Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования  
«ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(ПетрГУ)  
Физико-технический факультет  
Кафедра электроники и электроэнергетики

**Создание лабораторного стенда:**

**Векторный электропривод**(Выпускная квалификационная работа)Электроэнергетика и электротехника (электропривод и автоматика)

Выполнил:  
Студент 4 курса  
Физико-технического   
Факультета, гр.21419  
Рубахин Константин Владимирович  
Научный руководитель:  
старший преподаватель КЭиЭ  
Приходченко Роман Викторович

Петрозаводск 2016

Оглавление

[Введение 2](#_Toc453365613)

[Теоретическая часть 3](#_Toc453365614)

[Git 4](#_Toc453365615)

[Общие данные 4](#_Toc453365616)

[Три состояния 7](#_Toc453365617)

[Установка и дальнейшее использование Git'а 7](#_Toc453365618)

[Векторное управление 11](#_Toc453365619)

[Общие сведения 11](#_Toc453365620)

[Неподвижная трехфазная система координат и понятие обобщенного вектора 13](#_Toc453365621)

[Неподвижная Декартова система координат 14](#_Toc453365622)

[Вращающаяся Декартова система координат XY 15](#_Toc453365623)

[Геометрический вывод формул преобразований Парка 16](#_Toc453365624)

[Смысл системы координат XY 17](#_Toc453365625)

[Основная идея построения системы регулирования скорости синхронного двигателя 17](#_Toc453365626)

[Датчики 18](#_Toc453365627)

[Практическая часть 20](#_Toc453365628)

[Датчики 20](#_Toc453365629)

[Программа для микроконтроллера 27](#_Toc453365630)

[Выводы 38](#_Toc453365631)

[Список используемых источников 39](#_Toc453365632)

# Введение

Мы живем в двадцать первом веке и нашу жизнь не возможно представить без электричества, во всяком случае в городах. Электричество даёт "жизнь" всем электронным приборам. Их назначение может быть всевозможное. Одни помогают в повседневной жизни, а другие помогают создавать первые. Стратегическая часть жизни государства это коммуникация населения, его жизнеобеспечение и обеспечение различными товарами. Для последней категории используются силовые, мощные двигателя работа которых управляется приводом. Общепризнанна целая классификация двигателей, и для каждого необходим свой способ управления, свой привод . Как уже было сказано речь идёт о мощных двигателях которые потребляют большое количество энергии. Вообще система электроснабжения непростая и требует к себе серьёзного обращения. На каждое предприятие выделяется по специальной заявке определённое количество энергии. И чем больше её выписывается, тем больше её стоимость. На предприятии двигателя потребляют большую часть энергии, исключение составляют печи. Естественно что чем меньше будет потребление двигателя, тем меньше придётся выписывать энергии и как следствие у предприятия будет меньше расходов. В зависимости от типа управления двигателем будут разные потери энергии. Рассмотренное в этой работе векторное управление хорошо смотрится на фоне других форм управления двигателем. Его использование помогает увеличить кпд двигателя и как следствие уменьшить потребление им энергии за цикл работы.

Цель работы: произвести векторное управление бесколлекторным электродвигателем.

Для реализации цели были поставлены следующие задачи:

1) Изучить процесс векторного управления.

2)Создать датчики тока, для отслеживания его величины и использования его при векторном управлении.

3)Написать программу по которой будет выполняться управление двигателем.

4)По возможности собрать стенд и запустить все комплектующие, чтобы убедиться в работоспособности метода.

# Теоретическая часть

Для выполнения этой работы необходимо было изучить само векторное управление. Это понятие довольно обширное, и под ним может быть заложена любая тема где упоминаются вектора. Управление можно производить как током так и напряжением. Векторное управление это управление моментом на валу двигателя. Если более точно, то необходимо чтобы поле статора всегда опережало поле ротора на 90 градусов. Для управления моментом необходимо знать ток на каждой фазе двигателя и отслеживать скорость ротора. Зная токи по фазам, преобразуем из трехфазной системы в двухфазную и после этого по полученным данные получаем суммарный вектор тока. Этот вектор задает момент на валу. Так как мы можем подавать напряжения на обмотки статора то сможем поддерживать опережение поля статора относительно поля ротора.

## Git

Перед изучением данных вопросов была изучена СКВ "Git". Система контроля версий (далее СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Этот продукт не имеет прямого отношения к основной теме, однако если изучать какую либо проблему не одному а в компании заинтересованных лиц, то он позволяет распределить нагрузку на каждого. В результате скорость работы увеличивается. Например при написании программного обеспечения, разные утилиты могут писать разные инженеры, а после всё полученное объединяется в готовый продукт. Однако это не единственное его преимущество. Ещё одна возможность Git'a это возможность просмотра старых версий документа. Это позволяет когда разработка какого либо вопроса в тупик вернуться на прежние этапы и пойти по другому пути решения поставленной задачи. Как пример можно вспомнить сохранение в играх. Более поверхностной функцией СКВ является схожесть с файлообменниками. То есть один человек может загрузить на сервер какие либо файлы, а другой эти файлы скачать. Также Git предоставляет возможность найти в архивах похожие работы, и если они также разрабатываются в данный момент и разрабатываются энтузиастами бесплатно, то можно подсмотреть какой нибудь момент на котором были сомнения в своём проекте.

### Общие данные

Есть несколько видов СКВ:

1. Локальные СКВ;
2. Централизованные СКВ;
3. Распределенные СКВ;

1) Локальные СКВ это СКВ с простой базой данных, в которой хранятся все изменения нужных файлов (см. рисунок 1). Одной из наиболее популярных СКВ такого типа является rcs, которая до сих пор устанавливается на многие компьютеры. Даже в современной операционной системе Mac OS X утилита rcs устанавливается вместе с Developer Tools. Эта утилита основана на работе с наборами патчей между парами версий (патч — файл, описывающий различие между файлами), которые хранятся в специальном формате на диске. Это позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.

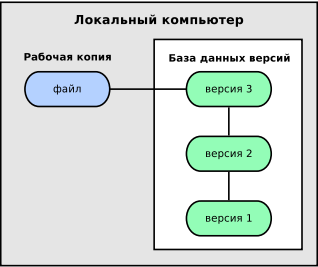


Рис. 1 Схема локальной СКВ

2) Централизованные СКВ решают проблему общения пользователей за разными ПК. Примерами такого типа СКВ являются CVS и Subversion. Они имеют один общий сервер, на котором хранятся все файлы под версионным контролем и доступ имеют все пользователи. Какое то время это являлось стандартом. Такого типа СКВ проще контролировать, но это одновременно является уязвимым местом, в том плане что если отключается центральный сервер, то отключается сообщение между всеми пользователями в системе. И ещё одной проблемой является то что вся история изменений файлов храниться в одном месте.

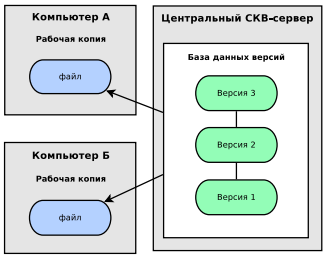


Рис. 2 Схема централизированной СКВ

3) Представителями распределенной СКВ являются Git и Mercurial. В данных СКВ пользователь копирует сразу весь запрашиваемый репозиторий. "Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно параллельно работать с разными группами людей по-разному в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах."

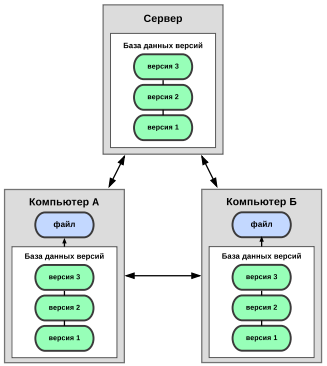


Рис. 3 Схема распределенной СКВ

### Три состояния

"В Git’е файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. «Зафиксированный» значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

Стандартный рабочий процесс с использованием Git’а выглядит примерно так:

1. Вы вносите изменения в файлы в своём рабочем каталоге.
2. Подготавливаете файлы, добавляя их слепки в область подготовленных файлов.
3. Делаете коммит, который берёт подготовленные файлы из индекса и помещает их в каталог Git’а на постоянное хранение.

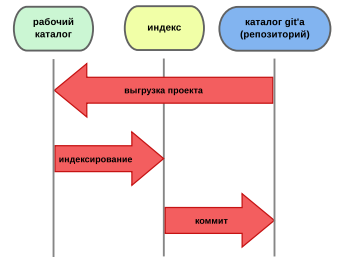


Рис. 4 Локальные состояния

### Установка и дальнейшее использование Git'а

Теперь, ознакомившись с общими данными о СКВ, рассмотрим как с ним работать. Наше изучение начинается с книги "Pro Git". Из неё мы узнаем теоретически все необходимые команды которые могут нам помочь при первом изучение Git'a, а также узнаем откуда взять этот продукт. Для начала зайдём на официальный сайт этого СКВ https://git-scm.com и скачаем бесплатное программное обеспечение с него. Внешнее оформление сайта на момент написания данной работы представлено на рис. Процесс установки этой программы не представляет собой ничего особенного и устанавливается как и большинство других программ. Клиент для работы с Git'ом на компьютере называется Git Bash. Он также автоматически устанавливается вместе со скачанной программой с официального сайта.

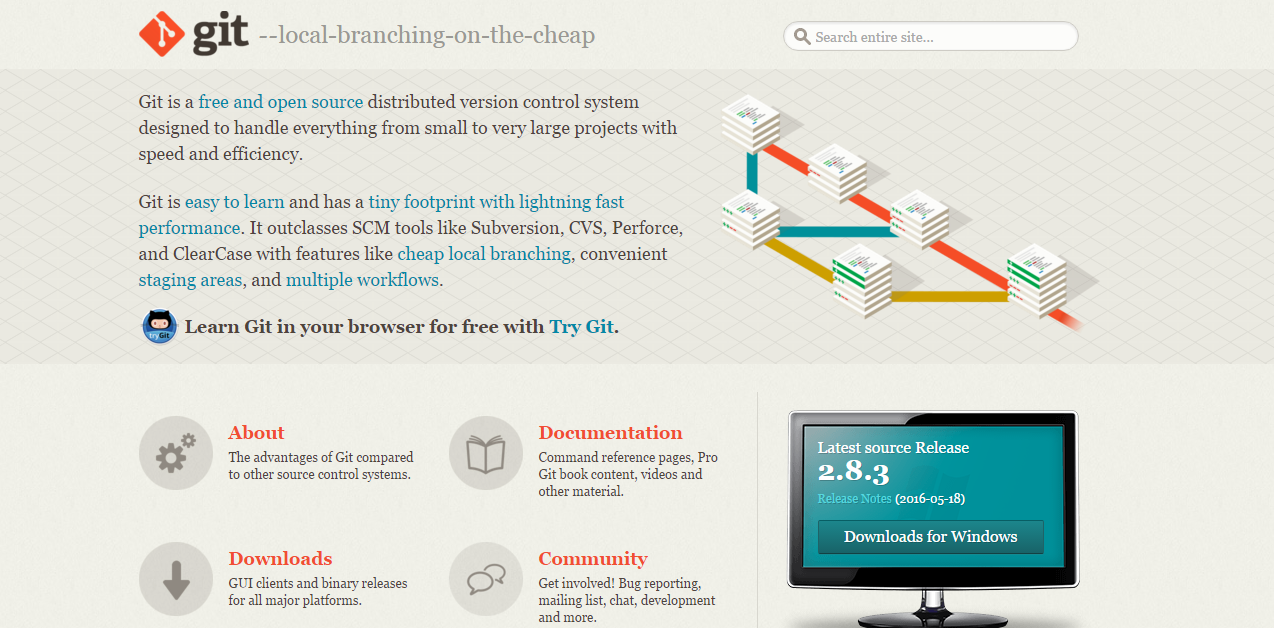


Рис. 5 Официальный сайт Git

Следующим шагом необходимо зарегистрироваться на сайте https://github.com. В процессе регистрации предлагается использовать платный режим. В нем все данные будут скрыты от общественности и доступны только разработчикам. Общение с Git'ом происходит как раз через клиент. Для начала работы необходимо создать репозиторий. Чтобы его создать, заходим в папку в которой будет находиться проект, если этой папки нет то создаем и её, и далее можно действовать двумя способами. Первый это нажав правую кнопку мыши на пустом месте в папке и выбрать "Git init here". Для Второго способа надо зайти в клиент, нажав ПКМ и выбрав "git bash here", и прописать соответствующую команду "$ git init".

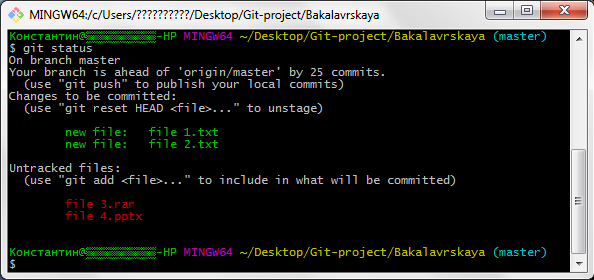


Рис. 6 Окно терминала Git

После этих действий создастся скрытая папка, в которой будут находится все настройки данного репозитория.

Возвращаясь немного к окну терминала можно сказать что оно сделано довольно неплохо, в том плане что все состояния файлов различаются по цветам (Рис. 6). Это помогает не ошибиться в выборе действия с файлами, например когда после продолжительной работы может теряться внимание.

Так как СКВ могут пользоваться разные люди, необходимо обозначить себя. Для этого прописываем команды с глобальными ключами "$ git config --global user.name "name"", где "name" это имя пользователя. Конфигурации могут быть локальными и глобальными. Локальные могут быть разные, для разных проектов. А глобальные будут использоваться по умолчанию, если не задать конфигурации вручную. Теперь зададим электронную почту пользователя. Сделаем это командой "$ git config --global user.email xxx", где xxx это адрес электронной почты. Настройка почты также является глобальной. Все настройки передаются в файл .gitconfig. Можно напрямую писать настройки в этом файле. Чтобы посмотреть все настройки в Git'e на данный момент можно воспользоваться командой "$ git config --list", и клиент выдаст на экран список всех настроек на данный момент. Среди них будут и те что уже настроены и те что идут по умолчанию.

Теперь необходимо узнать как добавлять файлы в Git. По умолчанию мы начинаем работу в ветке "master". Мы можем создавать и переходить между ветками, но об этом речь пойдет позже. В данной системе файлы могут иметь разный статус. Как только мы создаем в рабочей ветке какой либо файл, или копируем с внешнего источника какой либо документ, он получает не отслеживаемый статус. Чтобы увидеть как Git видит файлы можно воспользоваться командой "$ git status". Если нужно всё таки добавить файлы , то их начинают отслеживать. Для начала, чтобы добавить в коммит файлы, воспользуемся командой "$ git add xy", где "x" это ключ команды, а "y" имя файла. Если как ключ использовать ".", то в коммит будут добавлены все без исключения файлы. А если вместо ключа прописать полное имя файла, то проиндексируется только этот файл. Коммит это своего рода список всех изменений.

Как видно на рис. 6 если файл не отслеживается, то он находится в секции "Untracked files" и документ выделяется красным цветом. Если файл был подготовлен и добавлен в коммит, он выделяется зелёным цветом и находится в секции "Changes to be committed". Даже если в файле был изменён лишь один символ, то Git отобразит это терминале, при запросе "статуса". Если нужна более подробная информация, то вместо "$ git status" стоит использовать команду "$ git diff".

Если какой то файл был добавлен в коммит случайно, или произошла такая ситуация при которой проще добавить все файлы подряд, а потом удалить пару ненужных точечно. Чтобы удалить такой файл воспользуемся командой "$ git rm --cached "file"", где "file" это имя файла, которое совпадает с точностью до знака с тем, которое было прописано после команды add. Если ключа "--cached" не будет, то файл будет удален не только из коммита, но и из рабочего каталога. Как и других терминалах, команды можно просто пролистать и выбрать нужную, если она уже была использована, для этого пользуются стрелочками вверх и вниз.

Чтобы система запомнила состояние файлов необходимо провести коммит. Для этого необходимо воспользоваться командой "$ git commit -a -m "xxx"", где ключ -m означает добавление комментария к коммиту, ключ -a выберет все последние добавленные в коммит файлы, то есть слепок будет не на весь каталог, а только на последние добавленные или изменённые файлы после последнего коммита, а "xxx" это сам текст комментария. Он обязателен, и если его не добавить, то клиент выдаст окно, в котором нужно будет ввести комментарий.

В терминале Git'a как и в некоторых других терминалах есть специальные команды. Если достаточно много и неповерхностно работать с данным СКВ, то придется пользоваться немалым объёмом команд. К счастью можно воспользоваться командой "$ git help" и тогда пользователю будет предоставлен список всех ключей и команд.

Даже при поверхностном рассмотрении нужно будет заливать на сервер свои данные. Во-первых это дает постоянный доступ к предыдущим версиям файлов. А во-вторых если работа над проектом ведётся несколькими людьми, то использования сервера позволит экономить время при передачи файлов. Данная процедура сопровождается командой "$ git push xxx", где xxx это адрес репозитория.

Обратная процедура это получение данных с репозитория. Её можно провести двумя способами: зайдя на сайт и сохранить или скопировать нужные файлы, или вручную через терминал. Для этого используется команда "$ git pull xxx", где xxx это адрес файла. Также существует команда "$ git clone xxx", где xxx это адрес репозитория. При использования первой команды скачается точный файл, который указан, а во втором случае скачается весь репозиторий. Скачивание файлов происходит в папку, которая указана в строке перед командой.

Git имеет ещё множество функций, и часть даже была изучена, но ей не пришлось пользоваться и поэтому она не описана.

## Векторное управление

Данный метод появился в семидесятых годах прошлого века и имеет ряд преимуществ и недостатков. К преимуществам можно отнести:

1. Высокая точность при регулировании скорости;
2. Более высокий кпд двигателя;
3. При изменении нагрузки почти не меняется скорость;
4. Плавность;

К недостаткам можно отнести следующее:

1. Относительно высокая сложность вычислений;
2. Необходимость задания параметров двигателя;
3. Стоимость оборудования;

Токи, потоки, напряжения двигателя переменного тока могут быть представлены как векторы в некоторой системе координат. В зависимости от используемой системы координат соотношения, описывающие процессы, происходящие в двигателе, будут иметь различный вид. Ниже рассмотрим системы координат, используемые при векторном регулировании.

### Общие сведения

Векторное управление это управление моментом, поэтому необходимо знать все его составляющие, момента который заставляет ротор вращаться. Для создания лабораторного стенда имеется arduino, мост управления по фазам, спаянные датчики тока, защита, драйвера и сам двигатель с элементами холла. Весь процесс будет идти следующим образом: на компьютере пишется программа на arduino состоящая из 3 подпрограмм, по этой программе происходит подача нужного напряжения по фазам. Более подробно этот процесс будет рассмотрен позднее. На пути к двигателю стоят датчики тока "на шунтах", которые показывают как меняется ток в фазах с течением времени. В двигателе есть датчики Холла и с помощью них будет отслеживаться в каком положение находится ротор. В каждой фазе мы знаем как расположены вектора, от этого и будем отталкиваться.

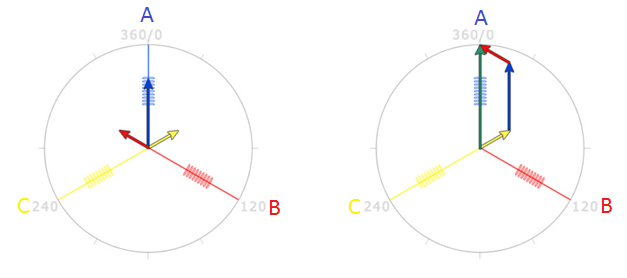


Рис. 7 Вариант расположения векторов

На левом рис. 7 пример для положения ротора в фазе "А". Для двух других фаз ситуация будет симметричная. На правом рис. 7 показано как будут суммироваться вектора. В данном случае у нас есть система координат с двумя осями. Значит мы можем представить вектор от каждой оси относительно общей точки. Если по примеру на левой картинке то место начала векторов можно принять за ноль. Тогда вектор фазы "А" будет иметь максимальное значение по "Y" и нулевое по "Х". Все фазы сдвинуты на 120 градусов, значит на суммарный вектор каждая фаза имеет ту же треть цикла с максимальным влиянием. В примере как раз такой случай. Для фазы "В" максимальное значение будет как раз спустя 120 градусов относительно фазы "А" и также для фазы "С" спустя 240 градусов относительно "А". Так как мы приняли двухмерную систему координат, мы можем спроектировать вектор каждой фазы на оси и тем самым знать какое значение надо задавать. В результате мы будем иметь суммарный вектор, который вращается вместе с ротором и который мы должны приводить в движение. Значит нашей целью является написание программы, которая будет описывать цикл при котором суммарный вектор сделает один полный оборот. Одна подпрограмма будет отслеживать ротор и считывать скорость его вращения, другая переводить трёхфазные величины в двухфазные и будет производить преобразование Парка и Кларка, а также связывать системы координат статора и ротора. В конечном итоге нам и надо чтобы суммарный момент статора на 90 градусов обгонял суммарный момент ротора.

С помощью компьютера написана и загружена в микроконтроллер программа. Эта программа производит вычисления по заданной программе и отправляет необходимое напряжение на каждую фазу с помощью широтно-импульсной модуляции. По обратной связи на контроллер приходят положение ротора и ток в каждой фазе. Эти данные обсчитываются на контроллере и снова выдаётся необходимое напряжение на мост управления фазами и всё повторяется.

### Неподвижная трехфазная система координат и понятие обобщенного вектора

Система имеет три оси, расположенные на плоскости под углом в 120 градусов. Их пересечение показано на рисунке 8.

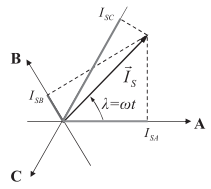


Рис. 8 Пересечение осей в начале координат

Данная система позволяет отобразить процессы, происходящие в двигателе, на плоскость при представлении токов, напряжений и потокосцеплений в виде вращающихся векторов. Например, трехфазный ток в обмотках статора двигателя можно представить в виде вектора IS, имеющего следующие свойства:

* амплитуда вектора равна амплитуде тока в фазе (IS);
* начало вектора совпадает с началом координат;
* вектор вращается на плоскости вокруг начала координат с угловой скоростью, соответствующей частоте переменного тока (ω = 2πf).

При вращении такого вектора с частотой ω его проекции на соответствующие оси будут меняться по синусоидальному закону, при этом между синусоидами в каждой фазе будет сохраняться сдвиг в 120 градусов. Кроме того, для значений проекций вектора на оси системы координат будет всегда выполняться условие симметрии трехфазной системы: Isa+Isb+Isc=0.

Таким образом, три связанные скалярные величины фазных токов характеризуются в данной системе координат обобщенным вектором тока. Аналогично, в виде обобщенных векторов, в данной системе координат могут быть представлены фазные напряжения и потокосцепления асинхронного двигателя. Во время работы двигателя создается вращающееся магнитное поле, то есть реально, физически существует вращающийся магнитный поток. Его можно разделить на составляющие, сцепленные со статором и ротором, поток намагничивания и потоки рассеяния, замыкающиеся только через статор или ротор. Можно считать, что неподвижная трехфазная система координат ориентирована по статору, а ее оси совпадают с электрическими осями соответствующих обмоток двигателя.

### Неподвижная Декартова система координат

Система имеет две перпендикулярные оси α и β (рис. 9). Удобно выбирать оси так, чтобы одна из них совпадала с одной из трехфазных осей. Начало координат совпадает с началом координат описанной выше трехфазной системы.

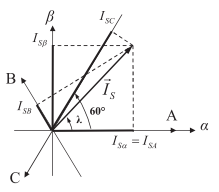


Рис. 9

Переход из трехфазной системы в Декартову (двухфазную) часто называют переходом от реальной трехфазной машины к абстрактной, обобщенной двухфазной. Получим формулы перехода из трехфазной системы в Декартову. В зарубежной литературе этот переход называют преобразованием Кларка. Глядя на рис. и воспользовавшись соотношениями прямоугольных треугольников, а также формулой для косинуса разности углов, запишем:

Выразив из суммы этих уравнений IS, получим:

.

С другой стороны, (рис. 9) Isb=IsSinλ;

Следовательно, можно записать .

Таким образом, формулы координатных преобразований при переходе из трехфазной системы в двухфазную выглядят так:

Формулы обратного преобразования координат из системы αβ в систему АВС, получающиеся из соотношений и условия симметрии трехфазной системы , приведены ниже:

Иногда удобно описывать вектора в системе αβ, пользуясь математическим аппаратом комплексной плоскости. При этом ось β считают мнимой, а ось α – вещественной.

### Вращающаяся Декартова система координат XY

Система имеет две перпендикулярные оси X и Y (рис.10 ).

Начало координат совпадает с началом координат описанной выше трехфазной (или двухфазной) неподвижной системы. Оси системы XY вращаются с произвольной скоростью ωК вокруг начала координат.

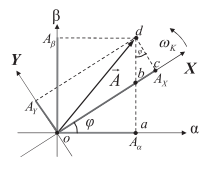


Рис. 10

Найдем формулы преобразования координат, с помощью которых можно осуществлять переход между неподвижной системой αβ и вращающейся системой XY. В зарубежной литературе этот переход называют преобразованиями Парка.

### Геометрический вывод формул преобразований Парка

Треугольники aob и cdb (рис. 10) подобны, следовательно, можно записать

отсюда

Из рис. 10 следует, что

Выразив из этого выражения Aα, получим

Данное соотношение подставим в формулу

и с помощью некоторых преобразований, выразив из нее Aβ, получим

Таким образом, обратное преобразование Парка, формулы перехода из системы XY в систему αβ, выглядит следующим образом:

Решив эти уравнения относительно AX и AY, получим прямое пре- образование Парка – формулы перехода из системы αβ в систему XY:

### Смысл системы координат XY

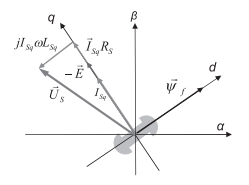
Вращающаяся система координат, на первый взгляд, кажется достаточно искусственной, но именно в ней строятся современные системы векторного регулирования. Система регулирования строится в системе координат, вращающейся вместе с управляемым вектором. Дело в том, что в такой системе дифференциальные уравнения, описывающие двигатель, принимают простейший вид. Кроме того, поскольку управляемый вектор не вращается относительно данной системы координат, его амплитуда и фаза определяются двумя скалярными величинами проекций на оси координат этой системы. Следовательно управление вектором, в данном случае, можно свести к управлению величиной и знаком его проекций. Вращение системы учитывают с помощью рассмотренных выше формул координатных преобразований, измеряя или вычисляя угол ее поворота относительно неподвижной системы координат.

### Основная идея построения системы регулирования скорости синхронного двигателя

Как уже было сказано выше, для управления моментом по минимуму статорного тока ток статора необходимо направлять по оси q. То есть угол между ротором и вектором статорного тока надо поддерживать равным 90 градусов. Система регулирования строится в координатах dq. Регулирование скорости осуществляется с помощью управления моментом двигателя. Если ток статора ориентирован по оси q, то он выражается через момент с помощью соотношения вытекающего из соотношения

.

Векторный регулятор тока строится во вращающейся системе координат dq и состоит из регуляторов d и q проекций. С помощью q-составляющей тока обеспечивается, в соответствии с формулой (33), необходимый момент, а d-составляющая поддерживается равной нулю, что обеспечивает нужную ориентацию вектора тока. В качестве обрат- ной связи регулятор использует измеренный и преобразованный в систему dq реальный вектор статорного тока. Векторный регулятор тока формирует в системе dq вектор напряжения статора, призванный обеспечить совпадение заданного и реального векторов тока статора. Затем с помощью координатных пре- образований вектор напряжения статора переводится в неподвижную систему координат, связанную со статором, где он и реализуется с по- мощью ШИМ.



Векторная диаграмма двигателя рис.11 .

## Датчики

Первой практической задачей в данной работе было изготовление датчиков тока чтобы снимать с двигателя недостающие данные для работы программы. При выборе нужного датчика необходимо было изучить какие вообще существуют датчики и чем хороши и плохи каждые из них. Основные типы датчиков тока это резисторные, датчики на основе элемента Холла и датчики с использованием трансформатора тока. Во всех трех типах есть варианты схем с использованием операционных усилителей. Также датчики различаются по типу измеряемой величины: датчики постоянного и переменного тока. Еще одна классификация датчиков это по типу сигнала. А именно: аналоговые, цифровые и бинарные. Если взять первую классификацию то датчики можно сравнить по основным параметрам. В плане точности все три типа показывают примерно равный результат. Резисторные датчики самые дешевые, но в отличие от двух других не имеют изоляции. Также они единственные из предложенных имеют сдвиг. Они потребляют мощность больше своих конкурентов, что является минусом. И для их работы датчикам необходим контакт. Резисторные датчики имеют самое простое изготовление. У датчиков тока на основе трансформатора тока самая большая сложность с созданием трансформатора тока, а датчики на основе элемента холла имеют малую прочность, их легко спалить по неопытности. Поэтому в нашей работе мы будем использовать резисторные датчики переменного тока. Его принципиальная схема представлена на рис. 12

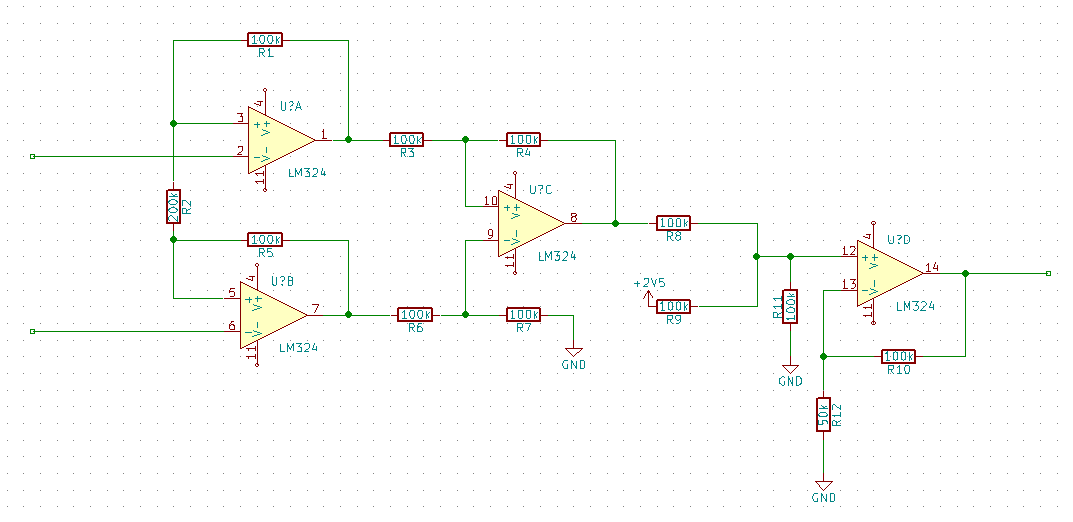


Рис. 12 Схема датчика "на шунте"

# Практическая часть

## Датчики

Схема состоит из операционного усилителя Lm324, шунта и нескольких сопротивлений, которые создают необходимый коэффициент усиления. Шунт подключен входным выводом к неинвертирующему входу усилителя, а выходным к инвертирующему через сопротивление.

Для начала обсчитаем саму схему. Все элементы стенда соединены последовательно, поэтому по закону Ома сила тока будет постоянной. На двигатель приходит около одного ампера, следовательно на шунте будет столько же. Чтобы не создавать значимого влияния на двигатель, на шунте не должно оседать больше 1% мощности. Зная напряжение на двигателе, которое равняется 12В, мы можем посчитать приблизительную мощность: Р=U\*I=12Вт. Исходя из этого расчёта, на шунте должно быть не более 0.12Вт. Зачастую для шунта берут сначала сопротивление около 100мОм, а потом подбирают для лучшего результата. Для шунта возьмём сопротивление 100мОм. Проверим что при таком номинале мощность не будет оказывать существенного влияния на двигатель.

U=I\*R=0.1\*1=0.1В; P=U\*I=0.01\*1=0.1Вт.

Датчик будет отправлять результат на arduino, которое может принять не более 5В. Следовательно коэффициент усиления не может быть больше 50. В усилителе на входе стоит эмиттерный повторитель на р-n-р транзисторах, и чтобы схема работала, нужно подать на базу эмиттера достаточное напряжение, чтобы он открылся и запустил в работу остальную часть операционного усилителя. Проблема в том что в даташите не написано какие именно это транзисторы, и подобрать входное сопротивление идеально не представляется возможным.

Нам необходимо измерять ток на каждой фазе, значит потребуется 3 таких датчика. Для удобства изготовим все три датчика на одной плате объединив питание у всех датчиков а также шину для напряжения смещения. Разработка платы производится в уже изученном ранее KiCad'e. Сначала создаем схему в редакторе схем. Чтобы показать более близко схему, и на ней можно было разобрать радиоэлементы, пришлось увеличить изображение и как результат на рисунке изображен "основной" датчик. "Основным" он называется по причине что он имеет делитель, который будет давать напряжение смещения на 2,5В. Более точная регулировка этого напряжения достигается включением в цепь потенциометра. Напряжение 2,5В будет подаваться на одну из четырех частей усилителя, которая выполняет роль сумматора. Первые три части представляют собой два каскада усиления. То есть на первую и вторую часть счётверенного операционного усилителя, а если быть более точным то на неинвертирующие входа подаётся напряжение. На первую часть идёт напряжение которое идёт до шунта, а на второй части снимается напряжение после шунта. Таким образом мы снимаем падение напряжения на шунте. Первые две части Lm324 за счёт обратной связи усиливают каждый свой сигнал на заданную величину, которая исчисляется как соотношение сопротивления обратной связи к входному сопротивлению. Эти два усиленных на определённую величину сигнала подаются на третью часть усилителя, на инвертирующий и неинвертирующий входы. Происходит усиление по тому же принципу как и на первом каскаде.

Если всё оставить как есть, то на выходе первых трёх частей будет напряжение равное разнице входных сигналов увеличенных на заданный коэффициент усиления. Это значение будет амплитудным и меняться от отрицательного до положительного с определённой частотой. Это ещё одна загвоздка, ведь если подавать на выход отрицательное напряжение, то оно не только будет не воспринято, но и скорее всего спалит микроконтроллер. Для этого и добавляем четвёртую часть усилителя. Надо получить такой результат, чтобы на выходе усилителя было от 0 до 5В и желательно оставить небольшой запас, который будет предохранять от случайных очень резких скачков напряжения. Так как на вход идет как положительное так и отрицательное значение равное по величине как в максимальном значение так и в минимальном то поделим и выходное напряжение пополам. Так получаем что при подаче равных сигналов с разных ножек шунта их разность будет равна нулю и на выходе напряжение должно быть 2.5В. Это и будет тем опорным напряжением которые будем подавать на сумматор. При суммировании опорного напряжения и выходного с первых трех частей усилителя, а также при разнице сигналов в "положительном" диапазоне на выходе будет 5В, а в "отрицательном" 0В.

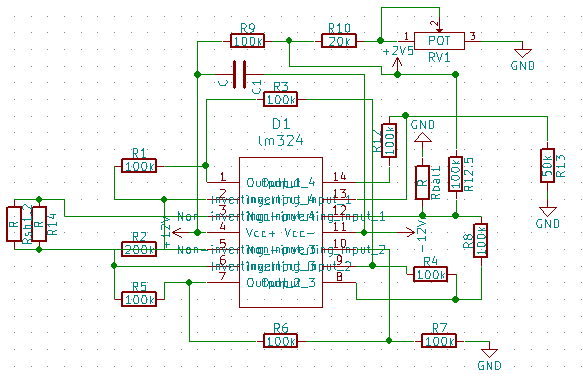


Рис. 13 Соединение радиоэлементов в Eeschema

Затем присуждаем всем элементам их посадочные места и создаем списки цепей.

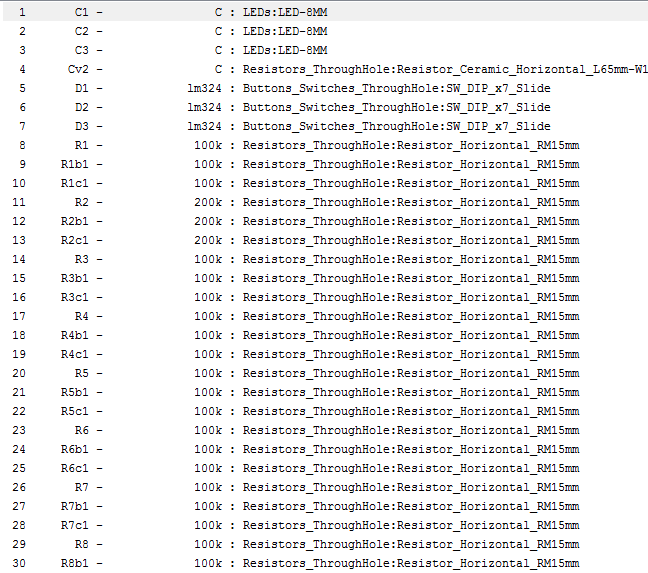


Рис. 14 Первая часть задания посадочных мест

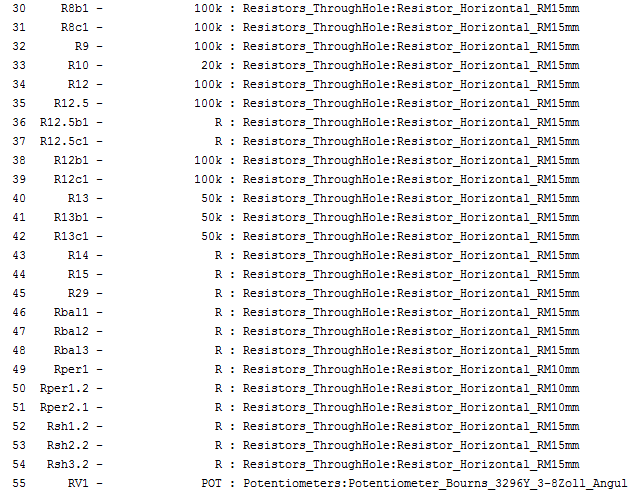


Рис. 15 Вторая часть задания посадочных мест

После этого заходим в редактор посадочных мест и распределяем элементы как они будут находится на плате.

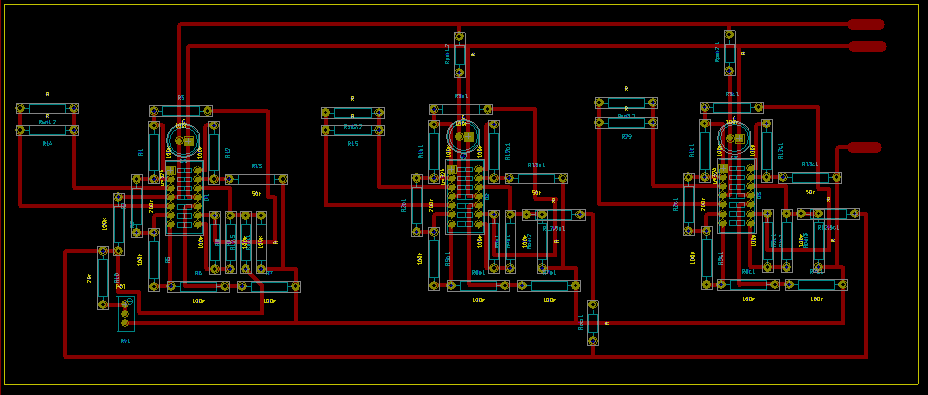


Рис. 16 Разводка дорожек в pcbnew

При распределение элементов возникают сложности. Во-первых пришлось обозначить шунт 2 параллельно соединенными сопротивлениями. Это сделано по двум причинам: чтобы не создавать новое посадочное место и не разбираться с библиотеками. И вторая причина в том что изначально не известно, какой будет шунт. Если брать специализированные шунты то они удобны, продуманы, но дороги, перевешивают в плане цены все остальные радиоэлементы все вместе взятые. Значит нужно найти либо сопротивление наиболее подходящее по номиналу, либо сделать его из нихромовой проволоки, которая обладает очень низким сопротивлением. Так как нам нужно чтобы было как можно меньше паразитных сопротивлений, то располагаем шунт на плате, а провода которые соединяют его и остальную цепь подводим на плату.

Во-вторых нам надо провести 4 шины, общие для всех трех датчиков. И тут представляется выбор: делать одностороннюю плату но с перемычками или делать двустороннюю плату, на которую всё прекрасно ложится и никаких проблем с распределением дорожек не возникает. К сожалению несмотря на всю удобность двусторонней платы при построении её в САПР, трудности возникают на этапе изготовления. Мне известно три способа такой печати.

Первый это печать по специальной "сетке" где основные отверстия можно отмерить по линейке и потом относительно этих точек правильно наложить чертёж, который будет перенесён на печатную плату.

Второй способ заключается в том чтобы сначала на один чертёж разместить сразу обе стороны и правильно их сориентировать, а потом, после распечатки чертежа сложить ровно пополам этот чертёж, внутрь положить подготовленный текстолит и правильно его сориентировать. Дальнейшее изготовление повторяет другие способы.

И третий способ это способ по контрольным точкам. При использовании данного метода сначала переносится на текстолит одна сторона платы, потом заготовка сверлится в нужных точках. После этого распечатывается вторая сторона и с помощью крепёжных устройств крепится к контрольным отверстиям. Затем повторяется последовательность других способов. В моем случае проблема заключалась в том что при крепеже чертежа второй стороны, он не плотно прилегал к текстолиту и плохо копировался.

Лично мною было испробовано два из трех способов и результат был плохим. Точность перевода была не очень высокой и в некоторых местах отверстие было на самом краю дорожки. То есть до 0,5-0,7мм. По этой и ещё одной причине была сделана односторонняя плата. Второй причиной было то что даже мировые производители приводов не брезгуют перемычками на платах. Возможно при наличие большего времени и средств были бы испробованы и другие способы печати. И ещё один момент, все выше описанные методы изготовления печатных плат относятся к лазерно-утюжному методу.

Полученную плату экспортируем со всему необходимыми настройками в Svg формат и печатаем на фотобумагу, не отзеркалив изображение. Следующим этапом берём фольгированный текстолит и очищаем его от окиси, слегка прошкурив поверхность. Переносим с помощью утюга изображение с фотобумаги на текстолит. Это самый опасный момент и не всегда получается с первого раза. Затем аккуратно отмываем фотобумагу зубной щёткой или губкой так чтобы не повредить готовые дорожки. После этого опускаем в железо для травления плату чтобы отчистить от остатков меди поверхность текстолита, и остаются только дорожки. Остается дело за малым, просверлить отверстия для ножек радиоэлементов и покрыть дорожки припоем. Для последнего используется сплав Розе. Он очень удобен в плане красоты, объёмов и в некоторых случаях скорости покрытия. Но за всё это приходится платить. Последним и самым важным этапом является пайка радиоэлементов на плату. Очень неудобно паять сам усилитель, по причине что надо припаять несколько ножек, и главное их нельзя перегревать, в противном случае усилитель сгорит. Для этого используем специальное посадочное место, которое изготавливается универсально, и подходит для любых многоножек в корпусе Soic-14. Ещё одно место, которое требует решения это выводы. В уже имеющихся компонентах стенда выводы сделаны так, чтобы их без проблем можно было соединить с arduino и друг с другом. Поэтому к выводным проводам от шунтов необходимо припаять такие же выводы.

После изготовления первоначального варианта датчиков была произведена их калибровка. Для калибровки сначала было выставлено на шине 2,5В соответствующее значение методом замыкания выводов шунта и подстройкой на потенциометре необходимого сопротивления. Затем была произведена проверка датчиков с помощью подручных средств, а именно батарейкой с делителем на сопротивлениях. Делитель нужен был для того, чтобы не спалить операционный усилитель. В процессе пробного запуска тот факт что на выходе датчика напряжение не выходило за допустимые рамки подтвердилось.

Далее датчик был испытан на стенде. Оказалось что в конце периода оборота двигателя появлялся скачок отрицательного напряжения. От него необходимо было избавится, и для этого на входе был сделан RC-фильтр и на выходе были поставлены диоды. RC-фильтр убирает всплески. Он рассчитан на частоту широтно-импульсной модуляции arduino. Использование данного контроллера имеет особенность. Среди прочих он один имеет фиксированную частоту модуляции равную 490 Hz. Для диодов было поставлено два основных требования, это быстродействие и малое падение напряжения. В связи с этими условиями сначала предполагалось использовать диоды Шоттки, но в конечном итоге были поставлены отечественные Д311. Они имеют ближайшее по значению быстродействие 50 нс против 35 нс у Шоттки (следующая промежуточная величина быстродействия была замечена в значение 100 нс), но зато компенсируют это небольшое отставание в плане падения напряжения.

Готовый вариант датчиков показан на рис. 17.

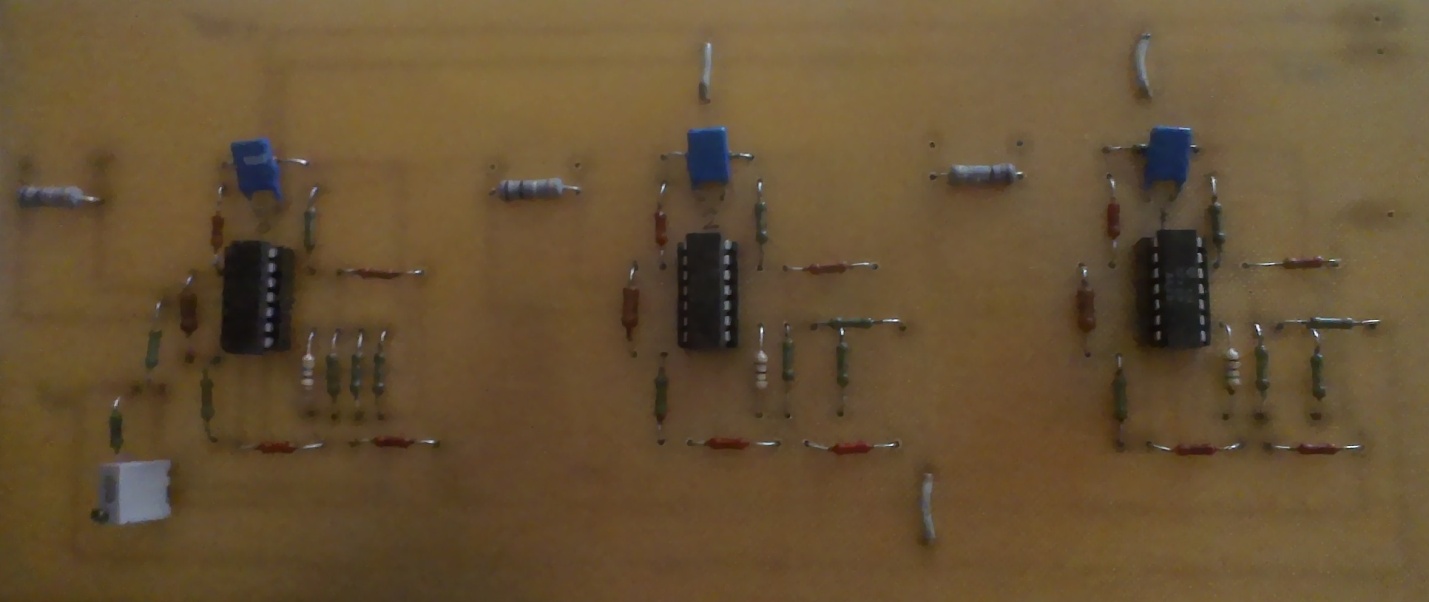


Рис. 17 Готовый вариант датчиков

## Программа для микроконтроллера

Вторая часть практики это программирование arduino так чтобы с неё получать ШИМ на мост управления фазами. Программа пишется в специальной среде разработки и на специальном языке, который написан с основой на С++.

Для начала необходимо описать все выводы , которые будут задействованы в микроконтроллере. Это описывается в самом начале программы:

const int pinAH = 3;

const int pinAL = 7;

const int pinBH = 5;

const int pinBL = 8;

const int pinCH = 6;

const int pinCL = 9;

const int pinAhol = A0;

const int pinBhol = A1;

const int pinChol = A2;

const int pinIA = A3;

const int pinIB = A4;

const int pinIC = A5;

Микроконтроллер имеет цифровые и аналоговые выводы, а также выводы питания. В этой программе будут использованы все аналоговые выводы, по трем будет приходить значение магнитного поля, которые считываются датчиками холла. А ещё по трём будет приходить значения токов пофазно. Для питания будут использоваться выводы 5V и GND. И для того чтобы передавать на мост управления фазами сигналы для смены положения ключей, используются цифровые выводы. Ключ может находиться в двух состояниях: открыт и закрыт, и для каждого состояния используется вывод.

Также нужно описать все глобальные переменные. Описание начинается со строк:

int Ahol;

int Bhol;

int Chol;

int realposition;

int oldrealposition;

float tim;

float oldtim;

float IA;

float IB;

float IC;

float ID;

float IQ;

float sped;

float Us;

float Is;

float M;

float oldM;//прошлое положение

const int midholl = 25;

Если значения переменной будут целочисленные, то достаточно инициировать как "int". В данном случае нам достаточно знать округленное целое число, как для значения позиции от 1 до 6, так и для значений приходящих с датчиков холла. Тип "float" позволяет выводить точное значение числа, и это понадобится для значений токов, приходящих с каждой фазы, и соответственно для других переменных которые будут рассчитываться с помощью данных токов.

После обозначения всех величин сделаны две подпрограммы. Первая обозначается "void BKP()"

void BKP() {

float realIA;

float realIB;

float realIC;

realIA = ((IA - 512) / 17.5) / 0.095;//в амперах

realIB = ((IB - 512) / 17.5) / 0.095;

realIC = ((IC - 512) / 17.5) / 0.095;

ID = ((cos(sped)) \* ((pow((2.0 / 3.0), (1.0 / 2.0))) \* realIA + ((- 1.0 / 2.0) \* (pow((2.0 / 3.0), (1.0 / 2.0)))) \* realIB + ((- 1.0 / 2.0) \* (pow((2.0 / 3.0), (1.0 / 2.0))) \* realIC)) + (sin(sped)) \* ((pow((2.0 / 3.0), (1.0 / 2.0))) \* realIA + ((- 1.0 / 2.0) \* (pow((2.0 / 3.0), (1.0 / 2.0)))) \* realIB + ((- 1.0 / 2.0) \* (pow((2.0 / 3.0), (1.0 / 2.0))) \* realIC)));

IQ = (((-sin(sped)) \* (((pow((2.0 / 3.0), (1.0 / 2.0))) \* (pow((3.0 / 2.0), (1.0 / 2.0)))) \* realIB + (((pow((2.0 / 3.0), (1.0 / 2.0)))) \* (-pow((3.0 / 2.0), (1.0 / 2.0)))) \* realIC)) + ((cos(sped)) \* (((pow((2.0 / 3.0), (1.0 / 2.0))) \* (pow((3.0 / 2.0), (1.0 / 2.0)))) \* realIB + (((pow((2.0 / 3.0), (1.0 / 2.0)))) \* (-pow((3.0 / 2.0), (1.0 / 2.0)))) \* realIC)));

}

В данной программе сначала происходит считывание локальными переменными значения токов, которые приходят с датчиков, а потом происходят обратные преобразования, чтобы получить реальные данные. Значения которые приходят, измеряются от 0 до 1023, что соответствует значения от 0 до 5В. Также в этой подпрограмме происходят преобразования Парка-Кларка. Они отражены в полной версии программы, которая находится в приложении 1. Весь смысл данных преобразований был описан ранее и останавливаться на них не стоит. Лучше было бы оставить данные преобразования так, как они идут в оригинале, а именно матрицами, но по причине отсутствия глубоких знаний программирования красиво сделать это не получилось. Поэтому в программе используется уже финальная формула.

Следующая подпрограмма называется "void moment()", она рассчитывает момент, относительно которого будет рассчитываться напряжение широтно-импульсной модуляции, которая приводит в движение ротор. Для того чтобы иметь точку отсчёта пришлось взять приблизительные значения в начальный момент времени. Это отражается в строке с заданием начального напряжения. Таким образом был получен начальный момент. Для двигателя постоянного тока график момента изменяется линейно, поэтому был принят коэффициент "k" который задавал как быстро будет увеличиваться момент. Данная подпрограмма доводит значение момента до максимального значения.

void moment() {

const int k = 1.5;

const int Uo = 0.001;

if ((oldrealposition = realposition) &&(Is != 0)){

oldM = Is \* Uo / sped;

M = oldM;

}

if ((oldrealposition != realposition) && (Is != 0)) {

oldM = Is \* Us / sped;

M = k \* oldM;

}

}

После описания этих вспомогательных подпрограмм идет основная часть программы "void loop()". В первых шести строках прописывается что на аналоговые входа приходят пофазно токи и значения магнитного поля и все эти значения считываются. Далее идут расчеты по считываемым данным. Сначала находится значение результирующего вектора тока, а затем значение напряжение, которое подается на широтно-импульсную модуляцию.

void setup() {

pinMode (pinAL, OUTPUT);

pinMode (pinBL, OUTPUT);

pinMode (pinCL, OUTPUT);

Serial.begin(9600);

}

void loop() {

const int ugol = 60;

int newugol;

Ahol = analogRead(A0);

Bhol = analogRead(A1);

Chol = analogRead(A2);

IA = analogRead(A3);

IB = analogRead(A4);

IC = analogRead(A5);

BKP();

Is = pow((pow(ID, 2) + pow(IQ , 2)), (1.0 / 2.0));

moment();

Us = sped \* M / Is;

if (oldrealposition != realposition) {

tim = millis();

sped

= newugol / (tim - oldtim);

oldtim

= tim;

oldrealposition

= realposition;

}

if ((realposition - oldrealposition) > 0) {

newugol = 0 + ugol;

}

else {

newugol

= 0 - ugol;

}

if (Us > 225) {

Us = 225;

}

if (Us < 0) {

Us = 0;

}

Это была основная часть программы, следом идут определенные ограничения выходного значения напряжения и расчет скорости двигателя. Последнюю можно было сделать третьей подпрограммой, но это больше для красоты оформления и не принесло бы значительных изменений в работу программы. Возможно повлияло бы на быстродействие микроконтроллера, но не более.

Последняя часть программы:

//1

if ((Ahol > midholl) && (Bhol < midholl) && (Chol > midholl)) {

//Serial.println(1);

analogWrite(pinAH, Us);

digitalWrite(pinAL, LOW);

analogWrite(pinBH, 0);

digitalWrite(pinBL, HIGH);

analogWrite(pinCH, 0);

digitalWrite(pinCL, LOW);

realposition = 1;

}//прописать через миллис

//2

if ((Ahol > midholl) && (Bhol < midholl) && (Chol < midholl)) {

//Serial.println(2);

analogWrite(pinAH, Us);

digitalWrite(pinAL, LOW);

analogWrite(pinBH, 0);

digitalWrite(pinBL, LOW);

analogWrite(pinCH, 0);

digitalWrite(pinCL, HIGH);

realposition = 2;

}

//3

if ((Ahol > midholl) && (Bhol > midholl) && (Chol < midholl)) {

//Serial.println(3);

analogWrite(pinAH, 0);

digitalWrite(pinAL, LOW);

analogWrite(pinBH, Us);

digitalWrite(pinBL, LOW);

analogWrite(pinCH, 0);

digitalWrite(pinCL, HIGH);

realposition = 3;

}

//4

if ((Ahol < midholl) && (Bhol > midholl) && (Chol < midholl)) {

//Serial.println(4);

analogWrite(pinAH, 0);

digitalWrite(pinAL, HIGH);

analogWrite(pinBH, Us);

digitalWrite(pinBL, LOW);

analogWrite(pinCH, 0);

digitalWrite(pinCL, LOW);

realposition = 4;

}

//5

if ((Ahol < midholl) && (Bhol > midholl) && (Chol > midholl)) {

//Serial.println(5);

analogWrite(pinAH, 0);

digitalWrite(pinAL, HIGH);

analogWrite(pinBH, 0);

digitalWrite(pinBL, LOW);

analogWrite(pinCH, Us);

digitalWrite(pinCL, LOW);

realposition = 5;

}

//6

if ((Ahol < midholl) && (Bhol < midholl) && (Chol > midholl)) {

//Serial.println(6);

analogWrite(pinAH, 0);

digitalWrite(pinAL, LOW);

analogWrite(pinBH, 0);

digitalWrite(pinBL, HIGH);

analogWrite(pinCH, Us);

digitalWrite(pinCL, LOW);

realposition = 6;

/\*Serial.print(" Ahol=");

Serial.print(Ahol);

Serial.print(" Bhol=");

Serial.print(Bhol);

Serial.print(" Chol=");

Serial.print(Chol);

Serial.print(" IA=");

Serial.print(IA);

Serial.print(" IB=");

Serial.print(IB);

Serial.print(" IC=");

Serial.print(IC);

Serial.print(" ");\*/

}

}

Это прописывается сама ШИМ. Чтобы написать все правильно, сигналы подавались согласно схеме на рис. 18

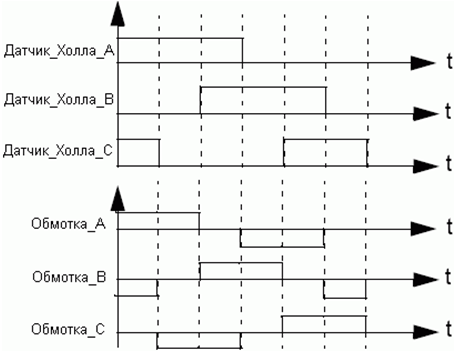


Рис. 18 График задания напряжений ШИМ

# Выводы

Проведя данную работу были получены новые знания о системах контрольных версий, строении и классификации датчиков и их важных элементах - операционных усилителях, изучен вопрос векторного управления и произведен на практике пуск бесколлекторного электропривода методом векторного управления. Все результаты были законспектированы и отражены в данной работе. Текст программы отражен в приложении, конечный вариант стенда представлен на рис.

Данные полученные с датчиков в виде осциллограмм показаны на рис.

# Список используемых источников

1. Калачев Ю. Н. Векторное регулирование (заметки практика) 2013 г-63с. http://www.privod-news.ru/docs/Vector\_Kalachev.pdf
2. http://cxem.net/beginner/beginner96.php
3. https://geektimes.ru/company/npf\_vektor/blog/269486/
4. Скот Чакон Pro Git изд.: Apress 2012г-286с.
5. http://rudatasheet.ru/datasheets/lm324/
6. https://www.youtube.com/watch?v=mpK\_MYb38zs
7. http://www.sensorica.ru/docs/d7\_overview.shtml
8. http://www.pvsm.ru/fizika/109643