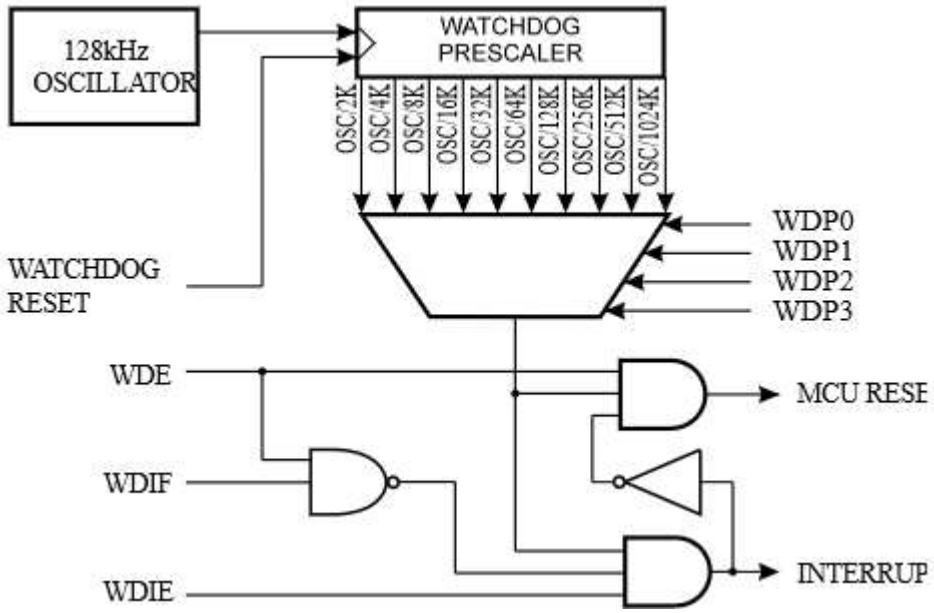


Рисунок 6.6- Функциональная схема работы сторожевого таймера

В режиме прерывания таймер дает прерывание по истечении времени. Это прерывание можно использовать для пробуждения устройства из режимов ожидания, а также в качестве общего системного таймера. Одним из примеров является ограничение максимального времени, разрешенного для определенных операций, что дает прерывание, когда операция работает дольше, чем ожидалось. В режиме сброса системы сторожевой таймер дает сброс, когда истекает время таймера. Обычно это используется для предотвращения зависания системы в случае пропуска выполнения кода. Третий режим, режим прерывания и сброса системы объединяет два других режима, сначала передавая прерывание, а затем переключается в режим сброса системы. Этот режим, например, позволит безопасное выключение, сохраняя критические параметры перед сбросом системы.

Предохранитель таймера всегда включен (WDTON), и если он запрограммирован, заставит сторожевой таймер переключиться в режим сброса системы. С запрограммированным предохранителем бит режима сброса (WDE) и бит прерывания (WDIE) блокируются на 1 и 0 соответственно. Чтобы обеспечить безопасность программы, изменения в настройке сторожевого таймера должны выполняться последовательностями по времени.

6.7 Порты ввода/вывода

Все порты AVR имеют истинную функцию Read-Modify-Write при использовании в качестве общих цифровых портов ввода-вывода. Это означает, что направление одного порта может быть изменено без непреднамеренного изменения направления любого другого порта с инструкциями BOO и CBI. То же самое происходит при изменении значения с LOW на HIGH (если оно настроено как выход) или включение/выключение подтягивающих резисторов (если сконфигурировано как вход). Драйвер ввода достаточно прочный, чтобы напрямую управлять светодиодными дисплеями. Все выводы портов имеют индивидуально выбираемые подтягивающие резисторы с сопротивлением напряжения питания. Все выводы имеют защитные диоды как для напряжения питания, так и для заземления, как показано на следующем рисунке.

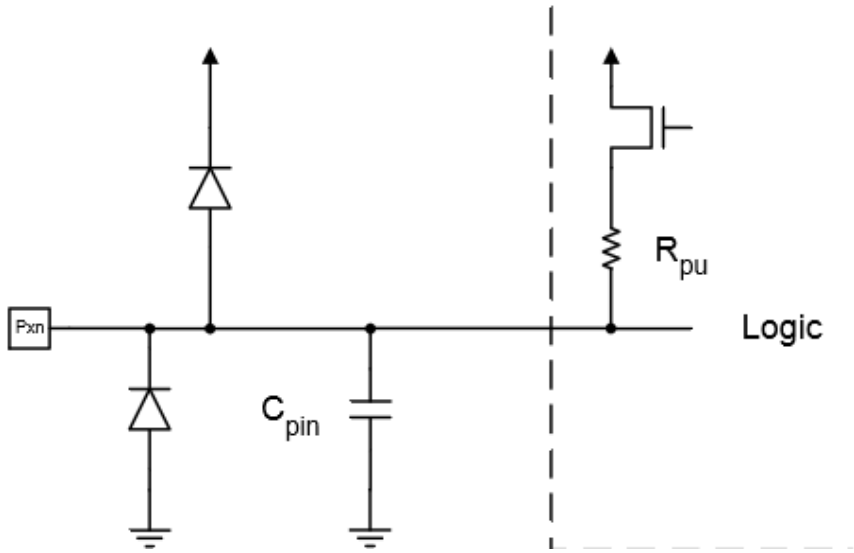


Рисунок 6.7 - Схема ввода/вывода

Каждый порт микроконтроллера представляют собой двунаправленные порты ввода-вывода с дополнительными внутренними подтягивающими резисторами. На предыдущем рисунке показано функциональное описание одного порта, названный Pxn.

Каждый порт состоит из трех битов регистров: DDxn, PORTxn и PINxn. К битам DDxn обращаются по адресу DDRx, битам PORTxn на адресе ввода-вывода PORTx и битам PINxn на адресе ввода-вывода PINx. Бит DDxn в регистре DDRx выбирает направление этого вывода. Если DDxn записывается в «1», Pxn конфигурируется как выходной вывод. Если DDxn записывается в «0», Pxn конфигурируется как входной вывод. Если PORTxn записывается в «1», когда контакт сконфигурирован как входной контакт, активируется нагрузочный резистор. Чтобы отключить подтягивающий резистор, PORTxn должен быть записан в «0», или контакт должен быть сконфигурирован как выходной вывод. Порты трижды указываются, когда условие сброса становится активным, даже если таймеры не работают. Если PORTxn записывается в «1», когда порт сконфигурирован как выходной вывод, присваивается значение высокого уровня. Если PORTxn записывает логическую нуль, когда контакт сконфигурирован как выходной вывод, присваивается значение низкого уровня.

Если некоторые контакты не используются, рекомендуется убедиться, что эти контакты имеют определенный уровень. Несмотря на то, что большинство цифровых входов отключены в режимах глубокого сна, следует избегать плавающих входов, чтобы уменьшить потребление тока во всех других режимах, где цифровые входы активированы (сброс, активный режим и режим ожидания). Самый простой способ обеспечить определенный уровень неиспользуемого вывода - это включить внутреннее подтягивание. В этом случае подтягивание будет отключено во время сброса. Если важна низкая потребляемая мощность во время сброса, рекомендуется использовать

внешнее подтягивание. Не рекомендуется подключать неиспользуемые контакты непосредственно к источнику питания или заземлению, так как это может привести к чрезмерным токам, если контакт случайно сконфигурирован как выход.

6.8 EXINT - внешние прерывания

Внешние прерывания запускаются с помощью контактов INT или любого из контактов PCINT. Если включено, прерывания будут срабатывать, даже если контакты INT или PCINT настроены как выходы. Эта функция обеспечивает способ программного прерывания. Запрос прерывания будет срабатывать, если какой-либо активированный контакт переключится. Запрос прерывания смены будет срабатывать, если активирован любой контакт.

Это означает, что эти прерывания могут использоваться для пробуждения также из режимов сна, кроме режима ожидания.

Внешние прерывания могут быть вызваны спадающим или нарастающим фронтом или низким уровнем. Это настроено так, как указано в спецификации для регистра управления внешними прерываниями A (EICRA). Когда внешние прерывания включены и сконфигурированы как срабатывание уровня, прерывания будут срабатывать до тех пор, пока контакт будет удерживаться на низком уровне. Прерывание низкого уровня в INT определяется асинхронно. Это означает, что это прерывание можно использовать для пробуждения части также из режимов сна, кроме режима ожидания. Таймер ввода/вывода останавливаются во всех режимах сна, кроме режима ожидания. Если для пробуждения от отключения питания используется прерывание с уровнем, требуемый уровень должен быть достаточно длительным, чтобы регистр контроля MCU завершил пробуждение, чтобы вызвать прерывание уровня. Если уровень исчезнет до истечения времени запуска, MCU все равно проснется, но прерывание не будет сгенерировано.

6.9 TC0 - 8-битный таймер / счетчик с ШИМ

ШИМ на Arduino целиком и полностью реализован благодаря встроенным таймерам - и у Arduino Nano их 2 - два 8-битных (Timer0 и Timer2). Каждый таймер управляется несколькими регистрами. Основные регистры управления - TCCRnA и TCCRnB. Ниже представлены биты, принадлежащие этим регистрам, которые нам понадобятся в рамках данной статьи:

CSn[2:0] (Clock Select) — биты регистра TCCRnB отвечающие за выбор источника синхронизации, предделителя и режима работы таймера.

WGMn[2:0] (Waveform Generation Mode) — выбор режима генерации волны. Биты расположены в регистрах TCCRnA и TCCRnB

COMnA[1:0] и COMnB[1:0] — режим совпадения вывода A/B — включен/выключен/инвертирован (биты расположены в регистре TCCRnA).

Timer/Counter0 (TC0) - это 8-битный модуль таймера/счетчика общего назначения с двумя независимыми модулями сравнения выходных данных и поддержкой ШИМ. Он позволяет точно определять сроки выполнения программы (управление событиями) и генерировать ШИМ. Ниже приведена упрощенная блок-схема 8-битный таймера/счетчика. TC0 активируется путем записи бита PRTIM0 на «0». TC0 активируется, когда бит PRTIM0 в регистре снижения мощности (PRR.PRTIM0) записывается в «1».

Регистр таймера/счетчика (TCNTn) и регистр выходных данных TC0x (OCRnx) являются 8-битными. Сигналы запроса прерывания (сокращенно Int.Req. на рисунке) видимы в регистре флагов прерывания таймера 0 (TIFRn). Все прерывания индивидуально маркируются с помощью регистра маски прерывания таймера (TIMSKn). TIFR0 и TIMSK0 не показаны на рисунке. Таймер/счетчик может быть синхронизирован внутри, через предварительный делитель (Prescaler) или внешним источником синхронизации на выводе Tn. Логический блок «Выбор часов» (Clock Select) определяет, какой источник синхронизации будет использоваться таймером/счетчиком для увеличения (или уменьшения) его значения. Таймер/счетчик неактивен, если не выбран источник синхронизации. Выход из логики выбора часов называется часовым таймером (clkT0). Регистры сравнения с двойным буфером (OCRnA и OCRnB) всегда сравниваются со значением таймера/счетчика. Результат сравнения может быть использован генератором сигналов (Waveform Generation) для генерации ШИМ или выход переменной частоты на выводах сравнения вывода (OCnA и OCnB).

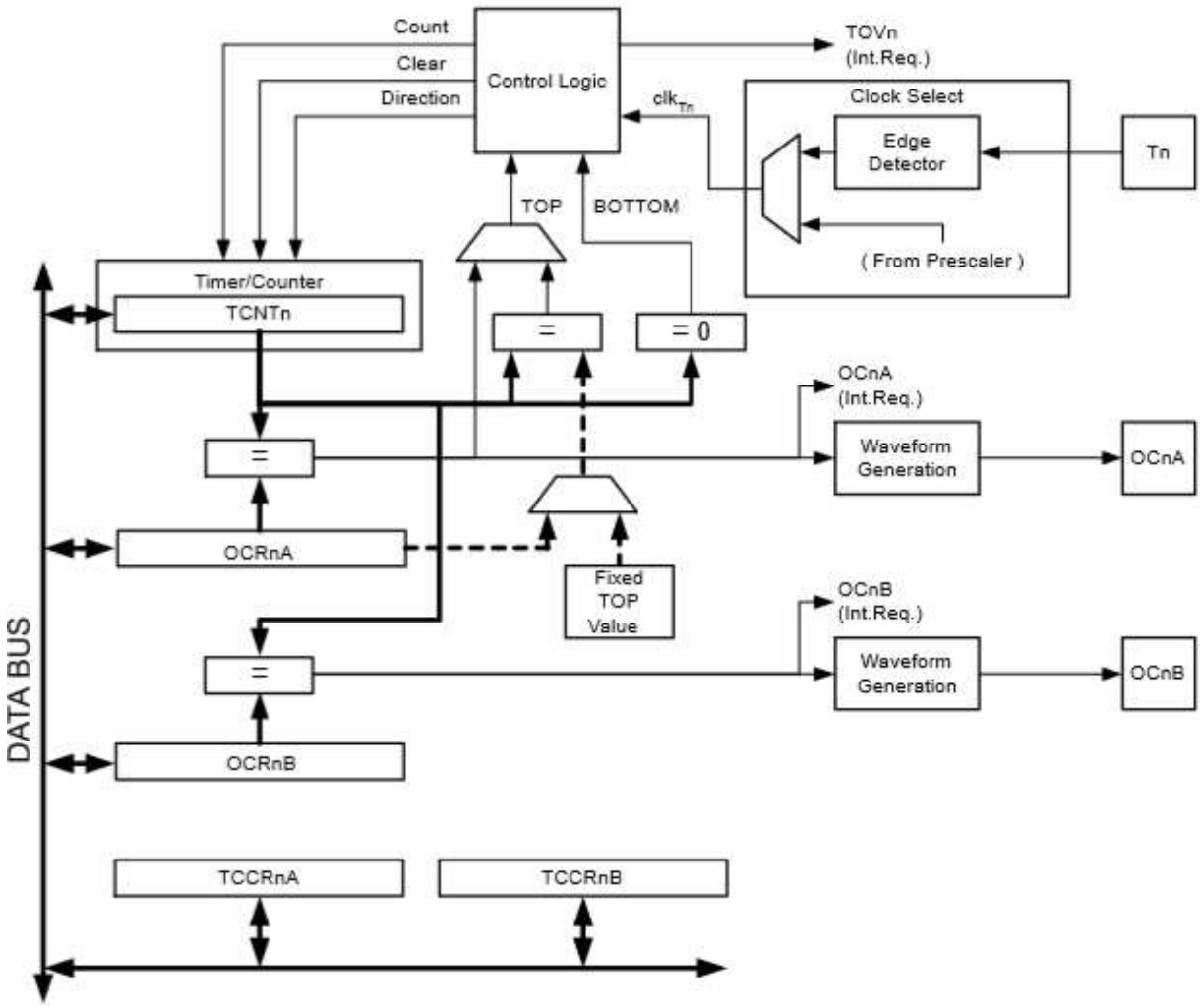


Рисунок 6.8 - Блок схема 8-битного таймера/счетчика

Основная часть 8-битного таймера/счетчика - это программируемый двунаправленный счетчик. Ниже приведена блок-схема счетчика и его компонентов. Счетный блок таймера - это программируемый двунаправленный счетчик. Ниже приведена блок-схема счетчика и его окружения. 8-битный компаратор непрерывно сравнивает TCNTn с регистрами сравнения выходных данных (OCRnA и OCRnB). Всякий раз, когда TCNTn равен OCRnA или OCRnB, компаратор сигнализирует о совпадении. Совпадение установит флажок сравнения (OCFnA или OCFnB) в следующем цикле таймера. Если соответствующее прерывание включено, флаг сравнения выходных данных генерирует прерывание сравнения выходных данных. Флаг сравнения автоматически очищается при выполнении прерывания. В качестве альтернативы, флаг может быть очищен программным обеспечением, написав «1» в его местоположении бит ввода-вывода. Генератор сигналов использует сигнал соответствия для генерации ШИМ на вфводе в соответствии с режимом работы, установленным битами WGM02, WGM01 и WGM00, и битами компаратора. Максимальные (top) и минимальные (bottom) сигналы используются генератором сигналов для

обработки особых случаев экстремальных значений в некоторых режимах работы.

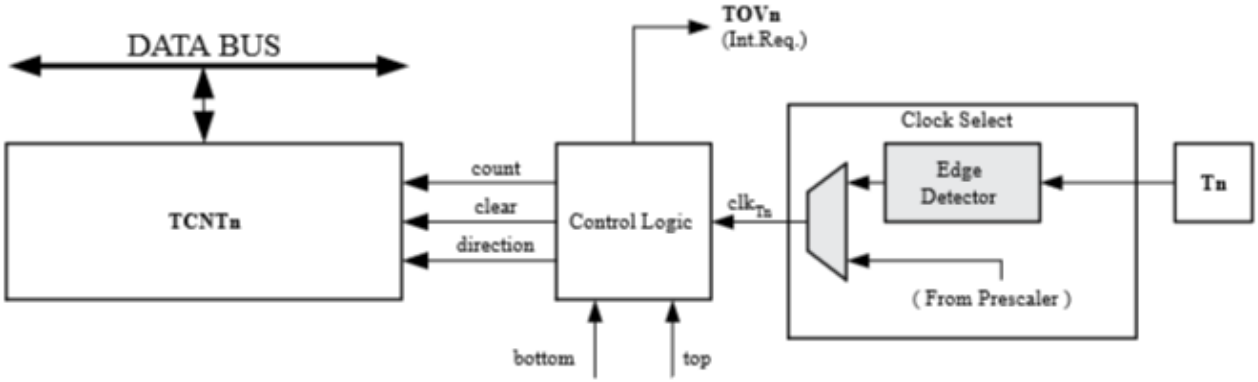


Рисунок 6.9 - Блок схема счетчика.

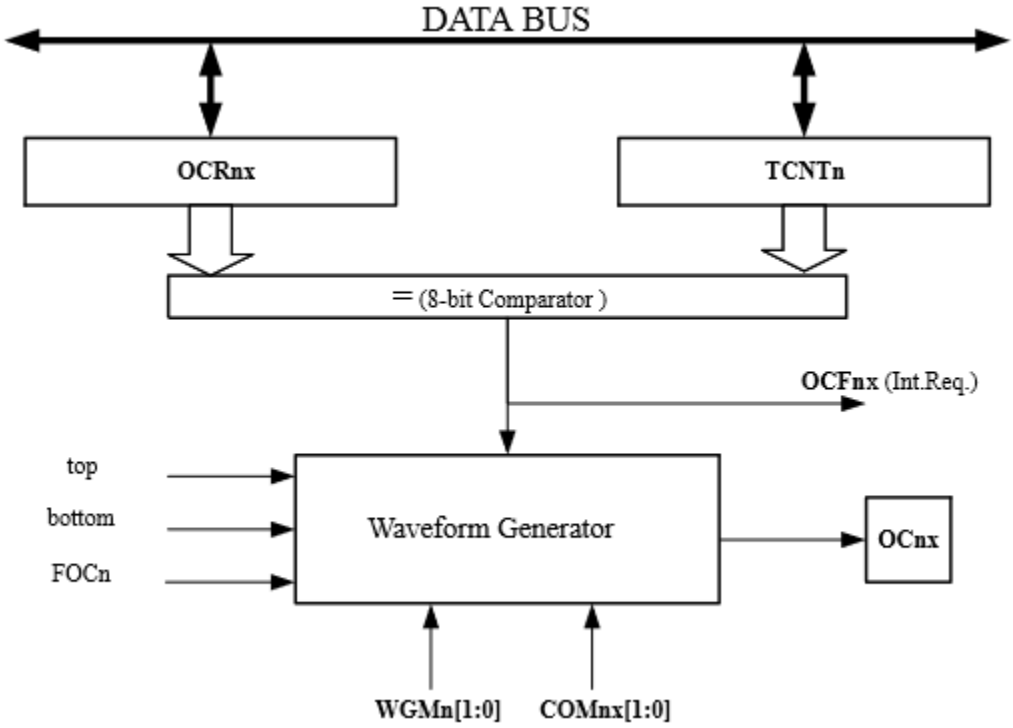


Рисунок 6.10- Блок схема 8-битного компаратора.

Регистры OCR0x имеют двойной буфер при использовании любого из режимов ШИМ. Когда включена двойная буферизация, процессор имеет доступ к регистру буфера OCR0x, отвечающего за синхронизацию. Синхронизация предотвращает появление несимметричных импульсов ШИМ нечетной длины, тем самым делая выход без помех. Двойная буферизация отключена для режима нормальной работы, а центральный процессор будет напрямую обращаться к регистру OCR0x.

Все операции центрального процессора в регистре TCNTn будут блокировать любое сравнение, которое встречается в следующем цикле

таймера, даже когда таймер остановлен. Эта функция позволяет инициализировать OCR1x с тем же значением, что и TCNTn, без срабатывания прерывания при включении таймера /счетчика.

Поскольку запись TCNT1 в любом режиме работы будет блокировать все сравнения совпадений для одного тактового таймера, существуют риски, связанные с изменением TCNT1 при использовании модуля сравнения выходных данных независимо от того, работает ли таймер / счетчик или нет. Если значение, записанное в TCNT1, равно значению OCR1x, сравнение будет пропущено, что приведет к неправильной генерации сигнала.

Биты в регистре контроля А таймера/счетчика (TCCR0A.COM0x) имеют две функции:

1)Генератор сигналов использует биты COMnx для определения состояния регистра выходных данных (OCnx) при следующем сопоставлении сравнения;

2)Биты COMnx управляют выходным источником вывода OCnx.

На приведенном ниже рисунке 6.11 показана упрощенная схема логики, затронутой COMnx. Отображаются только части общих регистров управления портами ввода-вывода, на которые влияют биты COMтx, а именно PORT и DDR.

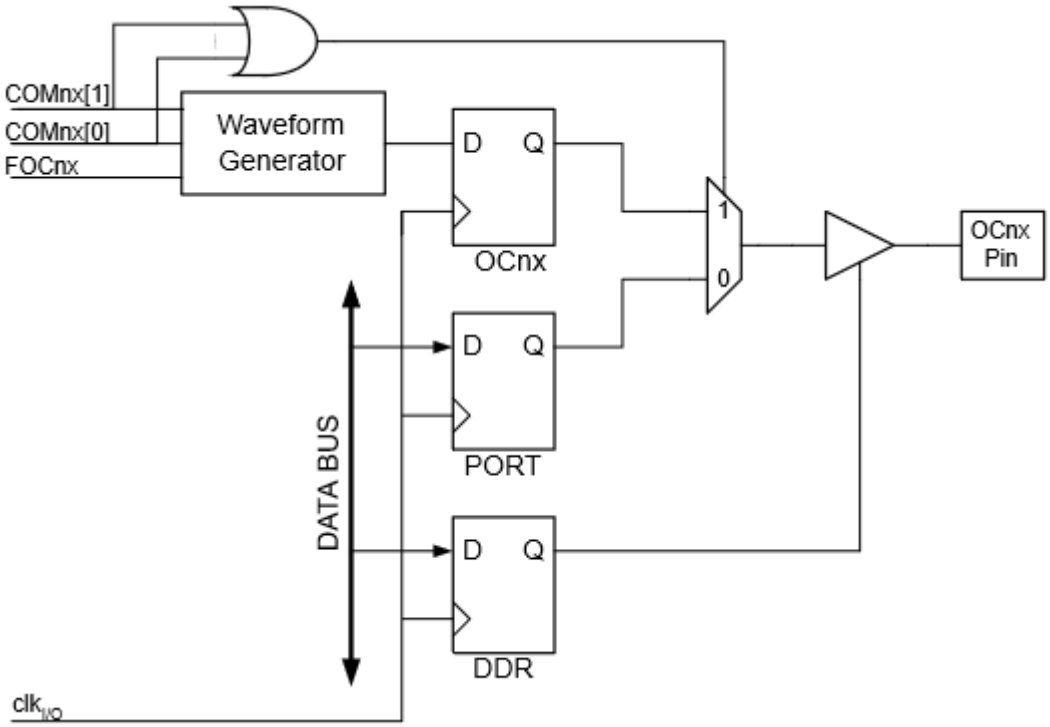


Рисунок 6.11- Схема модуля порта ввода/вывода

Общая функция порта ввода/вывода переопределяется выходным значением бита для OCnx pin из генератора формы сигнала, если установлен бит COMnx [1:0]. Однако направление порта OCnx (вход или выход) по прежнему контролируется регистром направления данных (DDR) для вывода порта. В регистре направления данных бит для OC1nx pin (DDR.OC0x)

должен быть установлен как выходной сигнал до того. Функция переопределения портов не зависит от режима генерации формы волны. Конструкция логики вывода позволяет инициализировать состояние регистра OC0x до того, как выход будет включен.

Режим работы генератора ШИМ определяет поведение контактов таймера/счетчика и вывода. Он определяется комбинацией битов режима генерации формы сигнала и режима сравнения выходного сигнала в регистрах управления таймером/счетчиком A и B. Биты COMnx [1:0] определяют, должен ли ШИМ вывод быть инвертирован или нет.

Простейшим режимом работы является режим нормальный (Normal). В этом режиме направление подсчета всегда увеличивается, а счетчик не очищается. Счетчик просто перехватывается, когда он передает свое максимальное 8-битное значение, а затем перезапускается. В режиме нормального режима флаг переполнения таймера/счетчика (TOV1) будет установлен в тот же такт, в котором TCNTn становится нулевым. В этом случае флаг TOV1 ведет себя как девятый бит, за исключением того, что он установлен, а не очищен. Однако в сочетании с прерыванием переполнения таймера, который автоматически очищает флаг TOV1, разрешение таймера может быть увеличено с помощью программного обеспечения. В нормальном режиме новое значение счетчика может быть записано в любое время.

6.10 Аналогово-цифровой преобразователь

Микроконтроллер оснащён 10-битным АЦП с последовательным приближением. АЦП подключен к 8-канальному аналоговому мультиплексору. Входы с односторонним напряжением относятся к 0 В (GND). АЦП содержит схему, которая гарантирует, что входное напряжение на АЦП удерживается на постоянном уровне во время преобразования. АЦП имеет отдельный аналоговый вывод напряжения питания, AVCC.

Канал аналогового ввода выбирается путем записи в биты MUX в регистре выбора ADC мультиплексора ADMUX.MUX [3:0]. Мультиплексор по команде АЦП из нескольких аналоговых входов использует только один. Любой из входных АЦП портов, а также GND и опорного напряжения может быть выбран в качестве одного состава входов в АЦП. АЦП активируется путем записи «1» в бит АЦП «Активировать» в регистре управления АЦП и регистра состояния (ADCSRA.ADEN). Источник опорного напряжения и входного канала выбор не вступит в силу до тех пор, пока ADEN установлен. АЦП генерирует 10-битный результат, который представлен в регистрах данных ADC, ADCH и ADCL.

Для осуществления преобразования АЦП использует эталонное значение напряжения, с которым сравнивается входной сигнал с аналогового входа микроконтроллера. Для этого предназначен источник опорного напряжения, в качестве которого используется напряжение питания.

Разрешение 10 бит позволяет получать значения сигнала от 0 до 1023 или 2^{10} .

Процесс оцифровки происходит подбором близкого значения напряжения по отношению к входному и состоит из 10 этапов. Сначала АЦП формирует напряжение равное половине опорного и сравнивает его с входным напряжением. Если напряжение АЦП меньше входного то соответствующему биту присваивается значение «1». Затем напряжение АЦП увеличивается на 1/4 опорного напряжения и если оно окажется больше входного, то биту присваивается значение «0», а напряжение АЦП уменьшается на 1/4 опорного. Далее коррекция составляет 1/8 опорного, затем 1/16 и т.д.

Для управления АЦП существует 5 основных восьмибитных регистров:

-Регистр ADCSRA (ADC Control and Status Register A), ADMUX, ADCSRB, ADCL, ADCH.

-Назначение битов регистра ADCSRA (ADC Control and Status Register A, регистр управления и состояния):

-ADEN (ADC Enable, включение АЦП) - флаг, разрешающий использование АЦП, при сбросе флага во время преобразования процесс останавливается.

-ADATE (ADC Auto Trigger Enable) - выбор режима работы АЦП. «0» - разовое преобразование (Single Conversion Mode); «1» - включение автоматического преобразования по срабатыванию триггера (заданного сигнала). Источник автоматического запуска задается битами ADTS регистра ADCSRB.

-ADSC (ADC Start Conversion, запуск преобразования)-флаг установленный в «1» запускает процесс преобразования. В режиме Single Conversion (ADATE=0) для запуска каждого нового преобразования этот флаг должен быть установлен в единицу. После завершения преобразования, флаг ADSC сбрасывается в «0».

В режиме Free Running Mode (ADATE=1, ADTS[2:0]=000) этот флаг должен быть установлен в единицу один раз - для запуска первого преобразования, следующие происходят автоматически. Если установка флага ADSC происходит во время или после разрешения АЦП (ADEN), то перед запрашиваемым преобразованием происходит дополнительное (extended) преобразование, во время которого происходит инициализация АЦП. Сброс флага во время преобразования не оказывает никакого влияния на процесс.

Free Running Mode - режим свободного запуска, запускающий новое преобразование и оно будет начато сразу же после завершения преобразования, в то время как значение ADCRSA.ADSC остается высоким.

ADIF (ADC Interrupt Flag) -флаг прерывания от компаратора

ADIE (ADC Interrupt Enable) - разрешение прерывания от компаратора

ADPS[2:0] (ADC Prescaler Select) - комбинацией битов задается частота преобразования АЦП по отношению к частоте микроконтроллера. Выбранная частота влияет на точность - чем выше частота, тем ниже точность. АЦП тактируется через выбранный делитель от частоты ядра микроконтроллера.

Частота работы МК Atmega 328P 16МГц. При настройках по умолчанию используется предделитель 128 (ADPS[2:0]=[111]), а это значит, что АЦП работает на частоте 16МГц/128=125КГц. По умолчанию для последовательной схемы аппроксимации требуется входная тактовая частота от 50 кГц до 200 кГц, чтобы получить максимальное разрешение. Это даёт низкую скорость преобразования, но высокую точность. Если требуется меньшее разрешение, чем 10 бит, входная тактовая частота для АЦП может быть выше 200 кГц, чтобы получить более высокую частоту дискретизации. Модуль ADC содержит предварительный делитель, который генерирует приемлемую тактовую частоту АЦП с любой частотой процессора выше 100 кГц. Предварительная калибровка выбирается битами ADC Prescaler Select в регистре управления АЦП и регистром состояния А (ADCSRA.ADPS). Предделитель начинает отсчет с момента включения АЦП, записав бит ADCSRA.ADEN АЦП ADAD на «1». Предделитель продолжает работать до тех пор, пока ADEN = 1, и непрерывно сбрасывается, когда ADEN = 0. При инициировании однократного конвертирования путем записи «1» в бит конверсии запуска ADC (ADCSRA.ADSC) преобразование начинается с следующего возрастающего фронта цикла синхронизации АЦП.

Регистр ADMUX **(**ADC Multiplexer Selection Register)

Назначение битов регистра ADMUX:

REFS[1:0] (Reference Selection Bit) — биты определяют источник опорного напряжения.

ADLAR (ADC Left Adjust Result) — бит отвечающий за порядок записи битов результата в регистры ADCL и ADCH.

MUX[3:0] (Multiplexer Selection Input) — биты выбора аналогового входа.

7 Реализация управления двигателем

Для запуска двигателя используется сигнал генератора частот. Принимаемый сигнал поступает на микроконтроллер Areaduino Nano через оптрон. Применение оптопары необходимо для передачи сигнала включения без высокого напряжения с генератора, для борьбы с шумом и электрическими наводками.

Для коммутации обмоток и реализации обратной связи используется датчик Холла, встроенный в корпус платы. Датчик необходим для определения положение ротора двигателя и направления тока в обмотке. При каждом прохождении магнитного полюса двигателя рядом с датчиком Холла, он выдаёт высокий или низкий уровень сигнала, что указывает на то, каким именно полюсом повёрнут магнит: северный или южный. То есть возможны два положения датчика. С помощью сочетания этих сигналов с датчика Холла формируется точная последовательность коммутации.

При поступлении сигнала с генератора микроконтроллер посылает сигнал ШИМ на один драйвер int100s открывая сопутствующие транзисторы и по обмотке двигателя проходит ток в течении времени, равном времени прохождения от одного магнита двигателя до другого, относительно датчика Холла. Использование ШИМ позволяет изменять напряжение на обмотке, тем самым меняя скорость вращения. Сигнал с датчика поступает на микроконтроллер и в соответствии с данным сигналом, микроконтроллер отключает подачу сигнала с текущего драйвера и подаёт ШИМ на второй цифровой драйвер int100s, который также управляет парой транзисторов. Направление тока меняется на противоположное, в результате появляется пусковой электромагнитный момент.

Если при открытых транзисторах одного из драйверов int100s по истечении короткого времени двигатель может не запуститься. Это означает что не возникает электромагнитный момент, значит полюса ротора и зубцов статора находятся противоположно друг другу и они взаимно притягиваются и ток в обмотке протекает в неверном направлении.

Планировалось подавать питание на Arduino Nano с микросхемы TOP210PFI, установленной на плате двигателя через блок питания постоянного напряжения. Блок питания необходим для подачи рекомендуемого напряжения и стабильной работы микроконтроллера.

В данном проекте Arduino Nano используется вместо микроконтроллера PIC, который необходимо было заменить аналогичным по функциональному назначению устройством.

ЗАКЛЮЧЕНИЕ

В результате данной работы была разработана система управления запуска вентильного двигателя. В процессе работы были приведены:

- описание объекта и его применение;

-описание характеристик и принципа работы оборудования, используемого в плате управления;

- схемы управления, подключения электропривода, питания контроллеров и оборудования, подключения оборудования ко входам ПЛК и его модулей;

-описание программируемых логических контроллеров ArduinoMega 2560 и ArduinoNano;

- описание программного пакета среды Arduino IDE;

В процессе выполнения работы было изучены программное обеспечение Arduino IDE, методики проектирования и программирования систем управления с использованием микроконтроллера Arduino, а также разработано программа для управления двигателем, выполнены конфигурирование и наладка оборудования.

СПИСОК СОКРАЩЕНИЙ

ШИМ – Широтно-импульсная модуляция, стр. 7.
ПЛК – Программируемый логический контроллер, стр. 9.
UART – Universal Asynchronous Receiver-Transmitter, стр. 9.
USB – Universal Serial Bus, стр. 9.
ISCP – In System Programming и In Circuit Serial Programming, стр. 9.
LS IN – Low Signal, стр. 16.
HS IN – High Signal, стр. 16.
АЛУ – Арифметико-логическое устройство, стр. 37.
SRAM – Статическая паять с произвольным доступом, стр. 38.
CPU – Central Processing Unit, стр. 40.
MCU – Microcontroller Unit, стр. 41.
CLI – Command Line Interface, стр. 42.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 2.709-89 ЕСКД. Обозначения условные проводов и контактных соединений электрических элементов, оборудования и участков цепей в электрических схемах - Введ. 01.01.1990. – Москва :Стандартинформ России, переиздание 2007 г.

2. Офсетная печать [Электронный ресурс]//Сайт типографии «Лит». - Режим доступа: http://litvl.ru/stati/chto-takoe-ofsetnaya-pechat-(ploskaya-pechat).html

3. Arduino Mega 2560 [Электронный ресурс] //Arduino [Электронный ресурс]//Сайт пользователей«Arduino». - Режим доступа: http://arduino.ru/Hardware/ArduinoBoardMega2560

4. Arduino Nano[Электронный ресурс] //Arduino [Электронный ресурс]//Сайт пользователей «Arduino». - Режим доступа: http://arduino.ru/Hardware/ArduinoBoardNano

5. Arduino [Электронный ресурс]//Сайт программной среды и фирмы «Arduino». - Режим доступа: https://www.arduino.cc

6. Программирование Arduino [Электронный ресурс]//Сайт пользователей«Arduino». - Режим доступа: http://arduino.ru/Reference

7. Прерывания на микроконтроллере Arduino [Электронный ресурс]// Сайт по машинному обучению и программированию. - Режим доступа: http://robotosha.ru/arduino/arduino-interrupts.html

8. Аналоговые измерения с Arduino [Электронный ресурс]//Сайт пользователей«Arduino». - Режим доступа: http://robotosha.ru/arduino/analog-measurements-arduino.html

9. Функции Arduino [Электронный ресурс]//Сайт http://codius.ru. - Режим доступа: http://codius.ru/articles. html

10. Патентный поиск, поиск патентов и изобретений РФ и СССР [Электронный ресурс]//Патент: Однофазный вентильный электродвигатель- Режим доступа: http://www.findpatent.ru/patent/245/2453968.html

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра «Электротехнические комплексы и системы»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ В. И. Пантелеев
«13» 06 2018 г.

БАКАЛАВРСКАЯ РАБОТА

13.03.02 - Электроэнергетика и электротехника

ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ВЕНТИЛЬНЫМ ДВИГАТЕЛЕМ

Руководитель	13.06.18 подпись, дата	к.т.н., доцент должность, ученая степень	А.Н. Пахомов инициалы, фамилия
Выпускник	Иванов 13.06.18 подпись, дата		Ю.М.Иванов инициалы, фамилия

Консультант:

Генеральный директор ООО «Вертекс»	13.06.18г подпись, дата	Д.С. Жидков инициалы, фамилия
Нормоконтролер	13.06.18г подпись, дата	А.Н. Пахомов инициалы, фамилия

Красноярск 2018