Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

ИНСТИТУТ ЭНЕРГЕТИКИ И ЭЛЕКТРОТЕХНИКИ
(наименование института полностью)
Кафедра «Промышленная электроника»
(кафедра)
11.03.04. «Электроника и наноэлектроника»
(код и наименование направления подготовки, специальности)
«Промышленная электроника»
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему: Велокомпьютер с сенсорным управлением

Студент Пинчук А.В. _____
(И.О. Фамилия) (личная подпись)

Руководитель Прядилов А.В. _____
(И.О. Фамилия) (личная подпись)

Допустить к защите
Заведующий кафедрой _____ _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2018

Аннотация

Объем 92 с., 79 рис., 0 табл., 24 источников, 1 прил.

ВЕЛОКОМПЬЮТЕР С СЕНСОРНЫМ УПРАВЛЕНИЕМ

Цель работы: разработка велокомпьютера с возможностью замера скорости и пройденного расстояния

Задачи работы:

1. Обзор существующих решений
2. Расчет элементов в схему
3. Разработка аппаратной части модуля
4. Разработка программной части модуля
5. Сборка устройства

Работа состоит из пяти глав, в которых решены упомянутые задачи.

Разработка схем и печатных плат осуществлялась с помощью пакетов DipTrace, Компас V16, sPlan 7.0. Разработка программного кода велась в среде разработки Arduino IDE, а так же Nextion Editor.

Область применения данной работы, создан действующий макет.

Содержание

Введение

Необходимость в велокомпьютерах появилась ещё в 20 веке. Ведущая фирма, производитель измерительных приборов для автомобилей VDO в 1937 году был разработан и выпущен первый велокомпьютер, который по своему типу являлся механическим. Велокомпьютер – это электронное устройство для определения необходимых параметров велосипеда, таких как скорость и пройденное расстояние, а так же дополнительных параметров. Дополнительные параметры могут быть самыми различными. Это подсчёт каденса, средней, максимальной и минимальной скоростей, отображение фактического времени и времени в пути, пробег за сутки и общий пробег, температуры, влажности, давления, подсчёт пульса.

Однако востребованными велокомпьютеры стали лишь с 1990 года, когда в период резкого и мощного подъёма вело индустрию и использования велосипедов, как средство передвижения, так и в спорте в частности. Первый электронный велокомпьютер был разработан в 1981г, а первый в мире беспроводной велокомпьютер AERO 8 был создан именно VDO в 1991 году.

В тот период перед разработчиками возник ряд новых и непростых задач. Как создать устройство, которое обеспечивало бы велосипедиста как скоростными параметрами движения, а именно текущая, средняя, максимальная скорость, отклонение текущей скорости от средней, так и временными, такими как время в пути, секундомер, часы, таймер, но в тоже время обладающее компактными размерами, удобными органами управления и пользовательским интерфейсом, надежным креплением и корпусом, устойчивым к воздействию ударов, вибраций, неблагоприятных погодных условий.

Данные модели современных на тот момент времени электронных велокомпьютеров сразу же завоевали популярность во многих странах мира, собрали ряд наград от ведущих и авторитетнейших велосипедных изданий.

1. Состояние вопроса

1.1. Формулирование актуальности, цели и задач проекта

Актуальны ли данные, отображаемые велокомпьютером? Так ли они необходимы и для кого? В полупрофессиональном или профессиональном велоспорте спортсмен за один день проезжает от 100 до 250 км. Именно этим людям данные параметры могут быть как интересны и полезны в плане статистики тренировок, так и в плане ненанесения вреда здоровью.

Одним из таких важных для здоровья параметров, помимо скорости и пробега, что составляют основу велокомпьютера, является мониторинг каденса. Каденс – это частота оборотов педалей в минуту. Нормальный каденс велосипедиста составляет от 80 до 110 оборотов в минуту [1]. Если спортсмен не будет соблюдать данный параметр, то это чревато проблемами с суставами коленей, а так же появляется угроза инвалидности. Только из-за этих трёх параметров велокомпьютер уже представляет немалую ценность в полупрофессиональной и профессиональной среде.

Так же очень важен мониторинг пульса. Каждый велосипедист должен держать свой пульс в определённых границах. Существуют границы для эффективной тренировки сердечной системы, границы для эффективного похудения и границы повышенного износа сердечной системы.

Необходимость оставаться в этих границах свидетельствует о важности данного параметра, однако он очень затратен, так как требует высокой точности и обычно используется в качестве отдельного полноценного устройства.

Целью данной работы является разработка велокомпьютера с возможностью замера скорости и пройденного расстояния, по возможности создание действующего макета устройства.

Для достижения данной цели были поставлены, а затем выполнены следующие задачи:

1. Анализ существующих решений ;
2. Разработка схемы электрической принципиальной ;
3. Выбор и расчёт элементов схемы;
4. Разработка печатной платы;
5. Разработка программы устройства;
6. Изготовление устройства;
7. Отладка устройства.

1.2. Анализ исходных данных и известных решений

Самый простой велокомпьютер состоит из основного блока, датчика скорости и датчика оборотов (каденса).

Основной блок содержит в себе всю электронную часть велокомпьютера: процессор, электронную обвязку, дисплей, питание, а так же кронштейн крепления. На рисунке 1 изображён пример основного блока велокомпьютера.

Рисунок 1 – Пример основного блока велокомпьютера.

Датчик скорости представляет собой датчик холла в комплекте с магнитом, так же как и датчик оборотов (каденс). Примеры датчиков скорости и оборотов изображены на рисунке 2 и 3.

Условно велокомпьютеры можно разделить на проводные, беспроводные, GPS велокомпьютеры и smart-велокомпьютеры.

Проводные велокомпьютеры – это те компьютеры у которых все датчики подключены к основному блоку через провода. Как правило, обладает небольшим набором функций, имеет монохромный дисплей. Пример такого велокомпьютера изображён на рисунке 4.

Рисунок 2 – Датчик скорости.

Рисунок 3 – Датчик оборотов (каденса).

Плюсы данного вида: невысокая цена, простота конструкции, небольшие затраты ресурсов на питание, отсутствие помех от других устройств. Минусы данного вида: наличие проводов, проходящих по всему велосипеду, вероятность разрыва проводов.

Беспроводные велокомпьютеры – это те велокомпьютеры, в которых датчики передают всю собранную информацию по беспроводному интерфейсу в виде закодированного сигнала. Пример беспроводного велокомпьютера на рисунке 5.

Рисунок 4 – Проводной велокомпьютер.

Плюсы данного компьютера: отсутствие проводов, более компактный и предпочтительный внешний вид в сравнении с проводным велокомпьютером, отсутствие вероятности обрыва проводов. Отсутствие переходных контактов. Удобство съёма компьютера с велосипеда в целях предотвращения кражи. Минусы данного компьютера: необходимость в источнике питания для каждого датчика. Большие затраты на батарейки. Сильные помехи и ухудшение работы компьютера при проезде под высоковольтными линиями электропередач.

Рисунок 5 – Беспроводной велокомпьютер.

GPS-велокомпьютеры – это компьютеры со встроенным модулем GPS, что позволяет узнавать скорость велосипеда по данным спутников, просматривать при наличии специального приложения отрисовку пройденного маршрута, а так же видеть показания о положении над уровнем

моря, температуре и давлении вокруг. Один из таких велокомпьютеров изображён на рисунке 6.

Рисунок 6 – Велокомпьютер с модулем GPS.

Плюсы данного компьютера: Больший функционал. Возможность подключения к персональному компьютеру для возможности просмотра и анализа показаний. Возможность обновления прошивки. Минусы данного компьютера: Малое время работы из-за большего энергопотребления. Необходимость персонального компьютера для просмотра всех данных. Возможная потеря GPS сигнала. Большая стоимость.

Smart-велокомпьютеры – это компьютеры которые находятся на уровне, примерно равном уровню сотовых телефонов. У таких компьютеров есть операционная система, так же имеются датчики пульса, температуры, влажности и давления. Данные компьютеры могут прямо во время движения корректировать вашу нагрузку на сердце и по собранной информации составлять режим вашей езды. Помимо этого происходит запись маршрутов и параметров поездки для самостоятельного анализа. Пример такого велокомпьютера показан на рисунке 7.

Плюсы данного компьютера это высокая производительность. Высокая многофункциональность. Интуитивно понятный интерфейс. Сенсорное управление. Возможность контроля пульса. Минусы данного компьютера: Очень высокая стоимость. Более уязвимое к падениям исполнение. Большое энергопотребление.

Рисунок 7 - Smart-велокомпьютер.

2. Аппаратная часть

2.1. Разработка схемы электрической принципиальной

Для разработки схемы электрической принципиальной был выбран пакет программ Dip Trace, а именно в программе Schematic.

Для начала нужно обратиться к структурной схеме устройства и понять принцип действия устройства, для переноса принципов в электронное устройство с точными преобразованиями и работой. Структурная схема устройства изображена на рисунке 8.

Принцип работы велокомпьютера следующий. Обратимся к структурной схеме, изображённой на рисунке 8. Как видно на схеме, через разъём, гнездового типа XS1, подаётся напряжение с аккумулятора, на устройство, так как согласно одному из пунктов задания, питание должно быть автономным.

Далее оно поступает уже на плату защиты аккумулятора. Данная плата защиты должна иметь следующий функционал, это защиту от глубокого разряда, от перезаряда, от короткого замыкания и переплюсовки, всё это необходимо для безопасной эксплуатации устройства, и для защиты аккумулятора от быстрой деградации.

Затем уже напряжение поступает на транзисторный ключ, который коммутирует плюс в схеме. Так же помимо ключа, напряжение с аккумулятора поступает и на тактовую кнопку с двумя контактами, нормально-замкнутый и нормально-разомкнутый. Нормально-разомкнутый контакт используется для включения велокомпьютера и всё, а нормально-замкнутый используется в качестве подачи сигнала на микроконтроллер для дальнейшей его обработки. Использоваться будет сигнал логического нуля, для выключения велокомпьютера или перевода его в режим настройки и вывода из данного режима. А транзисторный ключ будет выполнять функцию самоблокировки и поддерживать питания в схеме пока оно нужно и

разрывать его, когда уже будет не нужно, без участия механических контактов. Самоблокировка – это определённый принцип включения некого устройства или агрегата кратковременным замыканием контактов, например кнопки, а так же по аналогии выключении при кратковременном размыкании контактов и поддержании работы без всяких фиксаций, например тумблеров или кнопок.

После транзисторного ключа, напряжение с аккумулятора приходит на повышающий DC/DC преобразователь, то есть преобразователь, который преобразует из постоянного в постоянное напряжение, но большее по номиналу от входного, поэтому данный преобразователь и называется повышающим. Преобразователь же в свою очередь выдаёт на своём выходе стабильные 5 вольт, в основную сеть питания велокомпьютера.

Так же с выхода платы защиты аккумулятора берётся напряжение на второй транзисторный ключ, для измерения напряжения аккумулятора

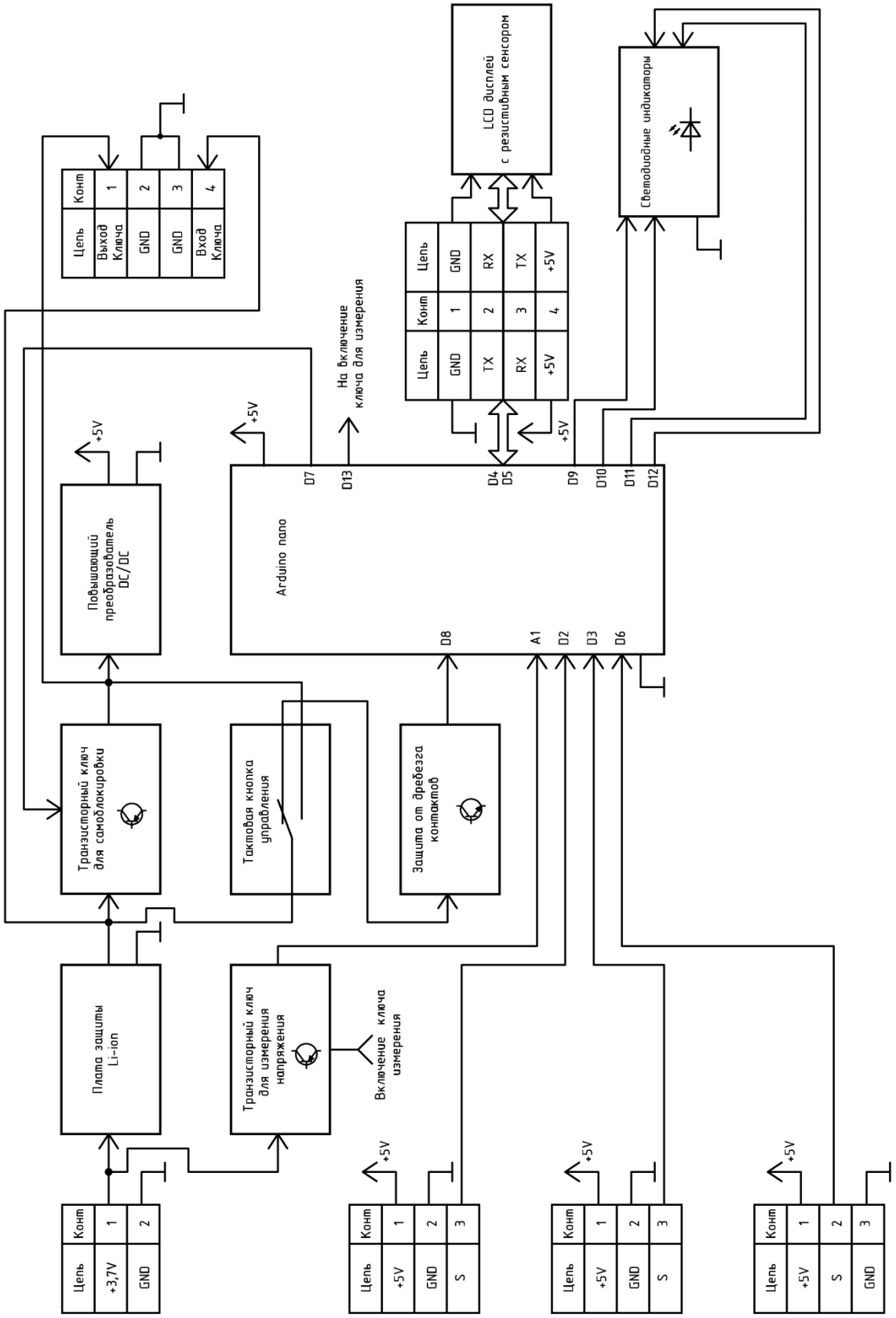

Рисунок 8 – Структурная схема.

микроконтроллером в схеме. Нужно данное действие для того, что бы отслеживать велокомпьютером уровень заряда аккумулятора и демонстрировать это пользователю данного устройства.

Если дальше изучать структурную схему, изображённую на рисунке 8, то можно увидеть что после нормально-закрытого контакта тактовой кнопки идёт плата защиты от дребезга контактов [2]. Ни для кого не секрет что у механических контактов имеется дребезг в момент их замыкания. Дребезг контактов – это явление, которое возникает в электронных контактах, коммутирующихся механическим способом, при котором они вместо некоторого стабильного переключения производят случайные многократные неконтролируемые замыкания и размыкания контактов (происходит в момент переключения, приблизительно в течение 40—100 мс). Вот поэтому в данном случае необходима защита от дребезга контактов.

Как видно на схеме, изображённой на рисунке 8, в качестве микроконтроллера планируется использовать платформу Arduino и открытую среду разработки Arduino IDE для написания программы, что бы данное устройство функционировало, как это требует техническое задание. Микроконтроллер в Arduino будет исполнять все основные функции велокомпьютера, измерения, а так же вычисления.

Из схемы видно, что разъёмы штырькового типа XP2, XP3, используются для коммутации сигнала на микроконтроллер. Сигнала с датчиков Холла, которые подключаются к этим разъёмам? А также подведения к ним питающего напряжения 5 вольт.

Так же разъём XP4 служит для коммутации с микроконтроллером датчика температуры и влажности, для получения информации, а так же для подведения питания к данному датчику.

Следует обратить внимание на разъём штырькового типа XP1. Данный разъём необходим, для калибровки измерения напряжения микроконтроллером, а так же последующей корректировки из-за небольшого падения напряжения на транзисторном ключе, что обуславливает его P-N

переход. Как известно для транзистора с кремниевой структурой это 0,6 – 0,7 вольта, а для транзистора с германиевой структурой это 0,1 – 0,2 вольта.

Далее через соединённый разъём X1, подключается дисплей с резистивным сенсором. Через данный разъём происходит коммутация на дисплей и сенсор, в данном случае необходимо 5 вольт, а так же присутствует шина для обмена информации дисплея, сенсора и микроконтроллера.

Если в велокомпьютере присутствуют датчики Холла, то необходимо облегчить их установку для этого необходимо при их установке на велосипед визуально контролировать срабатывает ли датчик Холла или нет, для этого было решено применить сравнивающее устройство что, и изображено на рисунке 8, для его работы необходим источник эталонного напряжения для последующих сравнений. Для визуальной части на схеме предусмотрены светодиодные индикаторы.

А теперь давайте рассмотрим принцип работы велокомпьютера более подробно на основе схемы изображённой на рисунке 8. Через разъём XS1 подаётся аккумуляторное питание на плату защиты аккумулятора, затем оно поступает на транзисторный ключ, который в закрытом состоянии не пропускает напряжение дальше в схеме. Так устройство находится в выключенном состоянии, а потребление тока в схеме равно, размеру номинального значения тока саморазряда, указанный в характеристиках аккумулятора. Так же параллельно ключу подключается тактовая кнопка, нормально-разомкнутыми контактами.

Когда необходимо включить велокомпьютер, нужно нажать на тактовую кнопку и поддержать её нажатой около 1-2 секунд, после чего можно отпустить и устройство останется включенным. Это работает следующим образом. При нажатии на тактовую кнопку, мы закорачиваем транзисторный ключ, тем самым подаём напряжение дальше на повышающий преобразователь. Который преобразует аккумуляторное напряжение в стабильные 5 вольт, которое уже поступает в основную

питающую цепь устройства, далее микроконтроллер включается, начиная выполнение программы и один из пунктов программы, является, открыть транзисторный ключ. В последующем напряжение уже проходит, через него и это будет продолжаться до тех пор, пока ключ не будет снова закрыт. Из этого следует, что для включения велокомпьютера нужно всего лишь нажать на кнопку и в последующем отпустить и всё, для этого в схеме используется принцип самоблокировки.

Велокомпьютер функционирует и через разъём XP4 получает данные от датчика температуры и влажности и отправляет их на дисплей, а через разъёмы XP2 и XP3 микроконтроллер получает данные с датчиков Холла, если велосипед движется или ничего не получает, если он стоит. Далее эти показания обрабатываются, подсчитывается скорость и обороты и так же отправляется в дисплей через шину данных. Если велокомпьютер только монтируется на велосипед, необходимо правильно установить датчики Холла. А для этого нам необходимо включить сравнивающее устройство, питание которого подаётся с цифрового выхода микроконтроллера. При включенном сравнивающем устройстве, светодиоды будут показывать, сработали датчики холла или нет и уже можно без проблем установить или заменить их, после чего питание сравнивающего устройства выключить.

В последующем, когда велокомпьютер необходимо выключить, нужно нажать на кнопку включения и держать её около 5 секунд пока не начнёт светиться красный светодиод, когда он засветился, мы отпускаем кнопку и велокомпьютер отключается.

Так же, ещё будет необходимость выставлять датчики Холла и проводить другие настройки велокомпьютера, для этого будет предусмотрен режим настройки и калибровки. Для его активации нажимаем на кнопку включения и держим около 3 секунд, пока не начнёт светиться зелёный светодиод, когда появилось зелёное свечение, нужно отпустить кнопку и можно производить настройки и сохранять их. После того как всё было сделано, необходимо снова нажать на кнопку и подождать около 3 секунд,

пока не погаснет зелёный светодиод, как это произошло, можно снова использовать велокомпьютер по его прямому назначению, пока не появится надобность его выключить.

На основе проделанного выше анализа была разработана схема электрическая принципиальная в программе Schematic из пакета Dip Trace. Схема электрическая принципиальная изображена на рисунке 9.

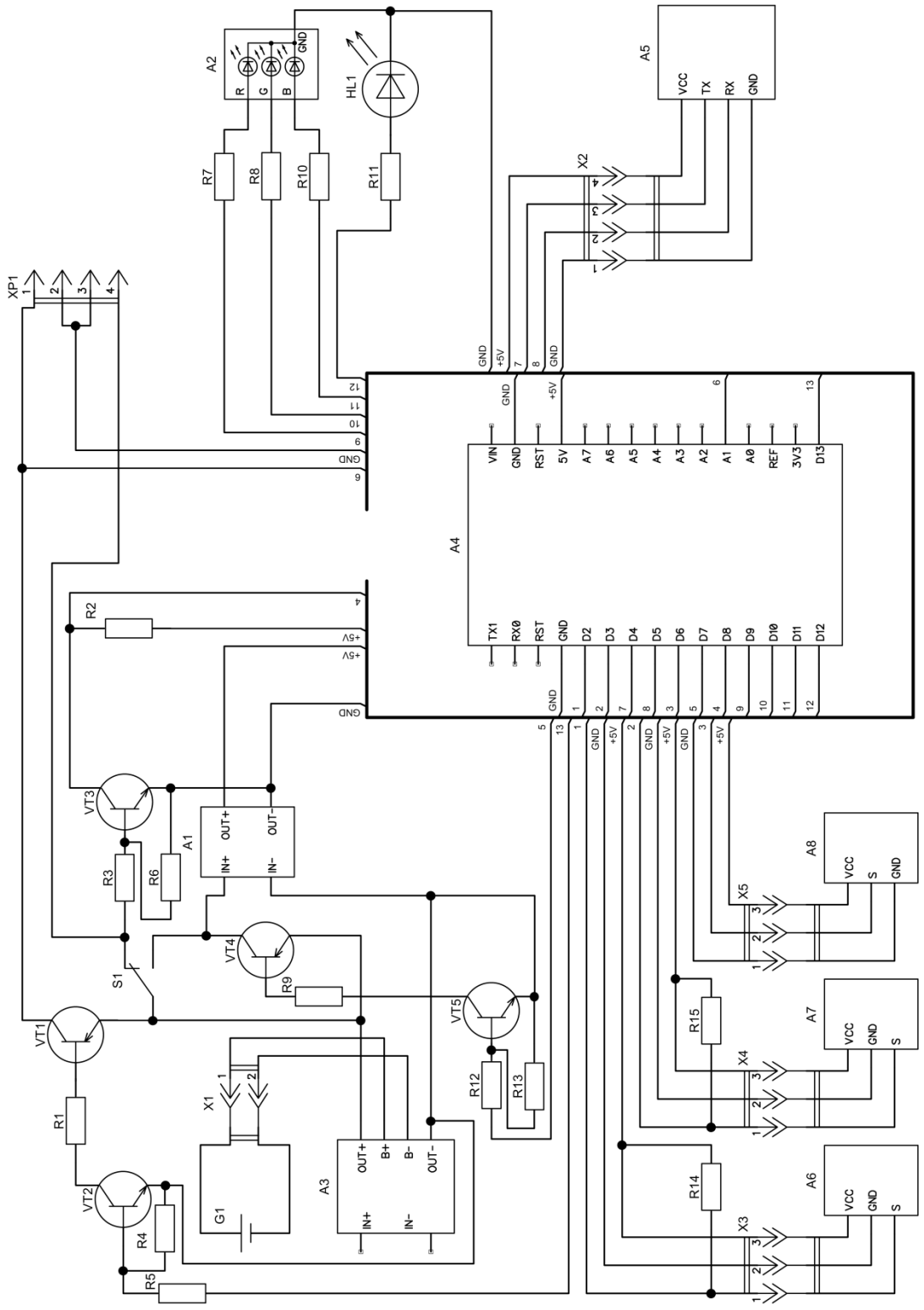

Рисунок 9 – Схема электрическая принципиальная.

2.2. Выбор и расчет элементов

Согласно техническому заданию, питания велокомпьютера должно быть автономным, в связи с этим для питания будет использоваться аккумулятор или несколько аккумуляторов соединенных последовательно или параллельно. На данный момент потенциально могут подойти несколько аккумуляторов:

1. Никель-кадмиевые (Ni-Cd);
2. никель-металлогидридные аккумуляторы (Ni-MH);
3. литий-ионные аккумуляторы (Li-ion);
4. литий-полимерные аккумуляторы (Li-pol).

Рассмотрим все аккумуляторы и их свойства по порядку. Никель-кадмиевые аккумуляторы состоят из гидроксида никеля и кадмия, а так же электролитом служит раствор жидкого гидроксида калия, который является очень щелочным, поэтому их именуют щелочными аккумуляторами [3]. Существуют три основных вида никель-кадмиевых аккумуляторов, а это негерметичные с ламельными электродами и спеченными электродами, а так же герметичные.

Как правило, первые два вида аккумуляторов используются для питания шахтных электровозов, подъемников, стационарного оборудования, для запуска дизелей и авиационных двигателей. Но в нашем случае нужны аккумуляторы герметичного типа, раньше они использовались в сотовых телефонах, а сейчас, же они используются в портативных радиостанциях, детских игрушках, портативных и бытовых устройствах. Никель-кадмиевые аккумуляторы имеют следующие характеристики:

1. Рабочее напряжение от 1,0 до 1,35 Вольта;
2. нормальный ток заряда от 0,1 до 1 от номинальной ёмкости аккумулятора;
3. саморазряд около 10% в месяц;
4. диапазон рабочих температур от -50 до +40 ℃;

5. срок службы аккумуляторов от 100 до 900 циклов заряда-разряда.

Из плюсов данных аккумуляторов можно отметить:

1. Низкую стоимость никель-кадмиевых аккумуляторов;
2. возможность отдавать наибольший ток нагрузки;
3. широкий диапазон рабочих температур;
4. возможность быстрой зарядка путём заряжания током номинальной ёмкости аккумулятора;
5. достаточно большое количество циклов заряда-разряда (при правильной эксплуатации).

Из минусов можно отметить:

1. Низкое напряжение аккумулятора;
2. быстрая скорость разряда под нагрузкой;
3. относительно высокий уровень саморазряда, примерно 8-10%;
4. содержит токсичный кадмий и по своей сути является токсичным аккумулятором;
5. эффект памяти, что заставляет перед зарядкой аккумулятора, полностью разрядить его;
6. после длительного хранения, для восстановления ёмкости аккумулятора, необходимо совершить не менее 5 циклов заряда-разряда.

Типоразмеры никель-кадмиевых аккумуляторов можно увидеть на рисунке 10.

Теперь рассмотрим никель-металлогидридные аккумуляторы [4]. Активным материалом отрицательного электрода никель-металлогидридного аккумулятора является интерметаллид, который обратимо собирает водород, фактически данный электрод является водородным электродом. Их область применения весьма разнообразна, это и космическая промышленность и медицина и автомобильная промышленность, но так, же тот типоразмер аккумуляторов, который нас интересует, используется в бытовых приборах и

игрушках [5]. Никель-металлогидридные аккумуляторы имеют следующие характеристики:

1. Рабочее напряжение от 1,0 до 1,25 Вольта;
2. нормальный ток заряда от 0,1 до 0,75 от номинальной ёмкости аккумулятора;
3. саморазряд до 100% в год;
4. диапазон рабочих температур от -60 до +55 ℃;
5. срок службы аккумуляторов от 300 до 500 циклов заряда-разряда.

Из плюсов данных аккумуляторов можно отметить:

1. Данные аккумуляторы не содержат токсичный кадмий и по своей сути не являются токсичными;
2. имеют гораздо меньший эффект памяти по сравнению с никель-кадмиевыми аккумуляторами;
3. данный тип аккумуляторов имеет гораздо большую ёмкость по сравнению с никель-кадмиевыми аккумуляторами;
4. широкий диапазон рабочих температур аккумуляторов данного типа.

Из минусов можно отметить:

1. Более дорогой тип аккумуляторов по сравнению с никель-кадмиевыми аккумуляторами;
2. величина саморазряда намного выше по сравнению с никель-кадмиевыми, примерно в полтора раза;
3. после 200-300 циклов заряда-разряда данного типа аккумулятора рабочая ёмкость начинает снижаться;
4. данный вид аккумуляторов имеет ограниченный срок службы;
5. Низкое напряжение аккумулятора.

Типоразмеры никель-металлогидридных аккумуляторов можно увидеть на рисунке 10. Так как для никель-металлогидридных и для никель-кадмиевых аккумуляторов одинаковые стандартизированные типоразмеры, поэтому используется один рисунок.

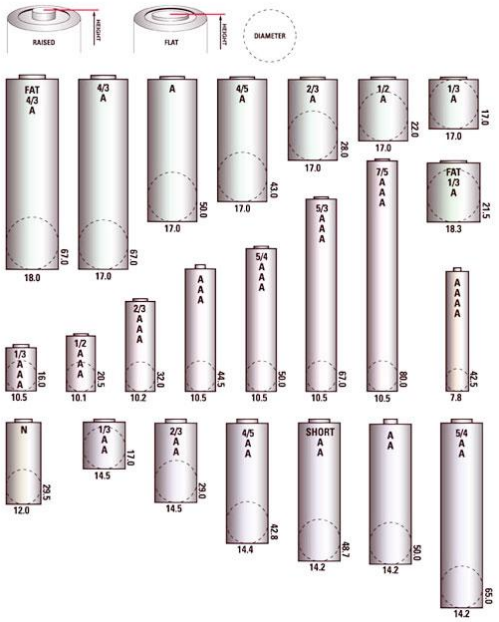

Рисунок 10 – Типоразмеры аккумуляторов.

Далее рассмотрим литий-ионные аккумуляторы [6]. В качестве отрицательного электрода применяется углеродистый материал, в который обратимо внедряются ионы лития. Активным материалом положительного электрода обычно служит оксид кобальта, в который также обратимо внедряются ионы лития. Электролитом является раствор соли лития в неводном апротонном растворителе. Данные аккумуляторы, относительно недавно использовались в сотовых телефонах, на данный момент используются в портативных зарядных устройствах, электронных сигаретах и других приборов, требующих высокую токоотдачу. Литий-ионные аккумуляторы имеют следующие характеристики:

1. Рабочее напряжение от 2,5 до 4,25 Вольта;
2. нормальный ток заряда от 0,2 до 0,1 от номинальной ёмкости аккумулятора;
3. саморазряд при температуре 25 °С и заряде 100% ≈1,6 % в месяц;
4. диапазон рабочих температур от -20 до +60 °С;

5. срок службы аккумуляторов 600 циклов заряда-разряда до достижения 80 % номинальной ёмкости, всего около 1000 циклов.

Из плюсов данных аккумуляторов можно отметить:

1. Отсутствие эффекта памяти, можно подзаряжать при необходимости в любое время;
2. достаточно высокая ёмкость данного типа аккумуляторов;
3. очень низкий уровень саморазряда;
4. возможность быстрой зарядки аккумуляторов данного типа;
5. высокое напряжение относительно других аккумуляторов.

Из минусов можно отметить:

1. Высокая стоимость данных аккумуляторов;
2. ограниченный срок службы;
3. сокращение времени работы при температуре ниже нуля;
4. деградация аккумулятора при его неиспользовании, за исключением хранения в специальных условиях;
5. требование в наличии специальной платы защиты от перезаряда и воспламенения.

Типоразмеры литий-ионных аккумуляторов указаны на рисунке 11.

Рассмотрим литий-полимерные аккумуляторы. Анодом у литий-полимерных аккумуляторов служит углеродистый материал, в который обратимо внедряются ионы лития. Активными материалами положительных электродов являются оксиды ванадия, кобальта или марганца. Электролитом является или раствор соли лития в неводных апротонных растворителях, заключенный в микропористую полимерную матрицу, или полимер, пластифицированный раствором соли лития в апротонном растворителе.

В настоящее время повсеместно используются в телефонах и планшетах, а так же по возможности в других проектах, требующие маленькие размеры аккумуляторов, например маленькие радиоуправляемые вертолёты.

Рисунок 11 – Типоразмеры литий-ионных аккумуляторов.

Литий-полимерные аккумуляторы от литий-ионных практически ничем не отличаются за исключением того, что другой технологией изготовления и имеют практически одинаковые характеристики [7].

Плюсы литий-полимерного аккумулятора следующие:

1. Увеличенный срок службы аккумуляторов от 2000 до 5000 циклов заряда-разряда;
2. возможность придания аккумулятору разнообразных форм, в том числе и совсем плоских;
3. уменьшения веса относительно литий-ионных аккумуляторов
4. высокий уровень безопасности
5. более высокая ёмкость при тех же размерах относительно литий-ионных аккумуляторов;
6. более низкий уровень саморазряда относительно литий-ионных аккумуляторов.

Минусы литий-полимерного аккумулятора следующие:

1. Более высокая стоимость производства относительно и без того дорогих литий-ионных аккумуляторов;
2. Деградация аккумулятора при его неиспользовании, за исключением хранения в специальных условиях, намного меньше относительно литий-ионных аккумуляторов;

3. требование в наличии специальной платы защиты от перезаряда и воспламенения.

Литий-полимерные аккумуляторы получили распространение в основном прямоугольной формы, поэтому их типоразмеры не стандартизированы и в характеристиках к ним приводятся их размеры, которые выражаются в длине, ширине, а так же в высоте.

По итогам и рассмотрении всех плюсов и минусов потенциальных аккумуляторов был выбран литий-ионный аккумулятор типоразмера 18650. Причины, послужившие выбору данного аккумулятора, послужили следующие самая главная это достаточно высокое напряжение для преобразований, что позволяет использовать всего один аккумулятор или несколько в параллельном соединении для увеличения суммарной ёмкости. Так же данный аккумулятор позволяет на протяжении почти всей работы поддерживать достаточный ток нагрузки до самого разряда аккумулятора. Следует заметить, что данный аккумулятор один из самых распространённых на сегодняшний день, что позволяет с лёгкостью купить его и по снисходительной цене.

По итогам был приобретён аккумулятор фирмы Поиск, имеющие следующие характеристики:

1. Номинальное напряжение - 3,7 Вольта;
2. ёмкость аккумулятора - 2400 миллиампер/часов;
3. встроенная защита от перезаряда – есть;
4. тип аккумулятора - литий-ионная (Li-ion);
5. типоразмер – 18650;
6. более 500 циклов заряда-разряда, без эффекта памяти.

Внешний вид аккумулятора изображён на рисунке 12.

Так как аккумуляторы были выбраны литий-ионные, для них необходима плата защиты в устройстве. Несмотря на то, что в данных аккумуляторах имеется встроенная защита, для создания более гибких возможностей замены аккумуляторов необходимо предусмотреть защиту.

Тогда будет возможность поставить аккумуляторы без встроенной защиты, а так же будет обеспечена дополнительная защита, что с литий-ионными аккумуляторами не лишнее.

Рисунок 12 – Внешний вид аккумуляторов фирмы Поиск.

При изучении доступного ассортимента в каталогах радиомагазинов, был обнаружен, уже готовый модуль TP4056 [8], внешний вид данного модуля изображён на рисунке 13. Поэтому было решено использовать уже готовый модуль из-за его компактных размеров, благодаря поверхностного монтажа и так же цена готового модуля получается дешевле, чем установка таких же элементов в схему.

Модуль имеет следующие характеристики:

1. Напряжение питания – от 4,5 до 8,0 Вольт;
2. потребляемый ток не более – 1,0 Ампера;
3. программируемый ток заряда – от 0,1 до 1,0 Ампера;
4. напряжение окончания заряда аккумулятора – 4,2 Вольта;
5. встроенная защита от переплюсовки;

6. встроенная защита от перегрузки по току – ограничение 3 Ампера;
7. встроенная защита от глубокого разряда аккумулятора – 2,4 Вольта;
8. наличие индикации (заряда) путём красного светодиода;
9. наличие индикации (конец зарядки) путём зелёного светодиода;
10. наличие USB разъёма - типа Micro-USB;
11. размеры модуля 15х25,5 миллиметров;
12. Вес модуля 2,1 грамм.

Схема данного модуля взята с сайта производителя и изображена на рисунке 14. Данный модуль собран на микросхеме TP4056, от сюда и название данного модуля. Данная микросхема представляет из себя контроллер зарядки литий-ионных аккумуляторов со встроенным термодатчиком. Выпускается с 2003 года [9]. Принцип заряда у данной микросхемы линейный и зарядка происходит по принципу постоянное напряжение/постоянный ток. Микросхема выпускается в корпусе SOP – 8,

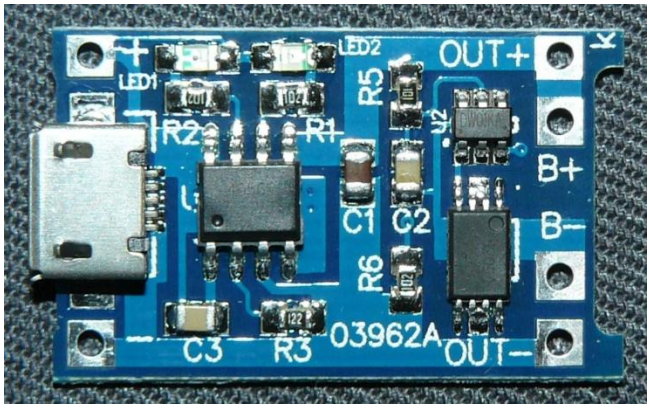

Рисунок 13 – Внешний вид модуля защиты TP4056.

в нижней части микросхемы присутствует металлическая площадка электронно не связанная с контактами микросхемы и служит для снятия выделяемого тепла, что позволяет данной микросхеме обеспечивать ток заряда до 1,0 ампера. Так же тот заряда можно понизить до 0,1 ампера, для

этого используется токозадающий резистор, на рисунке 14, R3 по умолчанию резистор установлен на ток заряда 1,0 ампер.

Данный модуль не предусматривает использование встроенных в аккумуляторы датчики температуры, поэтому для отключения этой функции 1 вывод микросхемы соединён с минусом питания. Так же микросхема может уходить в сон и выходить из него и данная функция так же не была выведена в этом модуле, поэтому 8 вывод микросхемы соединён с плюсом питания, так как из сна микросхема выходит при поступлении на 8 вывод логической единицы.

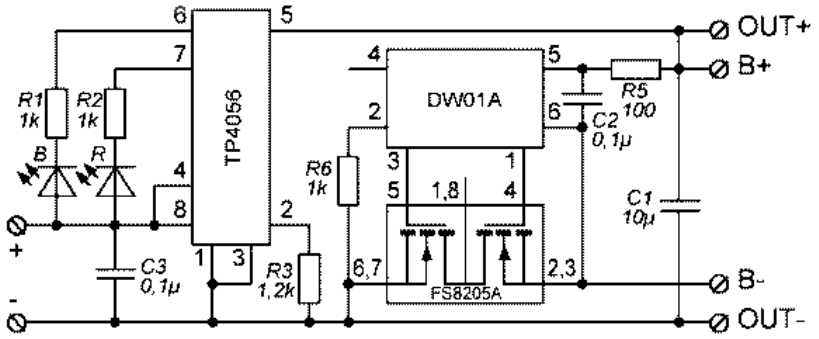

Рисунок 14 – Схема модуля TP4056.

Но хоть и микросхема имеет защиту от перезарядки аккумулятора, на этом её возможности заканчиваются, а возможности модуля защиты от глубокого разряда и короткого замыкания и ограничения потребляемого тока были реализованы на микросхеме DW01 и сдвоенном полевом ключе MOSFET FS8205.

Микросхема DW01 является контроллером для литий-ионного аккумулятора и следит за потреблением тока и переплюсовкой нагрузки, а так же достижения аккумулятором напряжение в 2,4 Вольта, для его отключения. Защита осуществляется при помощи двух полевых транзистора MOSFET N – типа но, к сожалению, данные транзисторы не входят в состав

микросхемы, для этого и используется сдвоенный полевой ключ FS8205, который отвечает всем требуемым характеристикам для полноценной работы защиты.

Рассмотрим назначение выводов данного модуля. Вывода IN+ и IN- служат для подключения входного (зарядного) напряжения, если необходимо использовать другой источник питания, а не от USB разъёма - типа Micro-USB, использовать их не обязательно. Так же эти вывода позволяют объединить несколько таких модулей защиты для работы от одного источника питания, но для этого необходимо обеспечить мощный блок питания что бы избежать просадок напряжения и получить необходимый ток. Так же вывода B+ и B- являются входом встроенной защиты и к ним подключается сам аккумулятор или два параллельно. А так же вывода OUT+ и OUT- служат выходом защиты и подключения нагрузки, в случае срабатывания защиты, аккумулятор будет отключён от нагрузки.

Далее рассмотрим модуль повышающего преобразователя. Данный модуль необходим в схеме, так как наши аккумуляторы не смогут обеспечить напряжение в 5 Вольт, а в дальнейшем нам будет необходимо напряжение в 5 Вольт, так это одно из самых распространённых напряжений для работы с микросхемами, датчиками и так далее.

Так же во время анализа каталогов радиомагазинов и соответствующих сайтов, был найден уже готовый модуль повышающего преобразователя MT3608 который имеет следующие характеристики:

1. Входное напряжение – от 2 до 24 Вольт;
2. максимальное выходное напряжение – 28 Вольт;
3. максимальный входной ток – 2 Ампера;
4. КПД преобразователя $\leq$ 93%;
5. размеры модуля 17x36 миллиметров;
6. вес модуля 1,4 грамма.

Внешний вид модуля изображён на рисунке 15, а так же схему данного модуля можно увидеть на рисунке 16. Схема так же была взята с сайта

производителя. Модуль имеет название MT3608 из-за использованной в нём микросхемы с таким названием что и видно если обратить внимание на рисунок 15. Данная микросхема представляет из себя регулируемый, повышающий преобразователь напряжения [10]. Называется он повышающим, потому что на его выходе не может быть напряжение меньше, чем на входе, либо примерно столько же, либо больше то, что нам нужно в нашей схеме.

Микросхема производится в корпусе SOT23. Так же на рисунке 16

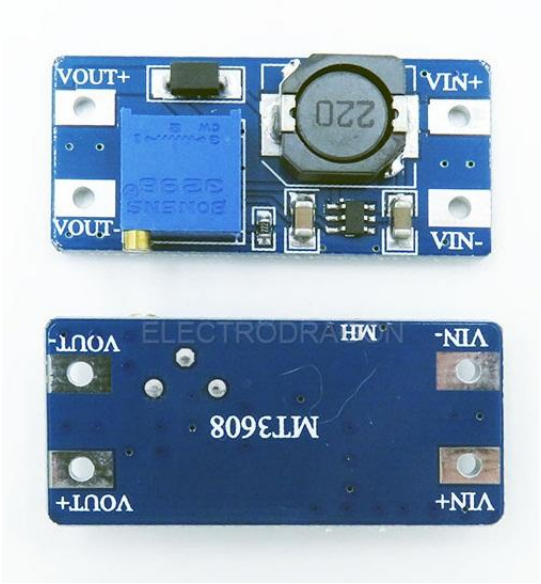

Рисунок 15 – Внешний вид модуля MT3608.

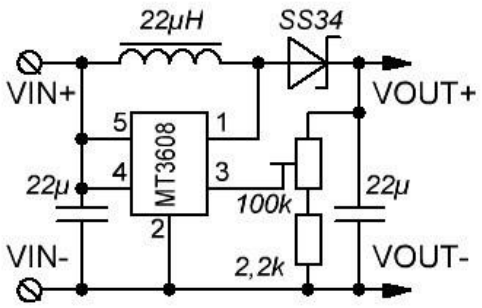

Рисунок 16 – Схема модуля MT3608.

необходимо обратить внимание, что в схеме присутствует подстроечный, много оборотистый резистор, что позволяет настраивать необходимое на выходе напряжение. В остальном можно сказать что схема преобразователя весьма классическая. Однако стоит подметить один незначительный минус, на данном модуле на выходе имеются достаточно значительные пульсации напряжения, но это можно легко решить. На выходе сразу после модуля установить керамический конденсатор на 10 микрофарад и электролитический 220 или 470 микрофарад, с допустимым максимальным напряжением на них, это приемлемо решит данную проблему.

Так же согласно схеме нам необходим микроконтроллер для осуществления всех вычислений, после изучения технического задания и каталогов, был выбран готовый модуль Arduino nano, внешний вид которого можно увидеть на рисунке 17. Данный модуль был выбран из-за своих компактный размеров, уже установленной на него обвязке и подходящего для устройства функционала, который рассмотрим ниже, а так же небольшой стоимости. Схему модуля изображена на рисунке 18, взята с сайта производителя.

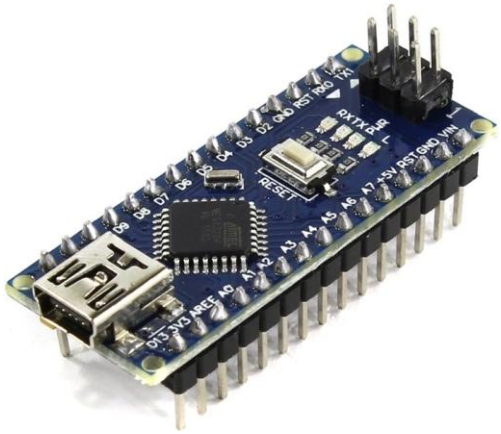

Рисунок 17 – Внешний вид модуля Arduino nano.

Как видно из рисунка 18, модуль состоит из микроконтроллера Atmega328, в который заливается программа и происходит её выполнение, а так же подключенная к ней обвязка из кварцевого резонатора и конденсаторов, которая является частотозадающей цепью для микроконтроллера.

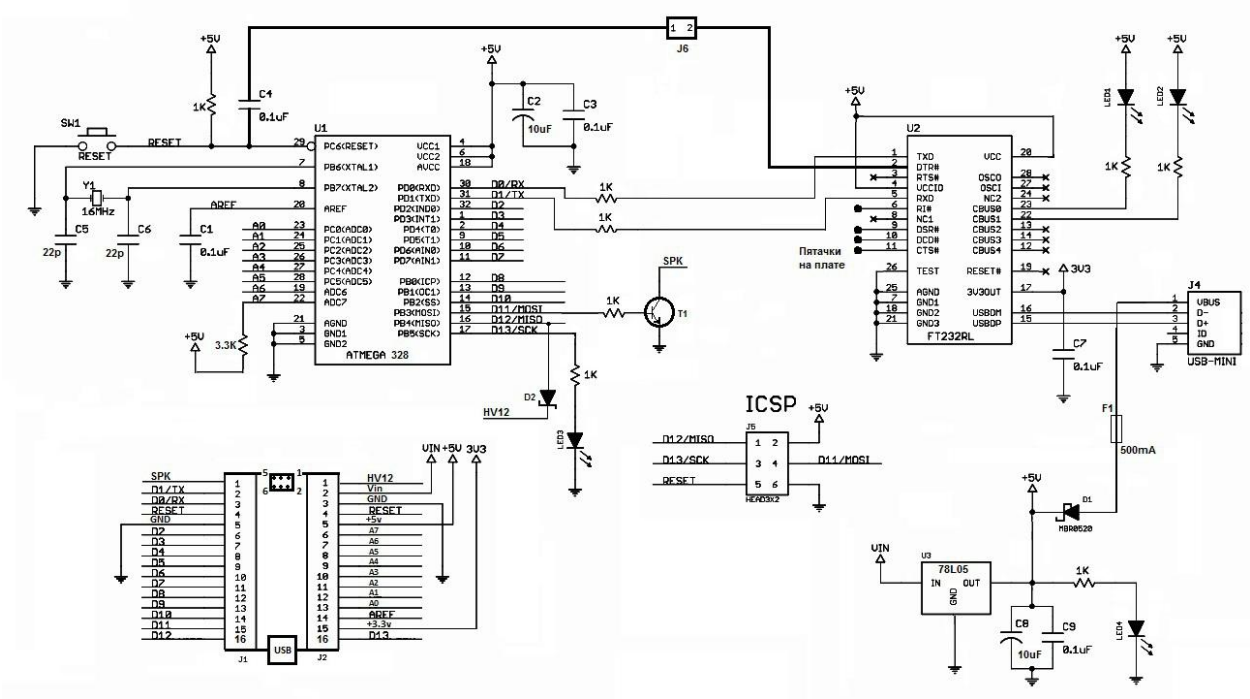

Рисунок 18 – Схема модуля Arduino nano.

Так же имеются светодиоды сигнализирующие о наличии питания на микроконтроллере, о наличии логической единицы на 13 выводе модуля, а так же ещё два с маркировкой RX и TX,по которым можно отслеживать передачу и приём сигналов по последовательному порту. Программу микроконтроллер получает по последовательному порту, который интегрирован в микроконтроллер. Что бы эту программу в последовательный порт отправлять, с компьютера, в данном модуле использована микросхема преобразователь FT232 [11]. Данная микросхема представляет из себя одночиповый переходник из USB в асинхронный последовательный интерфейс UART, протокол USB полностью интегрирован в данную

микросхему. Так же присутствует линейный стабилизатор, для питания модуля большим от 5 вольт напряжением. А так же имеется разъём ICSP для перепрошивки микроконтроллера arduino загрузчиком. Данный модуль имеет следующие характеристики, взятые с сайта производителя указанные на рисунке 19.

Краткие характеристики

Микроконтроллер	Atmel ATmega168 или ATmega328
Рабочее напряжение (логическая уровень)	5 В
Входное напряжение (рекомендуемое)	7-12 В
Входное напряжение (предельное)	6-20 В
Цифровые Входы/Выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	8
Постоянный ток через вход/выход	40 мА
Флеш-память	16 Кб (ATmega168) или 32 Кб (ATmega328) при этом 2 Кб используются для загрузчика
ОЗУ	1 Кб (ATmega168) или 2 Кб (ATmega328)
EEPROM	512 байт (ATmega168) или 1 Кб (ATmega328)
Тактовая частота	16 МГц
Размеры	1.85 см x 4.2 см

Рисунок 19 – Характеристики модуля Arduino nano.

Рассмотрим назначенные выводов модуля. Для начала обратим внимание на рисунок 20, на котором изображены все вывода модуля и их названия. Начнём слева-сверху, первое, что мы видим, это вывода TX и RX [12].

Это последовательная шина, в данном модуле, а данные вывода используются для получения и передачи данных по интерфейсу UART. Далее вывод Reset используется для возможности перезагрузки микроконтроллера, если нет возможности это сделать нажатием на кнопку Reset, которая так же установлена на плате модуля. Перезагрузка модуля происходит, при подаче низкого уровня сигнала на вход, то есть логического нуля. Затем вывод с названием Ground, используется для общего минуса всего модуля. Потом идут вывода с названием Digital, это означает, что данные вывода цифровые и могут настраиваться как на вход, так и на выход. Так же, данные вывода являются одними из основных на данном модуле. Они нумеруются, начиная

от нуля, и обозначаются английской буквой D, после которой сразу идёт число вывода. Всего на модуле 14 цифровых выводов, из них 6 могут генерировать широтно-импульсную модуляцию, с разрешением 8 бит. Так же модуль оснащён разъёмом Mini-B USB, для осуществления прошивки микроконтроллера или получения данных с того же микроконтроллера при обработке им каких-то сигналов.

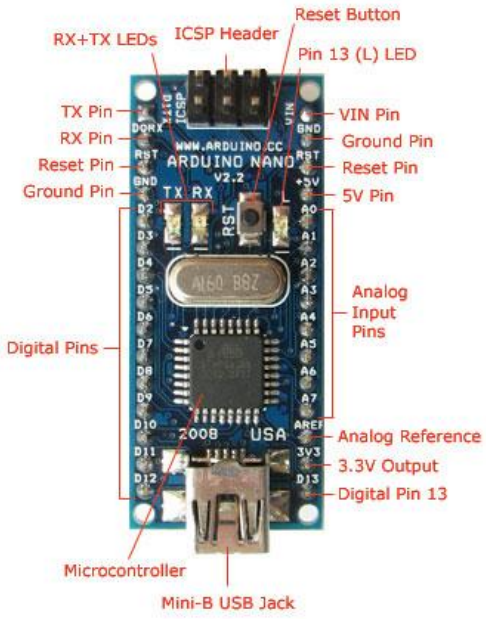

Рисунок 20 – Расположение и наименование выводов Arduino nano.

С правой стороны же мы видим последний цифровой вывод, который уже был рассмотрен, так же рядом с ним присутствует выход напряжения 3,3 Вольта, он нужен для питания датчиков с напряжением питания 3,3 Вольта, и для других возможных целей, где потребуется данное напряжение. Следующий вывод, который мы видим на рисунке 20, это Analog Reference. Данный вывод служит для подачи на него опорного напряжения, что бы в дальнейшем аналого-цифровой преобразователь микроконтроллера работал с ним, это необходимо для работы со слабыми сигналами, для повышения точности. Затем можно видеть, что идут вывода с названием Analog Input.

Это аналоговые входы, которые поступают на аналого-цифровой преобразователь. В данной микроконтроллере 10 битный преобразователь, это означает, что можно принимать 1024 различных значения. Всего в данном модуле 8 аналоговых входов, которые можно ещё использовать как цифровые.

Далее мы видим вывод для подачи на него 5 Вольт, или же если модуль уже запитан, например от USB или стабилизатора, тогда от туда можно взять 5 Вольт, например для питания датчиков. Так же рядом находится вывод Vin, это и есть вход для подачи питания, которое больше 5 вольт, но не больше предельно допустимого напряжения, указанного в характеристиках на данный модуль. Рядом, сверху модуля можно увидеть разъём ICSP. Он нужен там для установки загрузчика на новый микроконтроллер, что позволит прошивать данный микроконтроллер, по интерфейсу USB. А так же он даёт возможность использовать наш модуль в качестве USB программатора.

Необходимо обратить внимание на то, что некоторые вывода, выбранного модуля Arduino nano, имеют несколько функций, как например указывалось выше что некоторые вывода могут генерировать широтно-импульсную модуляцию, а в частности это цифровые вывода под следующими номерами: 3,5,6,9,10 и 11. А так же, на цифровых выводах 1 и 0 присутствует шина последовательного интерфейса, что описывалось чуть выше. На цифровых выводах 2 и 3 присутствует аппаратное прерывание. То есть на данных цифровых выводах, можно вызвать прерывание программы на переднем или заднем фронте импульса, а так же при изменении значения.

На аналоговых выводах модуля так же имеется дополнительная функция, а конкретно на аналоговых выводах 4 и 5. Посредством данных выводов, осуществляется связь по I2C интерфейсу. А конкретно вывод 4 является SDA а, вывод 5 SCL.

Для комфортного использования велокомпьютера велосипедистом, было решено добавить в устройство датчик температуры и влажности.

Данный датчик будет показывать актуальную температуру и влажность за бортом велосипеда. После изучения каталогов был выбран датчик температуры DHT11, внешний вид данного датчика изображён на рисунке 21. Так же основные характеристики датчика, взятые с сайта производителя, изображены на рисунке 22.

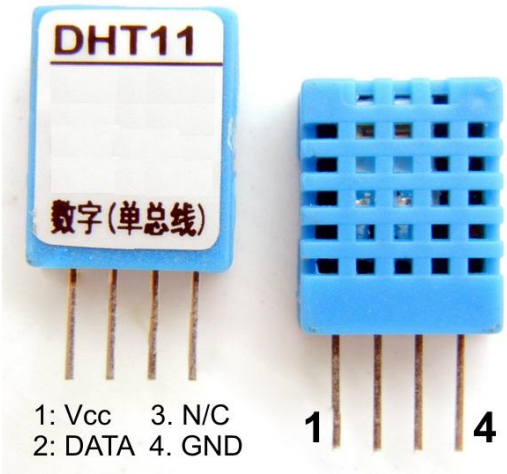

Рисунок 21 – Внешний вид датчика температуры DTH11.

Как можно видеть из рисунка 22, данный датчик измеряет только положительную температуру, что отлично подходит велокомпьютеру, так как велосипеды не рассчитаны на езду в холодные времена года и не так много людей этим занимаются, можно сказать что единицы. Так же погрешность датчика вполне допустимая для ознакомительных целей велосипедиста. Напряжения питания датчика, а так же его потребление тока полностью подходят. По своей сути, датчик является выносной, так как имеет свой отдельный корпус и может быть установлен в любой части велосипеда, что очень удобно.

Принцип работы датчика следующий [13]. Для измерения температуры, данный датчик использует термистор. Термистор – это резистор, в котором используется его сопротивление, но сопротивление сильно зависит от температуры. Если температуры будет увеличиваться, то

сопротивление будет стремиться к нулю, это называется отрицательный температурный коэффициент, который используется в данном термисторе для работы [14]. Так же ещё существуют терморезисторы с положительным температурным коэффициентом, то есть при увеличении температуры будет увеличиваться сопротивление до заданного максимального значения.

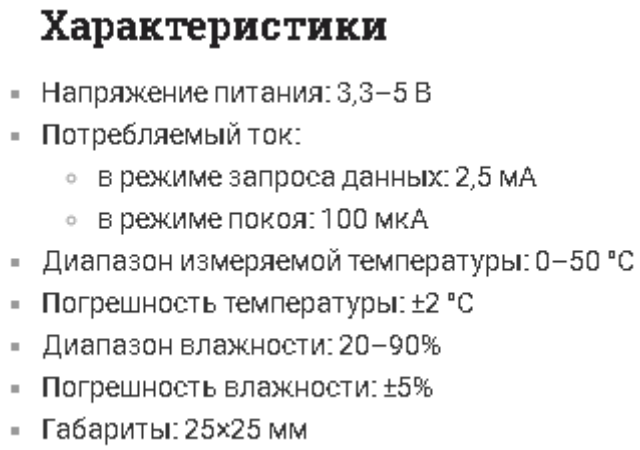

Рисунок 22 – Основные характеристики датчика.

Для измерения влажности используется конденсатор, в котором меняется ёмкость. По своей сути это является ёмкостным датчиком. На текстолите располагается токопроводящие медные обкладки. Данные обкладки заключены в своего рода герметичный чехол, поверх которого расположен влагопоглощающий слой. В моменты, когда вода попадает на этот слой, изменяется его диэлектрическая проницаемость, что в свою очередь приводит к изменению ёмкости нашего конденсатора.

Затем показания термистора и конденсатора обрабатываются специальной микросхемой, интегрированной, в корпус датчика и микроконтроллер уже получает в цифровом виде данные о температуре и влажности вокруг него.

Для подсчёта велокомпьютером скорости, а так же каденса, так же необходимы какие-либо датчики, которые способны фиксировать один оборот физического тела, то есть колеса и педалей. Для этих целей идеально

подойдёт Датчик Холла. Датчик Холла – это прибор, фиксирующий магнитное поле и его напряженность. Внешний вид датчика изображён на рисунке 23. Исходя из необходимой задачи, был выбран цифровой датчик

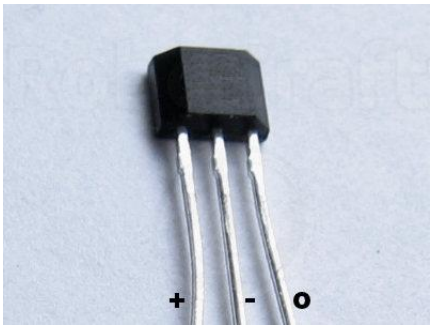

Рисунок 23 – Внешний вид датчика Холла.

Холла, а конкретно 44E 402. Данный датчик имеет следующие характеристики:

1. Модель - 44E 402;
2. напряжение питания – от 3,8 до 30 Вольт;
3. потребляемый ток в режиме покоя – 3 миллиАмпера;
4. потребляемый ток в режиме срабатывания – 8 миллиАмпер;
5. тип чувствительности к полю – униполярный;
6. индукция включения при 25℃– 60 Гаусс;
7. индукция выключения при 25℃– минус 60 Гаусс;
8. время нарастания сигнала - 1,5 микросекунды;
9. макс выходной ток - 20миллиАмпер;
10. температурный диапазон – от минус 40 до плюс 150 ℃;
11. корпус – TO-92.

Цифровые датчики Холла так называются, потому что их работа основана на принципе Холла, работают данные датчики следующим образом [15]. Если в магнитное поле, созданное близко находящемся магнитом, поместить пластину с протекающим через неё током, то электроны в данной

пластине станут отклоняться в направлении, перпендикулярном направлению тока в пластине. Так же следует отметить, что полярность магнитного поля важна и влияет на то, в какую именно сторону будут отклоняться электроны. Благодаря различной плотности электронов на сторонах пластины будет создаваться разность потенциалов, которую можно измерить, собственно этим датчики Холла и занимаются в итоге.

В данном случае, цифровой датчик Холла, после всего выше перечисленного выдаст нам лишь факт отсутствия или наличия магнитного поля, что идеально подходит для регистрации одного оборота вращающегося тела, с большой скоростью. Что, например не могут обеспечить герконы, так же для герконов необходимо строить систему защиты дребезга контактов, что усложняет схему и использование таковых датчиков. Но так, же стоит упомянуть, что существуют и аналоговые датчики Холла, которые в свою очередь после всего вышеперечисленного преобразуют индукцию магнитного поля в напряжение, знак и величина которого зависят от полярности и силы поля.

Вернёмся к цифровому датчику Холла, обратим внимание на рисунок 24, когда значение индукции выше порога срабатывания, датчик выдаёт логическую единицу, а когда ниже порога срабатывания, то логический ноль. Так же наличие зоны нечувствительности, называется гистерезисом. Она необходима для исключения ложных срабатываний датчика на помехи.

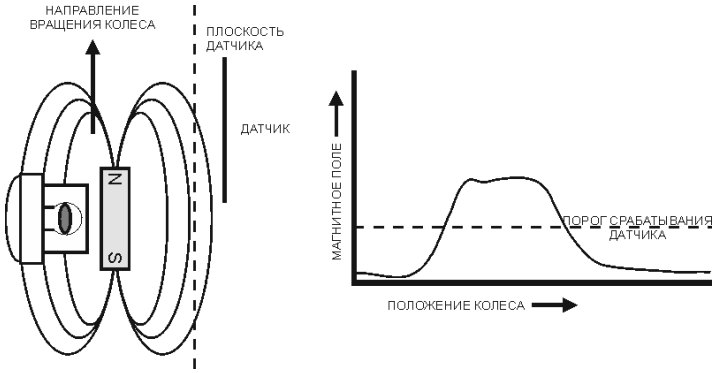

Рисунок – 24 Гистерезис датчика Холла.

Для того что визуально взаимодействовать с велокомпьютером, а так же управлять им посредством нажатия пальцем на плоскости, нам необходим дисплей с сенсором. Была выбрана данная модель Nextion NX4827T043_011R дисплея с сенсором в сборе [16]. Основные характеристики данного дисплея указаны на рисунке 25.

	NX4827T043_011R
Размер	4.3"
Разрешение	480*272
Touch Panel	RTP
Количество цветов	65536
Flash(MB)	16
RAM(Byte)	2048

Рисунок 25 – Основные характеристики дисплея.

Электрические характеристики дисплея следующие:

1. Номинальное напряжение питания – 5 Вольт;
2. пределы питающего напряжения от 4,75 до 7 Вольт
3. максимальное потребление тока дисплеем 0,25 Ампера;
4. потребление тока в режиме сна 0,015 Ампера;
5. рекомендуемый блок питания для нормальной работы 5 Вольт и 0,5 Ампер;
6. рабочее напряжение (логический уровень) – 5 Вольт;
7. тип подсвети дисплея – LED.

Внешний вид дисплея изображён на рисунке 26, а также внешний вид его тыльной стороны с указанием всех основных узлов на рисунке 27.

Данный дисплей называется Nextion ,а конкретно модель NX4827T043_011R, выпускается компанией ITEAD. Так как данный дисплей

является отдельным модулем и может работать самостоятельно, рассмотрим его подробнее [17].

Рисунок 26 – Внешний вид дисплея.

Рисунок 27 – Основные узлы дисплея.

Дисплей Nextion состоит из:

1. Микроконтроллера GD32F103R8T6;

2. flash память Winbond W25Q256FG (16 МБ, 100 000 циклов перезаписи);
3. разъём для SD карт памяти;
4. RAM память Winbond W9864G6KH-6 (8 МБ);
5. PLD Altera MAX II EPM570T144C5N;
6. драйвер резистивного дисплея XPT2045;
7. разъём UART инерфейса связи и питания.

Данный дисплей состоит, из самой матрицы и сенсора на ней, а так же микроконтроллера, который содержит в себе основную прошивку, и обрабатывает всю логическую информацию. Так же содержит flash память, для хранения изображений и других графических составляющих дисплея, данная память является энергонезависимой.

Плюс в данном устройстве имеется разъём для SD карты и он используется исключительно для перепрошивки дисплея если нет доступа к компьютеру или просто если так удобнее. Потом следует RAM память, её так же называют оперативной памятью. Данная память является энергозависимой и используется для сохранения в себе частей кода микроконтроллера или промежуточных значений, используется в данном дисплее для повышения производительности.

Дальше согласно рисунку 26, идёт микросхема PLD ,а так же она ещё может называться программируемая логическая интегральная микросхема. Данная микросхема так же используется для повышения производительности [18].

В этом устройстве так же присутствует микросхема драйвера резистивного сенсора, которая обрабатывает нажатия на сенсор, после чего отправляет данные в микроконтроллер дисплея.

А так же есть разъём UART интерфейса связи, через него может осуществляться программирование, а так же осуществляется связь всего дисплея с другими устройствами, поддерживающими этот интерфейс,

например микроконтроллерами. Ну, а так же подводится всё питание на дисплей.

Данный модуль дисплея был выбран по причине высокой производительности, а так же благодаря тому, что это отдельное устройство, оно не будет нагружать основной микроконтроллер, благодаря чему не требуется брать мощный микроконтроллер. А так же резистивный сенсор идеально подходит для велокомпьютера, так как не реагирует на воду и прикосновение других предметов, так как ему нужно именно нажатие, а не касание.

Велокомпьютеру так же необходима возможность визуально контролировать срабатывание датчика Холла, это необходимо при установке датчика на колесо или педали, а так же если возникли сомнения в его исправности. Имеется два способа реализовать данную задачу, программно и аппаратно. Схема визуального контроля датчиков Холла при помощи аппаратного решения изображена на рисунке 28.

Для аппаратного варианта решено было взять за основу схему компаратора [19] на операционном усилителе без отрицательной обратной связи. Обратим внимание на рисунок 28, микросхема DA это операционный усилитель, R2 и R3 представляют из себя делитель напряжения и подают это напряжение на инвертирующие входа операционного усилителя. Данное напряжение является опорным. А на неинвертирующие входа подаётся сигнал с выхода датчиков Холла. Когда напряжение на неинвертирующем входе станет больше, чем напряжение на инвертирующем, то на выходе операционного усилителя будет напряжение питания, в нашем случае это логическая единица 5 Вольт. Соответственно принимая это во внимание, можно на выходы операционных усилителей подключить светодиоды, в данном случае HL1 и HL2, которые дают возможность визуального контроля работоспособности датчиков Холла. Резисторы R1 и R4 являются токоограничивающими для светодиодов. То есть, при появлении логической

единицы на выходе операционного усилителя, будут загораться светодиоды давая понять человеку о том, что сработал датчик Холла.

Плюсы данного варианта в том, что вообще не нужно тратить вычислительной мощности микроконтроллера и память в программе того же микроконтроллера на выполнение данной операции.

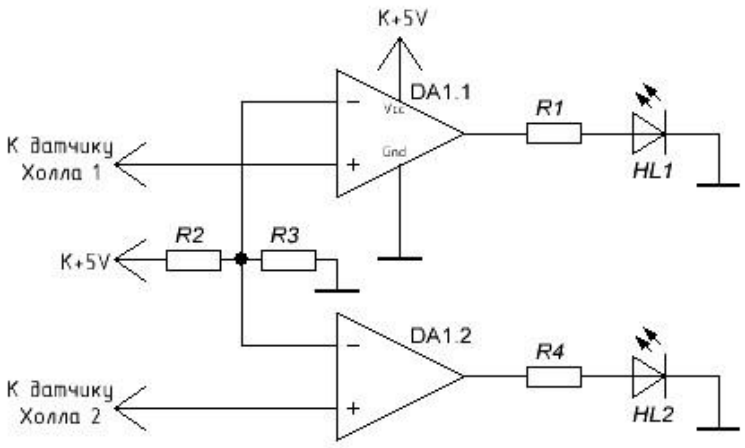

Рисунок 28 - Схема визуального контроля датчиков Холла.

Минусы же данного способа следующие, это усложнение схемы, а так же увеличение радиокомпонентов, что влечёт за собой усложнение печатной платы и снижение надёжности устройства.

Возможно, решить данную задачу так же и программным способом, при котором упрощается схема. Пример схемы изображён на рисунке 29. Как можно видеть на схеме, при таком решении задачи, требуется всего два светодиода для визуального контроля и соответственно два токоограничивающих резистора для светодиодов. Всю остальную часть будет выполнять часть программы в микроконтроллере.

Плюсы данного решения задачи это в разы упрощённая схема, упрощённая печатная плата, более высокая надёжность устройства, меньшее потребление тока.

Минусы данного варианта, затрачивание вычислительных ресурсов и ресурсов памяти микроконтроллера.

После рассмотрения двух вариантов, было решено выбрать программный способ решения данной задачи из-за её явных плюсов и при условии того что остаётся часть вычислительных ресурсов и ресурсов памяти микроконтроллера.

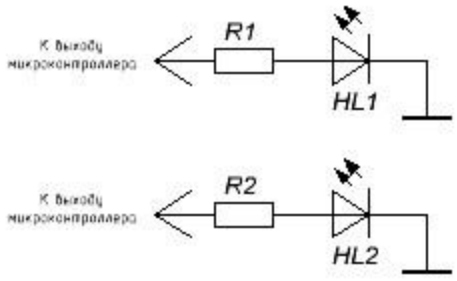

Рисунок 29 – Пример схемы при программном решении задачи.

Далее следует выбрать светодиоды и резисторы к ним. Так как всего по задумке требуется не два светодиода, а четыре, это два на настройку датчиков Холла и ещё два для системных сообщений велокомпьютера. А именно сообщений о приближении к полной разрядке велокомпьютера, о полной разрядке велокомпьютера, о вхождении в режим настроек и о выполнении алгоритма выключения. Рассмотрим немного подробнее именно системные сообщения.

Для предупреждения о скорой полной разрядке был выбран способ поочерёдного включения и выключения оранжевого светодиода с частотой в 1 секунду. При предупреждении о полной разрядке, поочерёдное включение и выключение сначала оранжевого, затем красного светодиода с частотой меньше секунды, после 10 циклов выполнение данного алгоритма полностью выключать велокомпьютер.

Так же при входе в режим настроек велокомпьютера будет загораться включаться светодиод и выключаться при выходе из данного режима. И при выполнении алгоритма выключения велокомпьютера будет включаться красный светодиод до самого выключения всего устройства.

Для данной задачи лучше всего подойдёт один RGB светодиод, и плюс к нему второй единичный светодиод. Второй светодиод для того, что бы имелась возможность непрерывного визуального контроля за двумя датчиками Холла, так как в RGB светодиоде всего три кристалла.

RGB светодиод называется так от первых букв английских названий цветов Red Green Blue, что означает красный, зелёный и синий. Данный светодиод содержит три основных цвета, при помощи которых можно получать все основные цвета и их оттенки. Но для этого помимо разных вариантов включения одновременно двух или трёх цветов, так же нужно регулировать их яркость, для этого так же необходимо что бы вывода микроконтроллера обладали возможностью широтно-импульсной модуляции. Окончательная схема визуального контроля датчиков Холла изображена на рисунке 30.

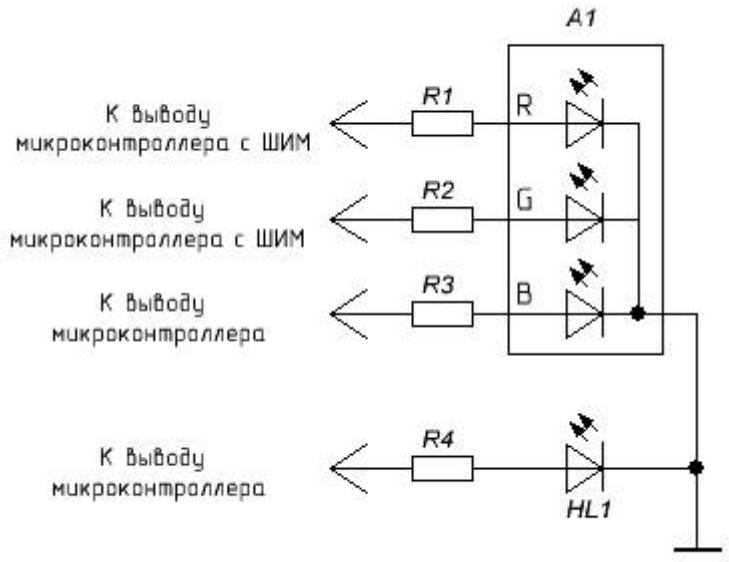

Рисунок 30 – Схема визуального контроля датчиков Холла.

В качестве RGB светодиода была выбрана модель FYL-5019RGBW1C обладающая следующими характеристиками, изображёнными на рисунке 31. В качестве синего светодиода была выбрана модель FYL-5002UBC1F обладающая следующими характеристиками, изображёнными на рисунке 32.

Произведём расчёт токоограничивающих резисторов для светодиодов, так как напряжение питания 5 вольт [20], а напряжение для светодиодов, согласно приведённым ниже характеристикам меньше напряжения питания, так же для ограничения тока светодиода, это нужно для исключения быстрой деградации кристалла светодиода.

Посчитать сопротивление токоограничивающего резистора R_{HL} можно по формуле 1.

$$R_{\text{то}} = \frac{(V_{\text{п}} - V_{\text{пн}})}{I_{\text{св}}}, \quad (1)$$

где $R_{\text{то}}$ – токоограничивающий резистор, Ом;

$V_{\text{п}}$ – напряжение питания, Вольт;

$V_{\text{пн}}$ – падение напряжения на светодиоде, Вольт;

$I_{\text{нсв}}$ – номинальный ток потребления светодиода, Ампер.

Далее необходимо посчитать мощность рассеивания для токоограничивающего резистора. Произвести расчёт можно по формуле 2.

$$P_{\text{рас}} = I_{\text{рсв}}^2 \cdot R_{\text{то}}, \quad (2)$$

где $P_{\text{рас}}$ – мощность, рассеиваемая на резисторе, Ватт;

$I_{\text{рсв}}$ – реальный ток потребления светодиода, Ампер;

$R_{\text{то}}$ – токоограничивающий резистор, Ом.

Рассчитаем токоограничивающий резистор для красного кристалла R_к RGB светодиода по формуле 1.

R_к= ((5-1,9))/0,013=238,4≈238 Ом

Для подсчёта токоограничивающего резистора, вместо 20 номинальных миллиампер было принято решение во всех подсчётах принимать значение равное 13 миллиампер, это необходимо для того что свести к минимуму

возможности деградации кристалла светодиода, а так же 13 миллиампер вполне достаточно для нормального свечения светодиода что уменьшает энергопотребление системы.

Характеристики

Цвет свечения	Красный/Зелёный/Синий Red/Green/Blue
Тип линзы	Белая прозрачная
Длина волны, нм	628/520/470
Интенсивность, мКд	340/370/260
Угол обзора, град.	50
Размеры (ВхШхГ), мм	\Nx\Nx\N
Размер точки, мм	\N
Расстояние между точками (X/Y), мм	\N/\N
Падение напряжения на кристале светодиода (Red), В	1,9
Падение напряжения на кристале светодиода (Green), В	2,3
Падение напряжения на кристале светодиода (Blue), В	2,1
Постоянный прямой ток, мА	20

Описание

Полноцветные RGB светодиоды общего применения.
Круглые ø5мм
RoHs

Рисунок 31 – Характеристики RGB светодиода.

Параметр	Показатель
Сила светового потока, Лм	*45-50*
Потребляемая мощность, мВт	*0,5*
Диапазон рабочих температур, в градусах С	*От -40 до +80*
Номинальный ток, мА	*20*
Рабочее напряжение, В	*2,1-2,3*
Угол освещения	*120 градусов*

Рисунок 32 – Характеристики синего светодиода.

Для подсчёта токоограничивающего резистора, вместо 20 номинальных миллиампер было принято решение во всех подсчётах принимать значение равное 13 миллиампер, это необходимо для того что свести к минимуму возможности деградации кристалла светодиода, а так же 13 миллиампер вполне достаточно для нормального свечения светодиода что уменьшает энергопотребление системы.

Согласно расчёту выбираем резистор из стандартного ряда E12 равному 220 Ом.

Рассчитаем мощность, рассеиваемую на резисторе $R_к$ по формуле 2.

$P_к = 0{,}013 \cdot 220 = 0{,}037$ Ватт

Согласно расчёту мощности для схемы подойдёт резистор стандартной мощности 0,25 Ватта. Из этого следует, согласно полученным данным, выбрана модель резистора CF-25, характеристики которого указаны на рисунке 33.

Технические параметры

Тип	c1-4
Номин.сопротивление	220
Единица измерения	ом
Точность,%	5
Номин.мощность,Вт	0.125/0.25
Макс.рабочее напряжение,В	250
Рабочая температура,С	-55...125
Монтаж	в отв.
Длина корпуса L,мм	6.3
Ширина (диаметр) корпуса W(D),мм	2.3

Рисунок 33 – Параметры выбранного резистора CF-25.

Рассчитаем токоограничивающий резистор для зелёного кристалла $R_з$ RGB светодиода по формуле 1.

$$R_з = \frac{(5-2{,}3)}{0{,}013} = 207{,}6 \approx 208 \text{ Ом}$$

Согласно расчёту выбираем резистор из стандартного ряда E12 равному 220 Ом. Так как сопротивление $R_з$ получилось таким же, как и $R_к$, то мощность, а так же и остальное получается идентично, как и у $R_к$, соответственно выбирается та же марка резистора CF-25 с теми же характеристиками как на рисунке 33.

Рассчитаем токоограничивающий резистор для синего кристалла $R_с$ RGB светодиода по формуле 1.

$$R_с = \frac{(5-2{,}1)}{0{,}013} = 223 \text{ Ом}$$

Согласно расчёту выбираем резистор из стандартного ряда E12 равному 220 Ом. Так как сопротивление $R_с$ получилось таким же, как и $R_з$, то мощность, а так же и остальное получается идентично, как и у $R_з$, соответственно выбирается та же марка резистора CF-25 с теми же характеристиками как на рисунке 33.

Дальнейший расчёт для отдельного синего светодиода получится практически идентичным, так как параметры падения напряжения и потребляемого тока практически такие же как выше, делать данный расчёт не имеет смысла и выбирается резистор марки CF-25 с такими же характеристиками как на рисунке 33.

Как видно из расчётов, значения сопротивлений для резисторов разных кристаллов светодиода практически одинаковы и имеют небольшую разницу, а так же они имеют достаточный запас по потребляемому току. То этой разницей можно пренебречь и выбрать резистор одного номинала и марки, для упрощения схемы, что даёт незначительную простоту при сборке, а так же незначительное удешевление, чем, если бы покупалось четыре разных резистора с почти одинаковым сопротивлением.

Согласно схеме электрической принципиальной, изображённой на рисунке 9, нам так же необходима тактовая кнопка для включения велокомпьютера и его последующего выключения. Для того что бы выбрать подходящую по параметрам тактовую кнопку необходимо посчитать

потребляемый ток всей схемы $I_{\text{с}}$. Для подсчёта потребляемого тока всей схемы согласно всем выше указанным характеристикам к компонентам и модулям можно по формуле 3.

$$I_{\text{с}} = I_{\text{а}} + I_{\text{дт}} + I_{\text{дх}} \cdot 2 + I_{\text{д}} + (I_{\text{рсв}} \cdot 4), \quad (3)$$

где $I_{\text{с}}$ – потребляемый ток всей схемы, миллиампер;

$I_{\text{а}}$ – потребляемый ток модуля ардуино, миллиампер;

$I_{\text{дт}}$ – потребляемый ток датчика температуры, миллиампер;

$I_{\text{дх}}$ – потребляемый ток датчиков Холла, миллиампер;

$I_{\text{д}}$ – потребляемый ток дисплея, миллиампер.

Умножение $I_{\text{дх}}$ на два нужно из-за того, что в схеме два датчика Холла, а так же соответственно умножение $I_{\text{рсв}}$ на четыре нужно, потому что в схеме два светодиода, один с одним кристаллом, а второй RGB с 3 кристаллами.

$$I_{\text{с}} = 60 + 2{,}5 + 8 \cdot 2 + 250 + 13 \cdot 4 = 380{,}5 \approx 380 \text{ миллиампер}$$

После произведённого расчёта мы получили то, что расчётное потребление всей схемы равняется $I_{\text{с}} = 380$ миллиампер, но так же следует учесть, что в схеме присутствует повышающий преобразователь напряжения, у которого коэффициент полезного действия не может равняться ста процентам. Поэтому рассчитаем ток потребления всей схемы $I_{\text{сп}}$ с учётом потерь на преобразование напряжения по формуле 4.

$$I_{\text{сп}} = \frac{(I_{\text{с}} \cdot 100)}{\eta_{\text{п}}}, \quad (4)$$

где $I_{\text{сп}}$ – потребление тока всей схемы с учётом потерь на преобразование , миллиампер;

$\eta_{\text{п}}$ – коэффициент полезного действия преобразователя напряжения, %.

$$I_{\text{сп}} = \frac{(380 \cdot 100)}{93} = 408{,}6 \approx 409 \text{ миллиампер}$$

После произведённого расчёта мы получили что $I_{\text{сп}} = 409$ миллиампер, после того как известен ток с учётом потерь, можно выбрать тактовую кнопку с соответствующими электрическими параметрами.

Согласно расчётам был выбран микропереключатель МП5 так как он соответствует необходимым условиям, а это наличие нормально-замкнутого и нормально-разомкнутого контакта, а так же максимально допустимый ток и напряжение для контактов. Электрическая схема коммутации микропереключателя МР5 изображена на рисунке 34 Полные характеристики данного микропереключателя изображены на рисунках 35 и 36.

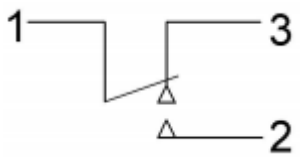

Рисунок 34 - Электрическая схема коммутации.

Основные технические и эксплуатационные характеристики микропереключателей МП5:	
Сопротивление контакта, не более, Ом	0.05
Электрическая прочность изоляции, В	
- для приемки «1»	750
- для приемки «5»	1100
Сопротивление изоляции, МОм, не менее	1000
Усилие прямого срабатывания, Н, не более:	
- для приемки «1»	0,98... 2,94
- для приемки «5»	0,98... 2,25
Рабочая температура окружающей среды, °С	от -60 до +125
Повышенная относительная влажность при 35°С, %	98
Гарантийная наработка, в течении гарантийного срока, ч	
- для приемки «1»	10000
- для приемки «5»	5000
Гарантийный срок с даты изготовления, лет:	
- для приемки «1»	15
- для приемки «5»	20
Масса, г	3,5

Рисунок 35 – Характеристики микропереключателя МП-5.

Для включения и поддержания во включенном состоянии устройства необходим транзисторный ключ, как и видно из рисунка 9.

Электрические режимы коммутации микропереключателей МП5:								
Типономинал микропереключателя	Род тока	Вид нагрузки	Напряжение, В		Ток, А		Максимальная коммутируемая мощность, Вт (ВхА)	Количество циклов переключений в НКУ
			не менее	не более	не менее	не более		
МП5 МП5В	постоянный	активная	0,2	30	$2x10^{-4}$	4	70	5000-200000
	постоянный	индуктивная	0,2	30	$2x10^{-4}$	2	70	5000-200000
	переменный	активная	0,2	250	$2x10^{-4}$	3	300	5000-200000
	переменный	индуктивная	0,2	250	$2x10^{-4}$	2	300	5000-200000

Рисунок 36 - Характеристики микропереключателя МП-5.

Так как необходимо коммутировать плюс питания, то необходим прямой транзистор PNP структуры, но мы не сможем напрямую управлять им микроконтроллером из-за особенностей управления PNP транзистора. В связи с этим, для управления прямым PNP транзистором, решено использовать обратный транзистор NPN структуры, которым уже можно легко управлять при помощи микроконтроллера. Схема транзисторного ключа с управлением изображена на рисунке 37.

Транзисторный ключ работает следующим образом. На резистор R1 подаётся логическая единица с цифрового выхода микроконтроллера, которая равна примерно 5 Вольтам. При этом на базе транзистора VT1 появляется управляющий ток, транзистор открывается и за счёт этого напряжение начинает протекать через R2 так же создавая управляющей ток на базе транзистора VT2, после чего данный транзистор открывается, пропуская через себя плюс напряжения с аккумулятора на повышающий преобразователь.

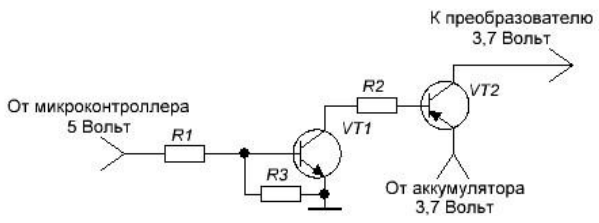

Рисунок 37 – Схема транзисторного ключа с управлением.

Для начала нам необходимо выбрать силовой транзистор VT2, а затем после расчёта уже с известными параметрами, можно будет выбрать транзистор VT1.

Как нам уже известно, из вышеуказанных расчётов потребления тока схемы с учётом потерь ровняется $I_{сп} = 409$ миллиампер, соответственно выбираем транзистор с запасом по току в несколько раз из-за особенностей конструкции велокомпьютера. Так как велокомпьютер должен быть не большого размера и компактный, радиатор для охлаждения транзистора, даже маленький не поместится, поэтому выбираем транзистор с большим запасом по току, тогда необходимость обеспечения отпадает, так как транзистор вообще не будет греться.

После изучения справочников по транзисторам, был выбран транзистор марки КТ-814Г, как самый подходящий по своим характеристикам в качестве силового ключа схемы. Характеристики транзистора КТ-814Г изображены на рисунке 38.

Структура **p-n-p**
- Максимально допустимое (импульсное) напряжение коллектор-база **100 В**
- Максимально допустимое (импульсное) напряжение коллектор-эмиттер **80 В**
- Максимально допустимый постоянный(импульсный) ток коллектора **1500(3000) мА**
- Максимально допустимая постоянная рассеиваемая мощность коллектора без теплоотвода (с теплоотводом) **1(10) Вт**
- Статический коэффициент передачи тока биполярного транзистора в схеме с общим эмиттером **30-275**
- Обратный ток коллектора **<=50 мкА**
- Граничная частота коэффициента передачи тока в схеме с общим эмиттером **=>3 МГц**
- Коэффициент шума биполярного транзистора **<0.6 дБ**

Рисунок 38 – Характеристики транзистора КТ-814Г.

Рассчитаем величину резистора R2 ,изображённого на рисунке 36 для работы транзистора VT2 в ключевом режиме.

Сначала необходимо рассчитать ток базы I_6, сделать это можно с помощью формулы 5.

$$I_б = \frac{I_к}{h_{21e}}, \quad (5)$$

где $I_б$ – ток базы транзистора, Ампер;

$I_к$ – максимальный ток коллектора, Ампер;

h_{21e} – усиление по току.

$$I_б = \frac{1,5}{30} = 0,05 \text{ ампер} \approx 50 \text{ миллиампер}$$

Так как согласно характеристикам, указанным на рисунке 37 параметр h_{21e} с большим разбросом, было взято минимальное значение.

Далее необходимо рассчитать напряжение на базе транзистора $U_б$. Напряжение на базе транзистора рассчитывается согласно формуле 6.

$$U_б = U_п - U_{пбэ}, \quad (6)$$

где $U_б$ – напряжение базы транзистора, Вольт;

$U_п$ – напряжение питания, Вольт;

$U_{пбэ}$ –падение напряжения на переходе база-эмиттер, Вольт.

$$U_б = 5 - 0,6 = 4,4 \text{ Вольта}$$

Так как уже известны нужные параметры, а это ток базы $I_б$ и напряжение базы $U_б$ согласно закону ома можно рассчитать резистор базы $R_б$ по формуле 7.

$$R_б = \frac{U_б}{I_б}, \quad (7)$$

где $R_б$ – сопротивление резистора базы, Ом;

$$I_б = \frac{4,4}{0,05} = 88 \text{ Ом}$$

Согласно расчёту выбираем резистор из стандартного ряда E24 равному 82 Ома.

Далее необходимо посчитать мощность рассеивания на базовом резисторе. Произвести расчёт можно по формуле 8.

$$P_{рас} = I_б^2 \cdot R_б, \quad (8)$$

где $P_{рас}$ – мощность, рассеиваемая на резисторе, Ватт;

$I_б$ – реальный ток потребления светодиода, Ампер;

$R_б$ – токоограничивающий резистор, Ом.

$P_{рас} = 0{,}05^2 \cdot 82 = 0{,}205$ Ватта

Согласно расчёту мощности для схемы подойдёт резистор стандартной мощности 0,25 Ватта. Из этого следует, согласно полученным данным, выбрана модель резистора CF-25, характеристики которого указаны на рисунке 39.

Технические параметры

Тип	с1-4
Номин.сопротивление	82
Единица измерения	ом
Точность,%	5
Номин.мощность,Вт	0.125/0.25
Макс.рабочее напряжение,В	250
Рабочая температура,С	-55...125
Монтаж	в отв.
Длина корпуса L,мм	6.3
Ширина (диаметр) корпуса W(D),мм	2.3

Рисунок 39 – Характеристики выбранного резистора.

После того как известно что ток базы силового транзистора составляет $I_б = 50$ миллиампер необходимо выбрать управляющий транзистор VT1. Из-за значительного маленького тока базы, может подойти почти любой NPN транзистор. После изучения справочника транзисторов был выбран транзистор КТ-315Г со следующими характеристиками, изображёнными на рисунке 40.

Рассчитаем величину резистора R2 ,изображённого на рисунке 36 для работы транзистора VT2 в ключевом режиме.

Сначала необходимо рассчитать ток базы $I_б$, сделать это можно с помощью формулы 5.

$$I_б = \frac{0,1}{200} = 0,0005 \text{ ампер} \approx 0,5 \text{ миллиампера}$$

Так как согласно характеристикам, указанным на рисунке 37 параметр h_{21e} с большим разбросом, было взято среднее значение.

Далее необходимо рассчитать напряжение на базе транзистора $U_б$. Напряжение на базе транзистора рассчитывается согласно формуле 6.

$$U_б = 2,6 - 0,4 = 2,2 \text{ Вольта}$$

Так как уже известны нужные параметры, а это ток базы $I_б$ и напряжение базы $U_б$ согласно закону ома можно рассчитать резистор базы $R_б$ по формуле 7.

$$I_б = \frac{2,2}{0,0005} = 4400 \text{ Ом}$$

Согласно расчёту выбираем резистор из стандартного ряда E24 равному 4,3 килоома.

Структура **n-p-n**
- Максимально допустимое (импульсное) напряжение коллектор-база **35 В**
- Максимально допустимое (импульсное) напряжение коллектор-эмиттер **35 В**
- Максимально допустимый постоянный(импульсный) ток коллектора **100 мА**
- Максимально допустимая постоянная рассеиваемая мощность коллектора без теплоотвода (с теплоотводом) **0.15 Вт**
- Статический коэффициент передачи тока биполярного транзистора в схеме с общим эмиттером **50-350**
- Обратный ток коллектора **<=0.5 мкА**
- Граничная частота коэффициента передачи тока в схеме с общим эмиттером **=>250 МГц**

Рисунок 40 – Характеристики транзистора КТ-315.

Далее необходимо посчитать мощность рассеивания на базовом резисторе. Произвести расчёт можно по формуле 8.

$$P_{рас} = 0,0005^2 \cdot 4300 = 0,0010 \text{ Ватта}$$

Согласно расчёту мощности для схемы подойдёт резистор стандартной мощности 0,25 Ватта. Из этого следует, согласно полученным данным, выбрана модель резистора CF-25, характеристики которого указаны на рисунке 41.

Так же для транзисторного ключа на транзисторе VT1 необходим резистор R3 изображённом на рисунке 36. Он нужен для того что бы полностью закрывать транзистор VT1 который в свою очередь полностью закроет VT2. Но из-за возможных наводок на плате, которых бывает много, транзистор VT1 может не полностью закрыться, поэтому если поставить между базой и эмиттером резистор, через его сопротивление все помехи будут подтянуты на общий контакт схемы, что исключит неполное закрытие. Так же благодаря резистору при подачи на базу управляющего напряжения, транзистор нормально откроется и не будет короткого замыкания. Однако его рассчитывать нет никакой необходимости, обычно значение этого резистора принимают равным от 5 до 10 килоом. Исходя из этого, был выбран резистор марки CF-25 со следующими характеристиками, изображёнными на рисунке 42.

Так же велокомпьютер измеряет напряжение аккумулятора через аналоговый вход с аналого-цифровым преобразователем, но если не разрывать эту цепь, то при попытке выключить велокомпьютер, контроллер будет питаться через эту цепь и не выключаться. Что бы решить данную проблему, было принято решение так же поставить транзисторный ключ, но не такой мощный и так как нам будет необходимо коммутировать напряжение положительной полярности, необходим транзистор PNP структуры, то нужен будет и второй транзистор NPN структуры, что бы им управлять. Схема транзисторного ключа с управлением для коммутации измеряемого напряжения изображена на рисунке 43.

Для начала нам необходимо выбрать транзистор VT2, а затем после расчёта уже с известными параметрами, можно будет выбрать транзистор VT1. Как как для измерение будет потребляться очень маленький ток, которым можно пренебречь и подойдёт практически любой транзистор нужной структуры. После изучения справочников по транзисторам, был выбран транзистор марки КТ-361Д, как самый подходящий по своим

характеристикам. Характеристики транзистора КТ-361Д изображены на рисунке 44.

Технические параметры

Тип	с1-4
Номин.сопротивление	4.3
Единица измерения	ком
Точность,%	5
Номин.мощность,Вт	0.125/0.25
Макс.рабочее напряжение,В	250
Рабочая температура,С	-55...125
Монтаж	в отв.
Длина корпуса L,мм	6.3
Ширина (диаметр) корпуса W(D),мм	2.3

Рисунок 41 – Характеристики выбранного резистора.

Технические параметры

Тип	с1-4
Номин.сопротивление	10
Единица измерения	ком
Точность,%	5
Номин.мощность,Вт	0.125/0.25
Макс.рабочее напряжение,В	250
Рабочая температура,С	-55...125
Монтаж	в отв.
Длина корпуса L,мм	6.3
Ширина (диаметр) корпуса W(D),мм	2.3

Рисунок 42 – Характеристики выбранного резистора.

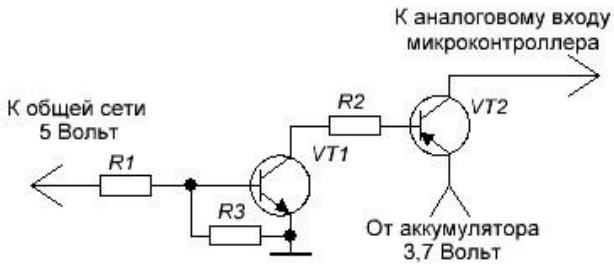

Рисунок 43 – Схема транзисторного ключа с управлением.

Рассчитаем величину резистора R2 ,изображённого на рисунке 43 для работы транзистора VT2 в ключевом режиме.

Сначала необходимо рассчитать ток базы $I_б$, сделать это можно с помощью формулы 5.

Структура **p-n-p**
- Максимально допустимое (импульсное) напряжение коллектор-база **40 В**
- Максимально допустимое (импульсное) напряжение коллектор-эмиттер **40 В**
- Максимально допустимый постоянный(импульсный) ток коллектора **50 мА**
- Максимально допустимая постоянная рассеиваемая мощность коллектора без теплоотвода (с теплоотводом) **0.15 Вт**
- Статический коэффициент передачи тока биполярного транзистора в схеме с общим эмиттером **20-90**
- Обратный ток коллектора **<=1 мкА**
- Граничная частота коэффициента передачи тока в схеме с общим эмиттером **=>250 МГц**

Рисунок 44 – Характеристики транзистора КТ-361Д.

$$I_б = \frac{0{,}05}{55} = 0{,}0009 \text{ ампер} \approx 0{,}9 \text{ миллиампера}$$

Так как согласно характеристикам, указанным на рисунке 44 параметр h_{21e} с большим разбросом, было взято среднее значение.

Далее необходимо рассчитать напряжение на базе транзистора $U_б$. Напряжение на базе транзистора рассчитывается согласно формуле 6.

$$U_б = 5 - 0{,}6 = 4{,}4 \text{ Вольта}$$

Так как уже известны нужные параметры, а это ток базы $I_б$ и напряжение базы $U_б$ согласно закону ома можно рассчитать резистор базы $R_б$ по формуле 7.

$$I_б = \frac{4{,}4}{0{,}0009} = 4889 \text{ Ом } \approx 4{,}9 \text{ килоома}$$

Согласно расчёту выбираем резистор из стандартного ряда E24 равному 4,7 килоома.

Далее необходимо посчитать мощность рассеивания на базовом резисторе. Произвести расчёт можно по формуле 8.

$$P_{\text{рас}} = 0{,}0009^2 \cdot 4700 = 0{,}0038 \text{ Ватта}$$

Согласно расчёту мощности для схемы подойдёт резистор стандартной мощности 0,25 Ватта. Из этого следует, согласно полученным данным, выбрана модель резистора CF-25, характеристики которого указаны на рисунке 45.

После того как известно что ток базы транзистора составляет $I_6 = 0{,}9$ миллиампера подойдёт практически любой транзистор нужной структуры в качестве управляющего транзистора VT1. В качестве транзистора VT1 подойдёт уже выше рассчитанный и выбранный КТ-315Г, поэтому делать повторный расчёт не имеет смысла.

Технические параметры

Тип	с1-4
Номин.сопротивление	4.7
Единица измерения	ком
Точность,%	5
Номин.мощность,Вт	0.125/0.25
Макс.рабочее напряжение,В	250
Рабочая температура,С	-55...125
Монтаж	в отв.
Длина корпуса L,мм	6.3
Ширина (диаметр) корпуса W(D),мм	2.3

Рисунок 45 – Характеристики выбранного резистора.

Так же необходимо решить проблему дребезга контактов с микропереключателя для взаимодействия с микроконтроллером. Решено решить проблему на аппаратном уровне. Схема защиты от дребезга контактов изображена на рисунке 46.

Как видно на рисунке 46, для защиты от дребезга контактов используется транзистор в режиме транзисторного ключа. Это работает следующим образом. Когда на кнопку не происходит механического воздействия, то есть она не нажата. На базу транзистора подано управляющее

напряжение с аккумулятора и он открыт. Следовательно, цифровой вход подтянут на землю и напряжение на нём ровняется нулю. Но если нажать на кнопку, контакт микропереключателя разомкнется, и цифровой вход микроконтроллера подтянется напряжение питания, $U_{\text{пит}} = 5$ вольт, через резистор R2, соответственно микроконтроллер сработает от этого условия и начнёт выполнять заданный алгоритм. В качестве транзистора VT1 идеально подходит уже выше рассчитанный КТ-315Г, поэтому повторно вести расчёт не имеет смысла. Так же для резистора R2 так же не нужен расчёт, так как обычно в качестве подтягивающих и стягивающих резисторов используют резисторы с сопротивлением от 5 до 10 килоом. Из этого следует, что подойдёт резистор марки CF-25 с характеристиками, изображёнными на рисунке 42.

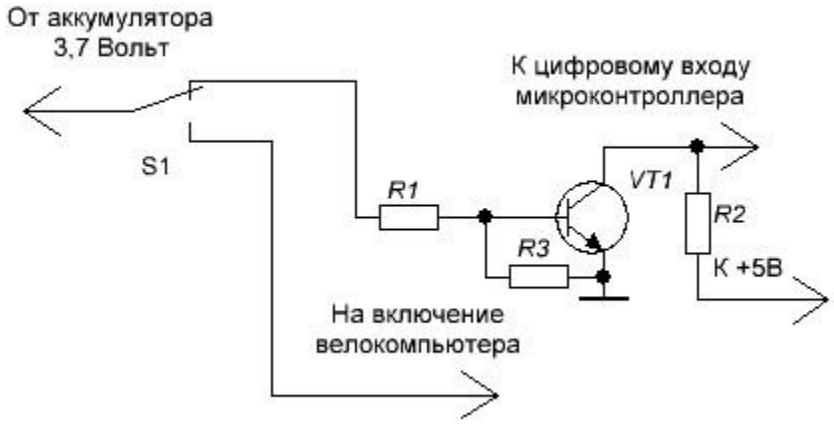

Рисунок 46 – Схема защиты от дребезга контактов.

2.3. Разработка печатной платы

После того как была разработана схема электрическая принципиальная, а так же был произведён выбор и расчёт элементов для устройства, далее необходимо разработать печатную плату. Печатная плата это важнейший элемент электронного устройства. Конструктивно печатная плата это деталь, состоит из диэлектрической подложки и токопроводящего слоя. В современное время в качестве диэлектрической подложки используется стеклотекстолит, но так же для более узконаправленных целей могут использоваться другие материалы. Например, анодированный алюминий, для установки на них мощных светодиодов или каптон для гибких клавиатур.

В качестве токопроводящего слоя используется медная фольга, в основном толщина данной фольги от 18 до 35 микрометров, но реже есть необходимость в использовании токопроводящего слоя толщиной 70, 105 и 140 микрометров. Как правило, такие более редкие значения толщины используются для устройств с большими токами, например для драйверов высоко токовых двигателей или мощных светодиодных прожекторов.

Печатные платы могут быть как односторонние, так и двух сторонние, а могу быть ещё многослойные. Для простых устройств используются односторонние печатные платы, они самые дешёвые в производстве и легки в проектировании и изготовлении, так как токопроводящий слой нанесён лишь с одной стороны.

Соответственно для двухсторонних плат, токопроводящий слой нанесён с двух сторон, что позволяет сделать печатную плату для более сложного устройства с меньшими размерами, чем, если это было бы изготовлено на односторонней, но такая плата стоит дороже в изготовлении и сложнее в проектировании в сравнении с односторонними.

Многослойные печатные платы позволяют собирать конструктивно сложнейшие устройства, но очень дорогие в изготовлении, а так же очень сложны в проектировании, так же при их проектировании необходимо

учитывать ряд тонкостей для стабильной работы. Ещё большой минус таких плат это не ремонтопригодность, при повреждении промежуточного слоя дорожек печатной платы ничего уже не сделать. Такие платы, как правило, используют для изготовления материнских плат, видеокарт персонального компьютера, что оправдано сложностью схем.

Изначально имеется лишь фольгированный стеклотекстолит, со сплошным слоем токопроводящего материала. В последующем на него, разными способами наносится рисунок печатной платы, и разными способами излишки меди удаляются со стеклотекстолита. Таким образом, формируются дорожки печатной платы, далее наносятся маски, защитные слои и после этого уже можно сказать, что это печатная плата.

Способы нанесения рисунка могут быть разными, это может быть лак либо фоторезист или другие подобные способы. Способы удаление излишек меди тоже могут быть разными, это механические, путём фрезерования или выжигания лазером, так и химический, путём вытравливания в химическом растворе.

Было принято решение разработать печатную плату так же как электрическую принципиальную в пакете Dip Trace [21], а именно в программе PCB Layout. После того как схема электрическая принципиальная из программы Schematic готова, схема изображена на рисунке 47. Далее необходимо зайти в меню и воспользоваться функцией, преобразовать в плату. После чего Dip Trace автоматически открывает программу Schematic с корпусами радиоэлементов и установленными связями между ними согласно нарисованной схемы электрической принципиальной. Рисуем необходимых размеров границу печатной платы, и расставляем внутри неё корпуса радиоэлементов. Так получается заготовка к трассированию печатной платы, границы печатной платы и корпуса со связями для велокомпьютера изображены на рисунке 48.

Далее необходимо запустить функцию преобразовать в плату, после чего будет уже печатная плата, так как все связи преобразовываются в

дорожки автоматически. На рисунке 49 изображена страссированная печатная плата.

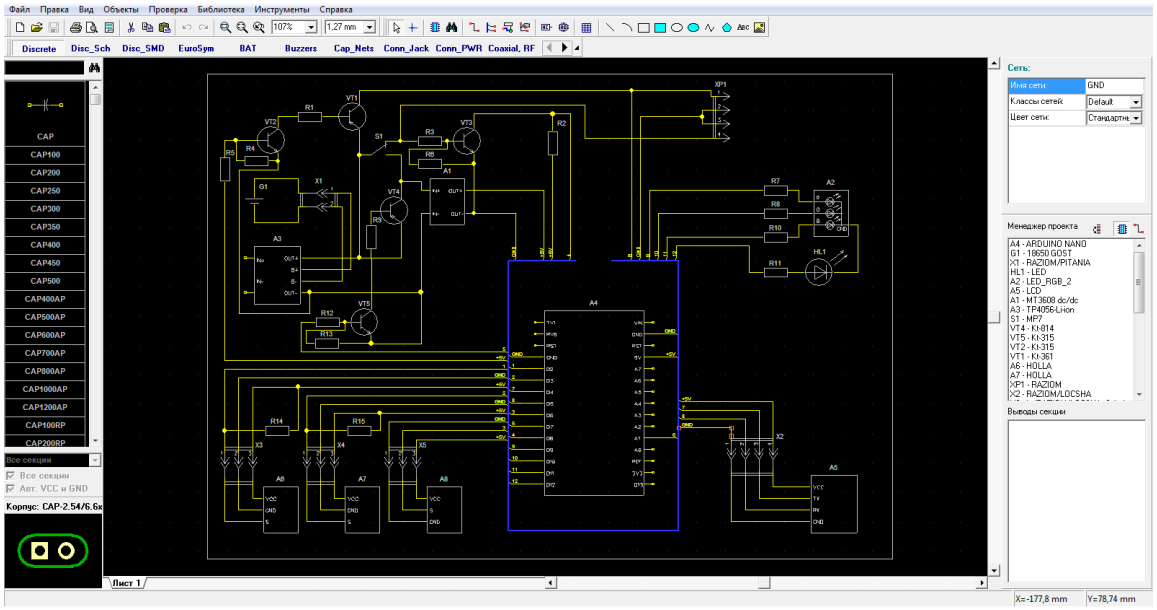

Рисунок 47 – Готовая схема для компиляции в печатную плату.

Затем необходимо определить ширину питающих дорожек для того, что бы исключить просадки по напряжению и току. Для этого воспользуемся онлайн калькулятором расчёт ширины дорожек печатной платы [22]. На рисунке 50 изображены начальные данный для расчёта. После чего получаем ответ для внешних слоёв печатной платы. Полученный ответ изображён на рисунке 51.

Согласно полученным результатам необходимо отредактировать печатную плату. Увеличить ширину дорожек, а так же правильно от позиционировать дорожки для получения правильного зазора между дорожками и контактами радиоэлементов. Так же свободные участки печатной платы необходимо металлизировать, для уменьшения затрачиваемого времени на изготовление печатных плат. Результат отредактированной печатной платы изображён на рисунке 52.

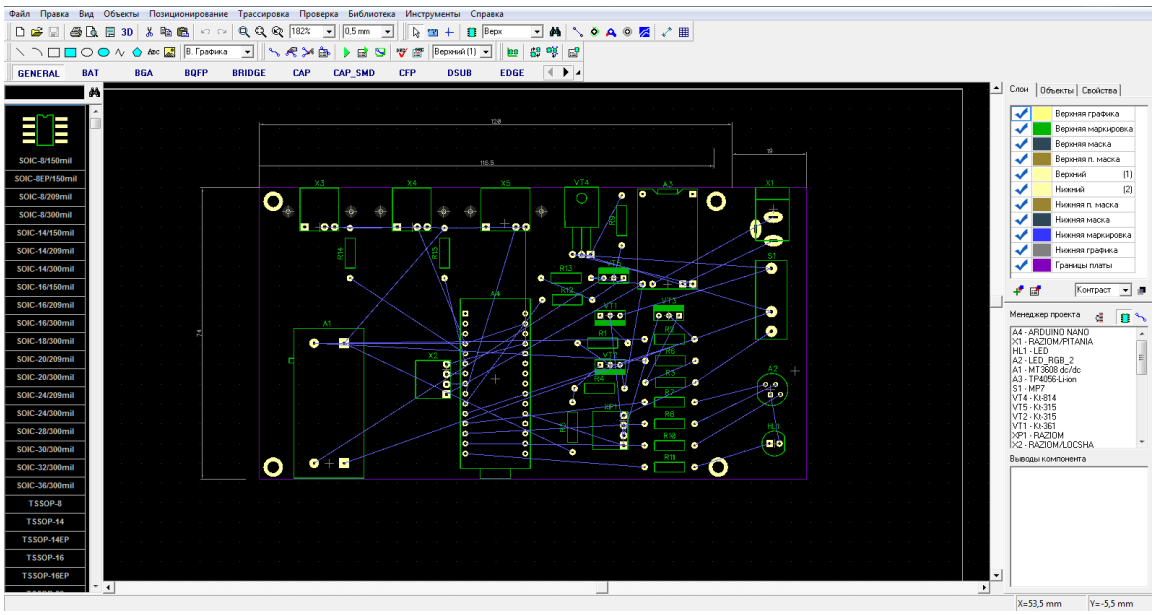

Рисунок 48 – Изображение границ печатной платы и связи радиоэлементов.

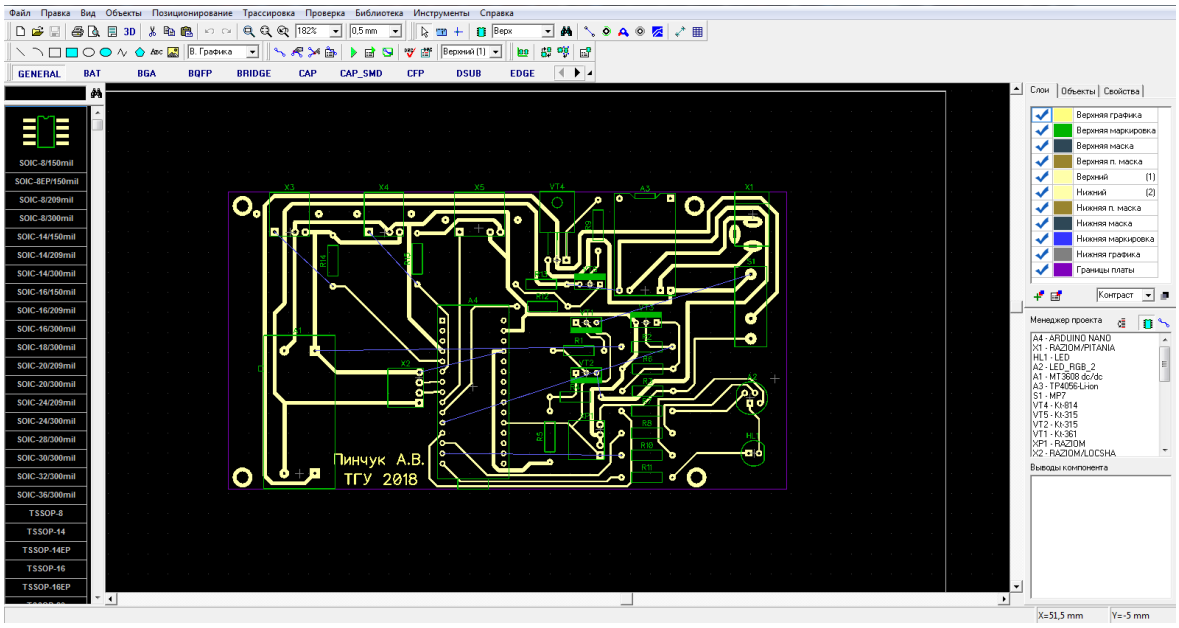

Рисунок 49 – Страссированная печатная плата.

В результате проделанной работы была получена печатная плата с размерами 139x74 миллиметров. Ширина сигнальных дорожек составляет 0,7 миллиметра, а ширина питающих дорожек в 1 миллиметр. Итоговая ширина

питающих дорожек выполнена шире с запасом, для более надёжной работы устройства. На плате не получилось избежать перемычек, и итоговое число перемычек составляет 8 штук. Можно было бы обойтись, без перемычек применив двухсторонний фольгированный стеклотекстолит и спроектировав двухстороннюю печатную плату с переходными отверстиями. Но такая плата выйдет сложнее и дороже и для данной компоновки радиоэлементов, двухсторонняя плата это не рентабельное использование двухсторонних печатных плат.

Чертёж печатной платы изображён на рисунке 53.

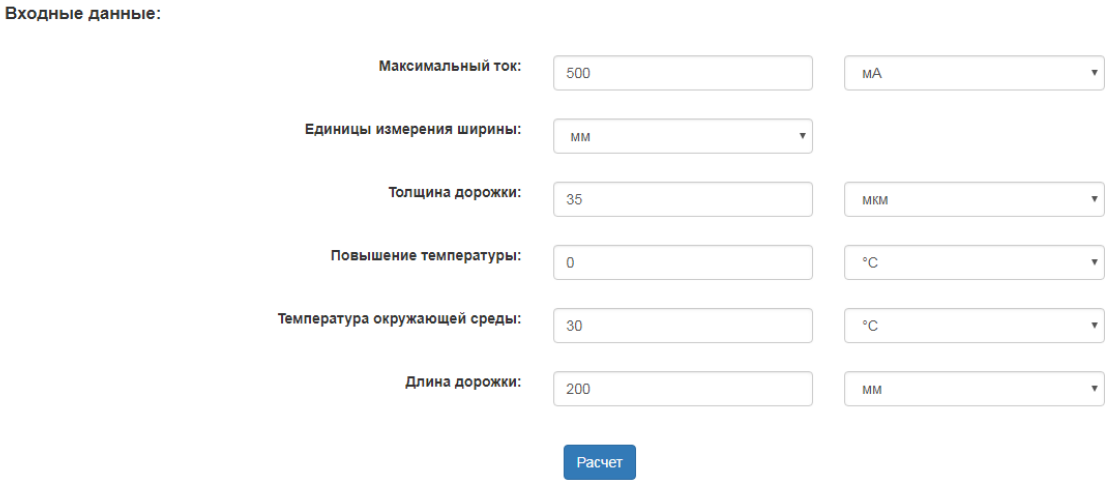

Рисунок 50 – Начальные данные для расчёта.

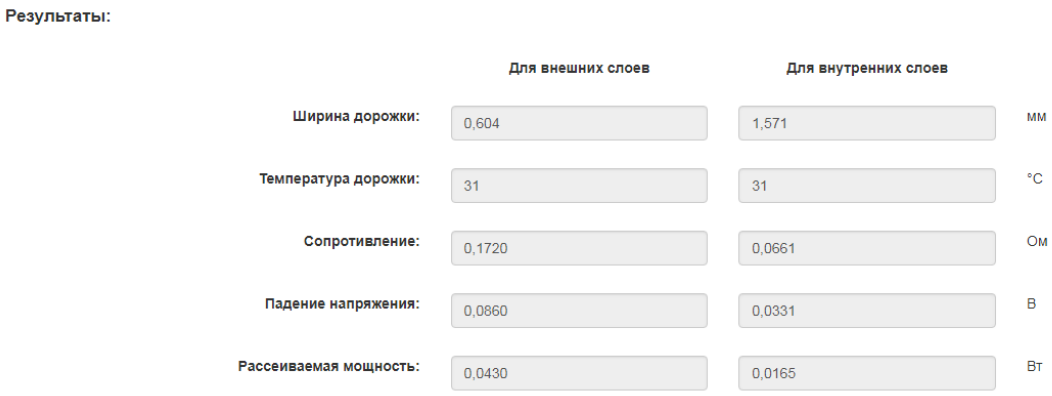

Рисунок 51 – Результаты расчёта ширины дорожек.

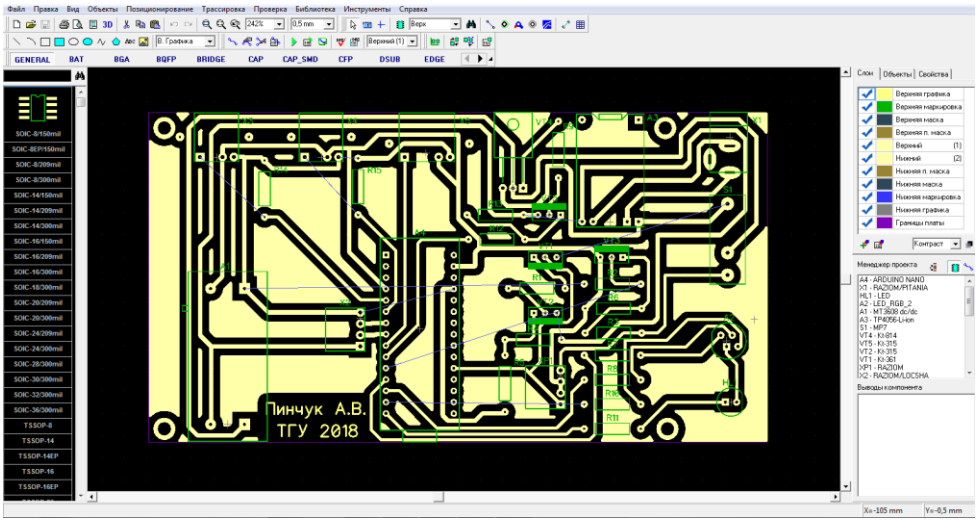

Рисунок 52 – Разработанная печатная плата.

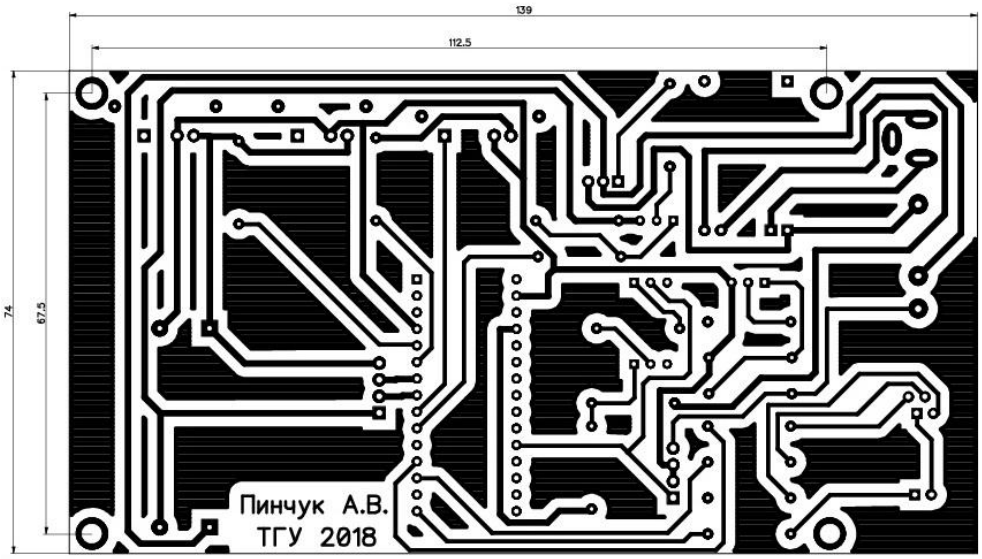

Рисунок 53 – Разработанная печатная плата.

3. Программная часть

3.1. Разработка алгоритма работы

Для выполнения основной и вычислительной работы устройства был выбран микроконтроллер, соответственно для него необходимо написать программу. Для того что бы написать программу, сначала необходимо продумать алгоритм работы и уже на основе алгоритма написать нужную программу.

Разработка алгоритма работы производилась в программе Splan так как она бесплатная для использования. Разработанная блок-схема алгоритмов программы изображена на рисунке 54.

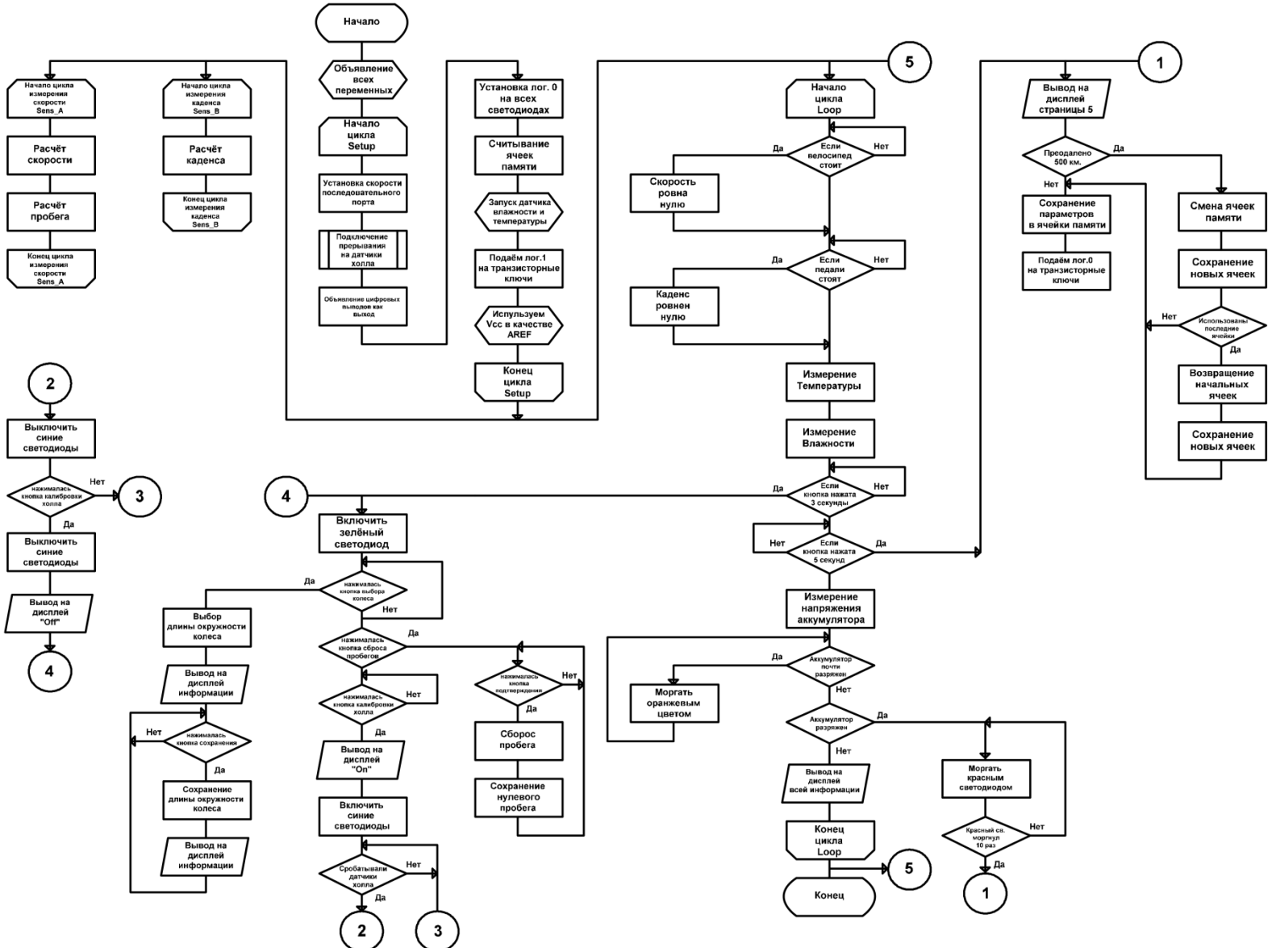

Рисунок 54 – Блок-схема алгоритма программы управления.

3.2. Разработка программы велокомпьютера

Далее на основе разработанной схемы алгоритма работы можно теперь написать программу для микроконтроллера. Полная программа для велокомпьютера с комментариями и пояснениями указана в приложении А.

Как известно среда разработки Arduino построена на библиотеках языке программирования C++ [23]. Программа пишется по принципу блокнота, внешний вид окна программы изображено на рисунке 55.

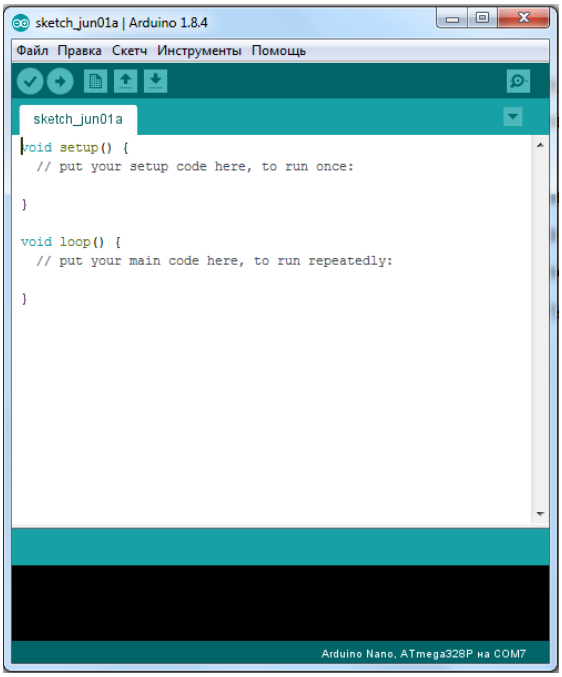

Рисунок 55 – Окно программы среды разработки Arduino.

Так как в схеме имеются модули, которые работают совместно с библиотеками, с самого начала программы подключаются используемые библиотеки.

```
#include <EEPROM.h>   // Библиотека для работы со внутренней памятью ардуино
#include "DHT.h" // Библиотека для работы с датчиками температуры
#include <SoftwareSerial.h> // Библиотека для программного создания последовательного порта, на нужных цифровых выводах
#include <Nextion.h> // Библиотека для работы с дисплеем
```

После чего объявляются все переменные для работы в программе. Со всеми переменными можно ознакомится в приложении А. Далее следует цикл setup, в котором устанавливаются все начальные значения и выполняются первые команды. Данный алгоритм выполняется один раз при каждом включении микроконтроллера. В данном цикле устанавливается скорость передачи последовательного порта для работы с дисплеем.

После чего подключается прерывание для выводов, к которым подключены датчики Холла. Это необходимо, что бы в любое время выполнения любого куска кода программы производился подсчёт скорости и расстояния.

Так же из ячеек памяти микроконтроллера считываются сохранённые значения длины окружности колеса, необходимые для правильного подсчёта скорости и расстояния. Значения полного пробега и суточного пробега после запятой. Хоть и значение после запятой не выводится, оно сохраняется, что бы была актуальность значения полного пробега.

А так же запускается датчик температуры и влажности. Так как этот цикл очень объёмный, с ним можно ознакомится в приложении А.

После чего следует цикл подсчёта скорости void sens_a и цикл подсчёта каденса педалей void sens_b.

```
void sens_a() // Произведём подсчёт текущей скорости и суточного пробега
{
 if (millis()-lastturn > 80) // Защита от случайных измерений (основано на том, что велосипед не будет ехать быстрее 120 кмч)
  {
   SPEED=w_length/((float)(millis()-lastturn)/1000)*3.6;  // Расчет скорости, км/ч
   lastturn=millis();  // Запомнинаем время последнего оборота
   DIST=DIST+w_length/1000;    // Прибавляем длину колеса к дистанции при каждом его обороте (делим на 1000 для перевода метров в километры)
   flag_sum=1; // Присваиваем переменной значение 1
   flag_vk=1; // Присваиваем переменной значение 1
  }
}
```

```
void sens_b() // Произведём подсчёт текущих оборотов педалей
{
 RPM=60/((float)(micros()-lastflash)/1000000);  // Расчёт оборотов, об/мин
 lastflash=micros();  // Запоминаем время последнего оборота
}
```

Скорость — векторная физическая величина, характеризующая быстроту перемещения и направление движения материальной точки относительно выбранной системы отсчёта; по определению, равна производной радиус-вектора точки по времени.

По заданию необходимо подсчитывать скорость движения велосипеда, из этого следует, что скорость можно посчитать по формуле 5

$$V = \frac{L}{T}, \tag{5}$$

где, V – скорость, Километров в час;

L – длина окружности колеса, Сантиметр;

T – период (время одного оборота), Секунда.

Для того что бы вычислить скорость, сначала нужно вычислить период колеса, то есть время, за которое совершается один полный оборот колеса, вычислить период можно по формуле 6

$$\frac{1}{T} = n, \tag{6}$$

где, n – частота оборотов.

Так же помимо скорости рассчитывается и пробег велосипеда, так как длина окружности колеса известна, подсчитать пробег становится элементарно. Объявляем переменную и при каждом полном обороте колеса, прибавляем к ней значение длины окружности колеса.

Помимо скорости также необходимо подсчитывать каденс педалей, это очень актуально при поездках на велосипеде. Каденс это количество оборотов педалей в минуту.

Оборот в минуту (обозначение об/мин, также часто используется английское обозначение rpm revolutions per minute) — единица измерения

частоты вращения: количество полных оборотов, совершенных вокруг фиксированной оси.

Теперь рассмотрим, как подсчитать количество оборотов в минуту педалей велосипеда. Из указанного выше определения следует, что количеств оборотов можно посчитать по формуле 7.

$$\mathrm{rpm} = \frac{60}{(\ t_{\text{н}} - t_{\text{п}}\ \ 1000000)} \tag{7}$$

где, $\mathrm{rpm} - $ оборот в минуту ;

$t_{\text{н}} -$ настоящее время за которое совершается один оборот, микросекунд;

$t_{\text{п}} -$ прошлое время за которое совершается один оборот, микросекунд.

После чего идут функции, необходимые для измерения напряжения микроконтроллером. Данный код, а так же эти функции были взяты из примера к данной среды разработки Arduino.

Далее начинается основной цикл всей программы loop, которая выполняется в цикличном режиме бесконечно. С полной версией данного цикла можно ознакомится в приложении А.

Сначала идёт цикл обработки скорости и пробега. Выполняет следующие функции, за счёт условия. Если больше секунды не наблюдается срабатывания датчика Холла, то присваиваем переменной скорости значение ноль. Так же, если велосипед остановился и скорость равна нулю, тогда весь пробег, набранный за время последнего движения велосипеда, прибавляется к полному пробегу.

После расположен цикл обработки оборотов. Данный цикл выполнен на основе условия, если больше секунды не наблюдается срабатывания датчика Холла, то присваиваем переменной оборотов значение ноль.

После чего следует цикл измерение температуры и влажности с датчика. Каждой заранее назначенной переменной присваивается своё

значение, а именно одной температура, а второй влажность. Измерения производятся каждые 250 миллисекунд.

Как можно видеть из приложения А, следующим расположен цикл сохранения пробега. Данный цикл достаточно большой и выполняет несколько функций. Построен он на нескольких условиях.

Первое условие выполняется при выключении велокомпьютера. Когда запускается цикл отключения, сначала на дисплей велокомпьютера выводится соответствующее сообщение по команде от микроконтроллера. Затем происходит сохранение полного пробега, а так же значение суточного пробега после запятой. После чего велокомпьютер отключается. А отключается он, устанавливая логический ноль на выводе микроконтроллера, отвечающий за открытие и закрытие транзисторного ключа, коммутирующий напряжение от аккумулятора к преобразователю напряжения.

Так же имеется условие в данном цикле, которое создано с целью продления времени пользования ячейками памяти микроконтроллера. Так как если постоянно использовать одну ячейку памяти, она быстро износится и перестанет хранить данные. Данное же условие срабатывает через каждые 500 километров преодолённые хозяином велокомпьютера. Когда условие становится верным, ячейки памяти сдвигаются на следующие по порядку. И так данный цикл повторяется пока не дойдёт до последних ячеек.

Третье же условие создано с целью, когда велокомпьютер работает с последними ячейками памяти, и по условию необходимо опять перейти на следующие новые ячейки, данной условие, возвращает самые первые начальные ячейки, после чего всё снова будет повторяться.

Следующим большим циклом, является тот, который отвечает за выбор длины окружности колеса. В данном цикле содержаться значения длины окружности колес для всех распространённых разновидностей. Реализовано всё на множестве условий, каждое из которых содержит определённое значение, длины окружности колеса и информацию об этом, отправляя её на дисплей велокомпьютера. Управляется это условие виртуальными кнопками

с дисплея, одна из них ведёт вверх по значениям, другая вниз. После того когда нужное значение выбрано, при нажатии на третью виртуальную кнопку, данное значение записывается в ячейку памяти микроконтроллера и на дисплей отправляется сообщение ok сообщая о том, что значение успешно сохранено.

Далее можно наблюдать цикл сброса пробега. Данный цикл устроен таким образом, что при нажатии на виртуальные кнопки, сбрасывается либо суточный, либо полный пробег, по выбору. Данный цикл построен на условии, если нажата кнопка сброса полного пробега и подтверждение, тогда переменным хранения значений пробегов обнуляются, а так же возвращаются первоначальные ячейки памяти и другие начальные параметры, связанные с подсчётом пробега. После чего всё это сохраняется в ячейках памяти микроконтроллера. Но если нажать на сброс суточного пробега и подтверждение, то обнуляется, только значение суточного пробега не затрагивая полный пробег. По сути это лишь нужно, что бы не перезапускать велокомпьютер.

Далее идёт цикл измерения напряжения [24]. Что выполняет данная часть понятно из названия. Напряжение с аккумулятора подаётся на аналоговый вход, где отцифровывается, сравнивается с внутренним опорным напряжением контроллера 1.1 вольта, и после математических вычислений получаем измеренное значение.

Данная процедура нужна для функции защиты аккумуляторов от глубокого разряда. При помощи нескольких условий это реализовано. Имеется две переменных, одна для предупреждения, что аккумулятор почти разряжен, вторая для отключения велокомпьютера при заданной критической отметки. Данные два значения можно задать при прошивке микроконтроллера, например для предупреждения, 3,2 вольта и 3,0 вольта для отключения. Когда условие становится верным, что настала граница 3,2 вольт. RGB светодиод в схеме начинает моргать оранжевым цветом предупреждая о разряде. При наступлении границы 3,0 вольта, запускается

вместе с первым и второе условие, 10 раз моргает красным цветом совместно с оранжевым и после 10 раз отключается велокомпьютер, автоматически.

Так как в схеме присутствует кнопка, то и должна быть возможность управления при помощи неё. Так и было сделано в данном устройстве, так как следующим идёт цикл управления с кнопки и выключения. Данная кнопка имеет изначально функцию включения велокомпьютера, путём замыкания контактов транзисторного ключа. Но когда устройство уже включено, данная кнопка выполняет две функции, это выключение устройства, а так же переход в режим внесения настроек. Из-за того что у кнопки всего один свободный контакт, который можно задействовать, а функции которые необходимо задавать этой кнопкой две, было принято решение сделать всё на нескольких условиях.

Мы зажимаем кнопку, на входе микроконтроллера появляется логическая единица, выполняется условие, при котором переменная, созданная для управления другими условиями, начинает увеличиваться. Для чего это сделано. Значение переменной один, мы зажали кнопку, после чего выполняется функция сложения, то есть у переменной становится значение два, происходит секундная пауза, затем три и так далее. Когда значение доходит до трёх, одно из условий становится верным, после чего начинает светиться RGB светодиод, что говорит о том, что велокомпьютер включается в режим настройки, и он включится, если отпустить кнопку, тогда начнёт выполняться другое условие, о котором ниже. Если условие начало выполняться, то значение для данной переменной присваивается другое, десять и, удерживая кнопку, мы доводим его до тринадцати и тогда срабатывает условие отключения режима настройки и обнуление переменной.

Но если, кнопку не отпустить, то цикл выполнения продолжится, и на значении пять сработает второе условие, RGB светодиод засветится красным цветом, и это будет означать, что выполняется вторая функция, возложенная на данную кнопку, отключение велокомпьютера, точнее это условие,

запускает условие сохранение и отключение показанное выше. Возможно при прочтении это сложно понять, но если читать это и просматривать данную часть программы в приложении А, то всё должно быть понятно.

После чего виден предпоследний цикл программы, который называется цикл диагностики. Называется он так, потому что создан для того что бы визуально наблюдать за работой датчиков Холла. А наблюдать за ними необходимо для их установки или замене или если необходимо убедиться в их работоспособности. Когда включен режим настройки велокомпьютера, нужно нажать на виртуальную кнопку на дисплее, тогда запустится условие, которое отправит на дисплей сообщение on, говорящее о том, что данная функция включилась и выполнится следующее условие, которое выключит зелёный светодиод. Когда на входах, к которым подключены датчики Холла будет появляться логический ноль, светодиоды будут включаться и когда логическая единица, они будут выключаться, так и работает данный цикл программы. При её выключении необходимо так же нажать на виртуальную кнопку, расположенную на дисплее и на дисплей пришлётся сообщение off, сообщающее нам, что данная функция отключилась, а так же сработает первое условие, которое включит зелёный светодиод.

И последний цикл в данной программе, это цикл отправки информации на дисплей. Тут всё предельно просто, первое условие служит для того, что бы при движении велосипеда на дисплей отправлялись только нужные в данный момент значения. Это суточный пробег, скорость и каденс, при остановке срабатывает второе условие, то есть на диспплей отправляется, что значение скорости и каденса равняется нулю и последнее актуальное значение суточного пробега и автоматически переходит на третье условие, которое при стоянке отправляет остальные оставшиеся параметры в велокомпьютере. Так же когда включается, режим настройки велокомпьютера ни одно из этих условий не выполняется и никакие значения в последовательный порт не отправляются.

На этом разработка программы для контроллера заканчивается, я считаю, что код достаточно подробно прокомментирован и если его читать вместе с данным пояснением, то всё должно быть понятно. Полноценный код программы для микроконтроллера показан в приложении А.

Однако ещё необходимо создать программу для дисплея. Дисплей программируется крайне легко, так как программирование визуальное. В специальной программе для разработки Nextion Editor, создаётся рабочий экран устройства, на которые из выбранного меня помещаются блоки, а именно кнопки, окно для текста и так же имеются другие. Каждому блоку присваиваются уникальное имя в пределах проекта и всё, вводятся некоторые настройки, это текс и тип шрифта и на этом всё. С блоком можно работать из ардуино. Пример программы для разработки показан на рисунке 56. Ниже приведены рисунки 57,58,59,60 и 61. показывающие программу дисплея велокомпьютера.

Во время работы в программе, было разработано пять страниц для дисплея, первая страница изображена на рисунке 57. Данная страница является начальной, которая приветствует пользователя при включении устройства, после того как будет нажата кнопка спасибо, велокомпьютер больше не вернётся на данную страницу, пока его снова не включить.

Вторая страница изображена на рисунке 58. Она является одной из основных и служит для индикации значений скорости велосипеда, каденса педалей, а так же суточного пробега. Двумя виртуальными кнопками можно переключиться на страницу значений при стоянке и на страницу настроек.

Страница три изображена на рисунке 59. Данная страница отображает параметры температуры, влажности и полный пробег велокомпьютера.

Предпоследняя, четвёртая страница изображена на рисунке 60. Данная страница создана для возможности настройки велокомпьютера. То есть на этой странице расположены все виртуальные кнопки, влияющие на настройку устройства. Слева, кнопка вверх и вниз, а так же сохранить, используется для выбора значения длины окружности колеса. В середине

кнопка сброса суточного пробега, полного и подтверждение действия. Справа кнопка для включения и выключения визуального контроля над датчиками Холла.

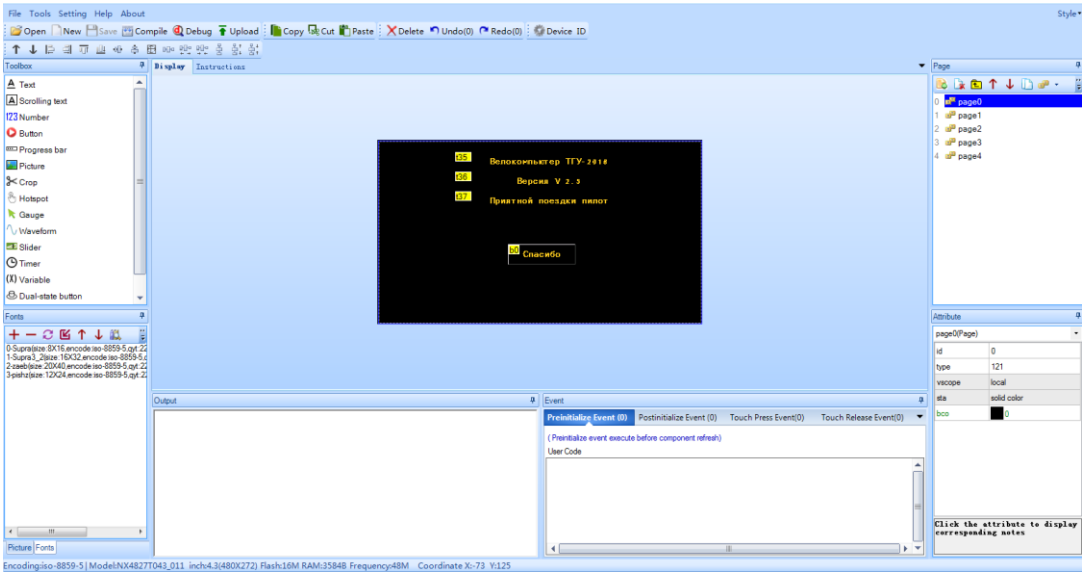

Рисунок 56 – Пример вида окна программы Nextion Editor.

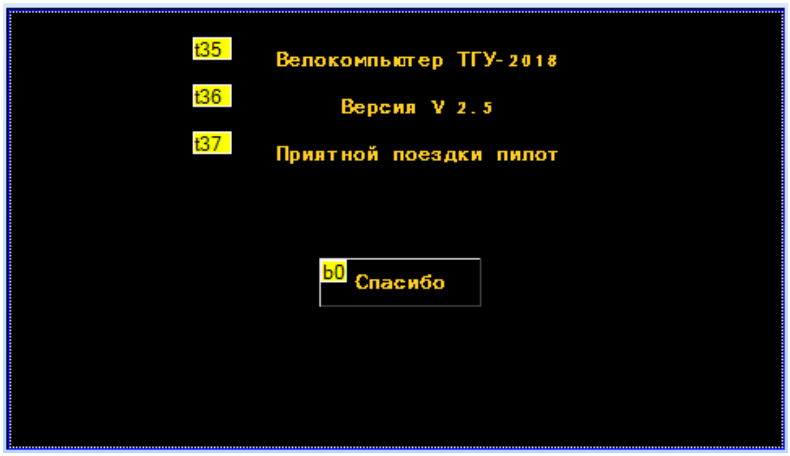

Рисунок 57 – Разработанная первая страница дисплея.

И последняя пятая страница изображена на рисунке 61. Используется для вывода сообщения во время выключения велокомпьютера.

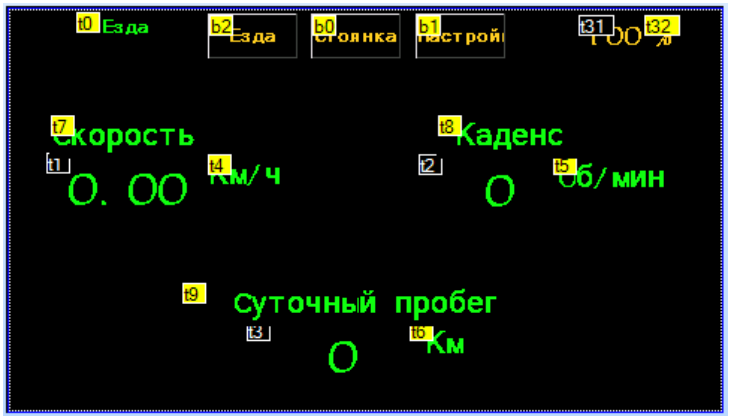

Рисунок 58 – Разработанная вторая страница дисплея

Рисунок 59 – Разработанная третья страница дисплея.

Рисунок 60 – Разработанная четвёртая страница дисплея.

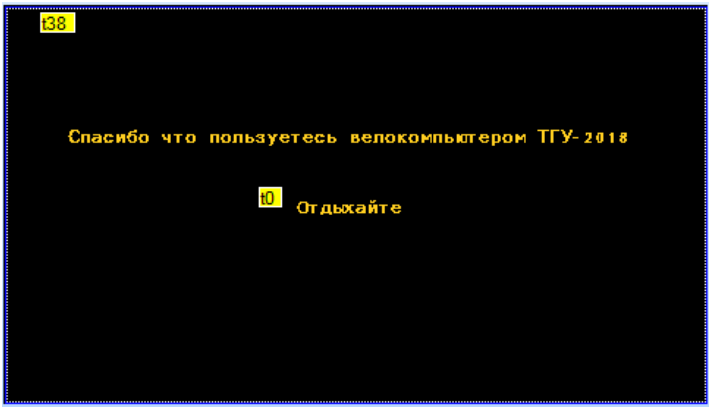

Рисунок 61 – Разработанная пятая страница дисплея.

4. Изготовление велокомпьютера

После того как разработана плата электрическая принципиальная, рассчитаны и выбраны элементы схемы, а также разработана печатная плата и написана программа для работы устройства, дальше необходимо собрать велокомпьютер.

Изначально, для сборки необходимо изготовить печатную плату. Для печатной платы велокомпьютера был выбран фольгированный стеклотекстолит толщиной 1,5 миллиметра и толщиной медной фольги 35 микрометров, один из самых распространённых и доступных значений для фольгированного стеклотекстолита.

Далее стеклотекстолит был нарезан до нужного размера спроектированной печатной платы. Затем были выполнены подготовительные работы, а именно, зачищены края платы, что бы об них нельзя было, поранится, данная операция необходима при изготовлении платы. После чего бы очищен от окислов медный слой, и обезжирен спиртом в последующем.

В последующем необходимо нанести рисунок на медный слой платы. Так как изготавливается рабочий прототип устройства в единичном экземпляре, и плату приходится изготавливать вручную, то был выбран следующий способ нанесения. Путём печати и перевода рисунка на плату. На глянцевой бумаги был распечатан рисунок печатной платы, затем данный рисунок был уложен на медном слое платы. После чего путём нагревания стеклотекстолита до примерно двухсот градусов рисунок дорожек был перенесён на медный слой платы.

Затем были удалены излишки меди, химическим способом. Плата была помещена в ёмкость с хлорным железом на пятнадцать минут, пока вся медь не стравилась. Данный процесс изображён на рисунке 62. Когда ожидаемый результат был, достигнут, плата была извлечена из раствора и очищена от раствора и рисунка. Вытравленная печатная плата изображена на рисунке 63.

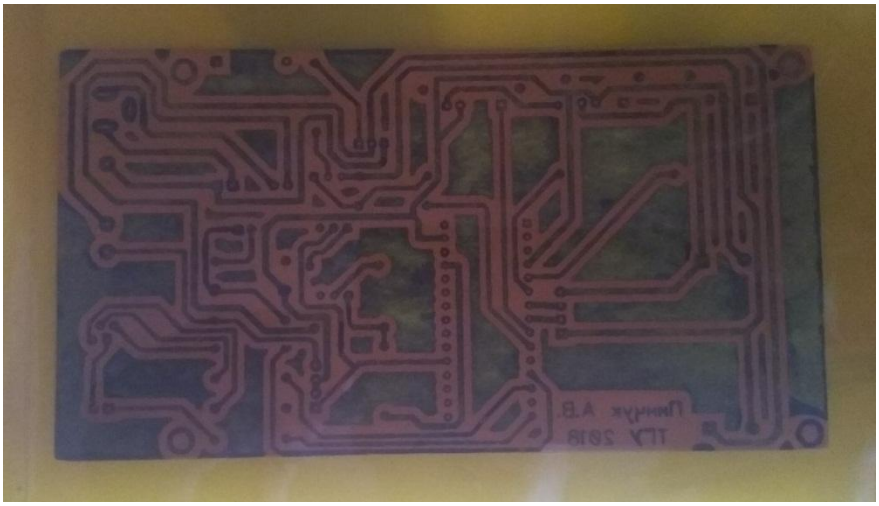

Рисунок 62 – Плата во время травления в химическом растворе.

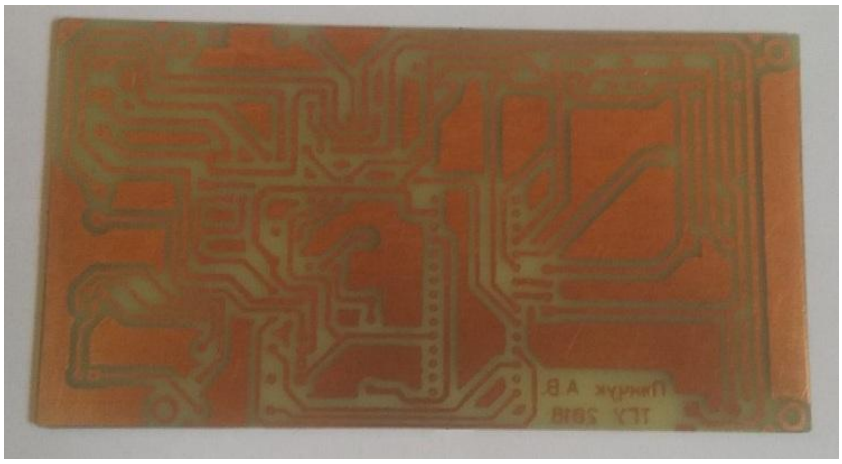

Рисунок 63 – Вытравленная и отчищенная плата.

Для защиты медного слоя от окисления дорожки печатной платы были покрыты припоем, то есть залужены. А потом просверлены все технические отверстия для крепления и отверстия под установку радиоэлементов и модулей устройства. Подготовленная к пайке плата изображена на рисунке 64.

Далее последовательно устанавливались элементы и модули устройства на печатную плату, а затем их вывода припаивались. После того как все элементы и модули были впаяны, производилась впайка перемычек

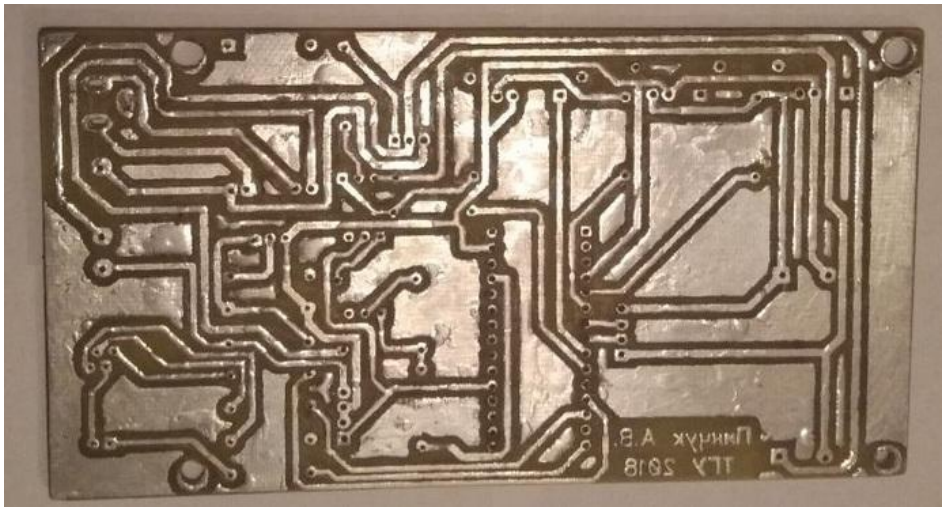

Рисунок 64 – Подготовленная к пайке печатная плата.

на плате. И затем уже была выполнена смывка флюса спиртосодержащей жидкостью. Готовая печатная плата изображена на рисунках 65 и 66.

Далее в ходе сбоки был подключён дисплей к велокомпьютеру через соответствующий разъём на плате. А так же самые высокие части схемы были заклеены термостойким скотчем для избегания коротких замыканий на плате дисплея, а так же нежелательного трения о плату. Подключение дисплея, а так же место наклейки термостойкого скотча изображены на рисунках 67 и 68.

После чего произведена конечная сборка велокомпьютера путём жёсткого крепления дисплея к печатной плате. Данное крепление было выполнено при помощи латунных стоек и коротких болтом М3. Полностью собранный велокомпьютер изображён на рисунках 69, 70 и 71.

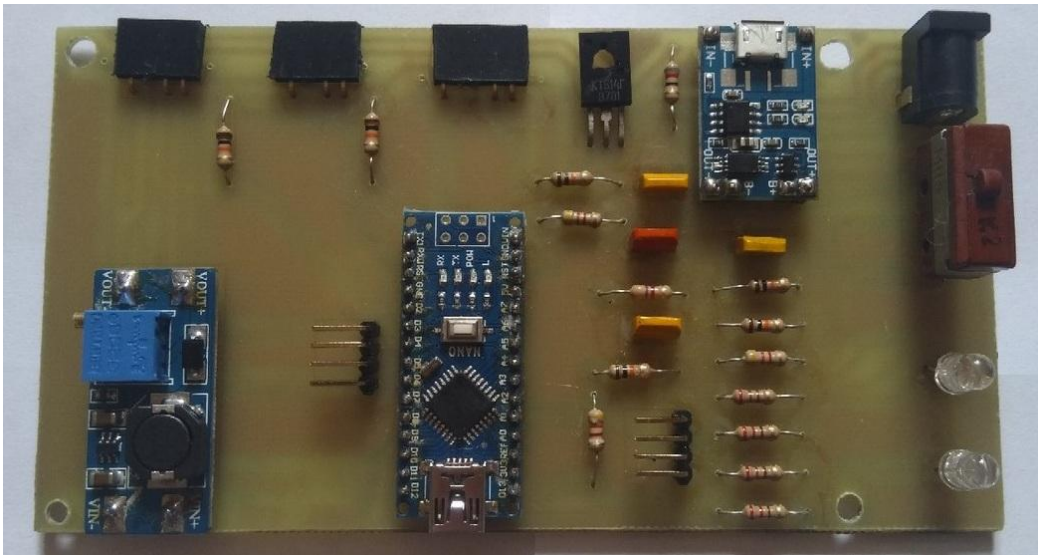

Рисунок 65 – Готовая печатная плата вид сверху.

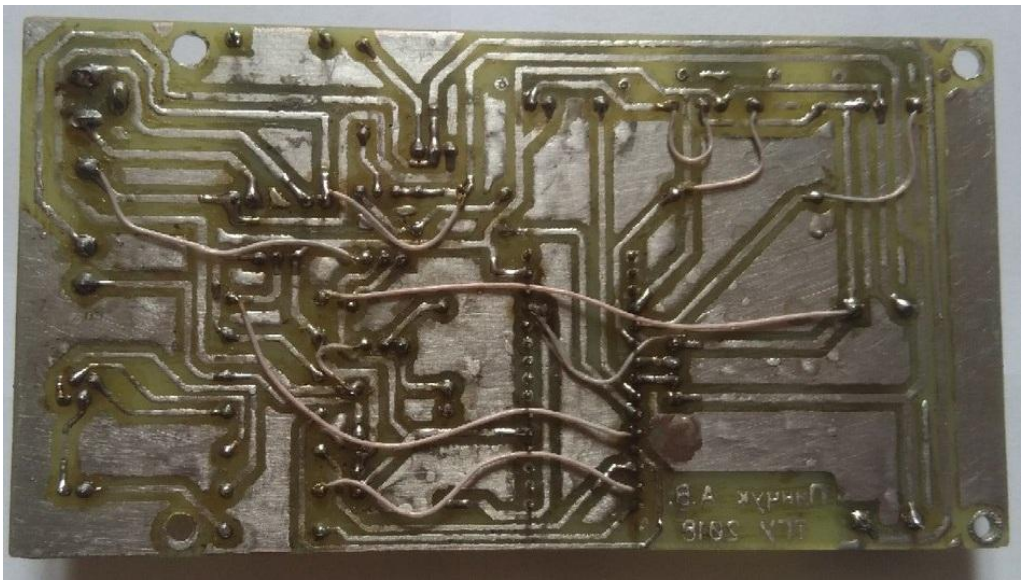

Рисунок 66 – Готовая печатная плата вид снизу.

После самого велокомпьютера была собрана вся периферия, а именно распаяны датчики Холла, датчик температуры, а так же держатель аккумуляторов типоразмера 18650 с разъёмом для подключения к велокомпьютеру.

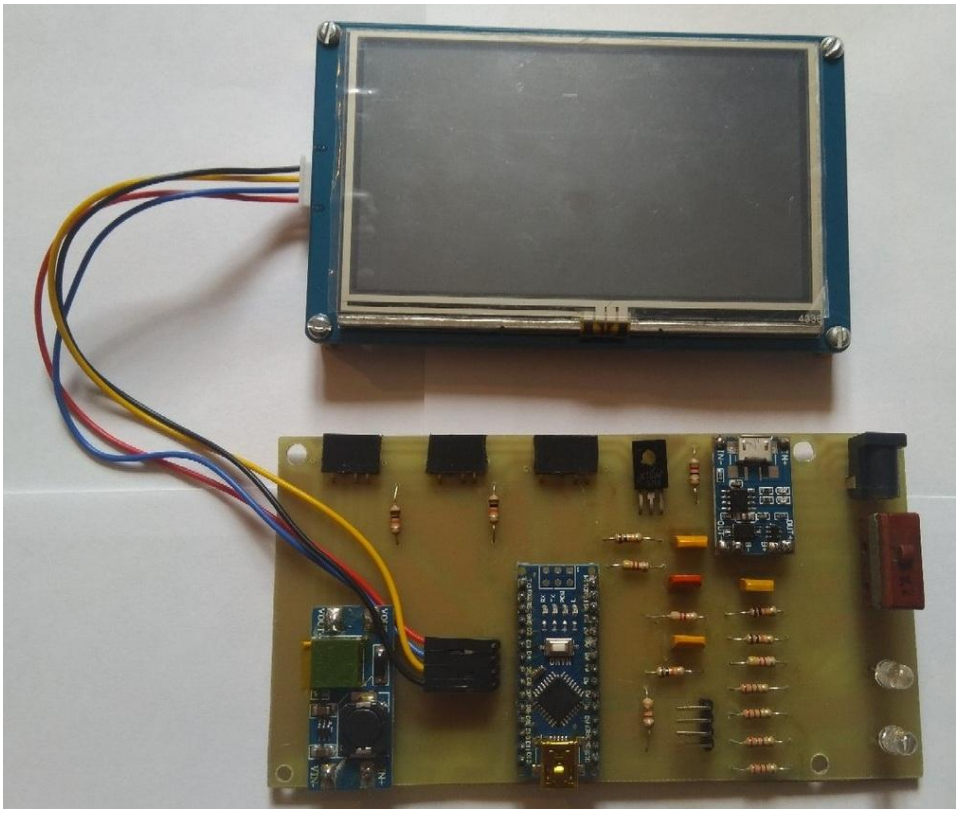

Рисунок 67 – Подключение дисплея в соответствующий разъём.

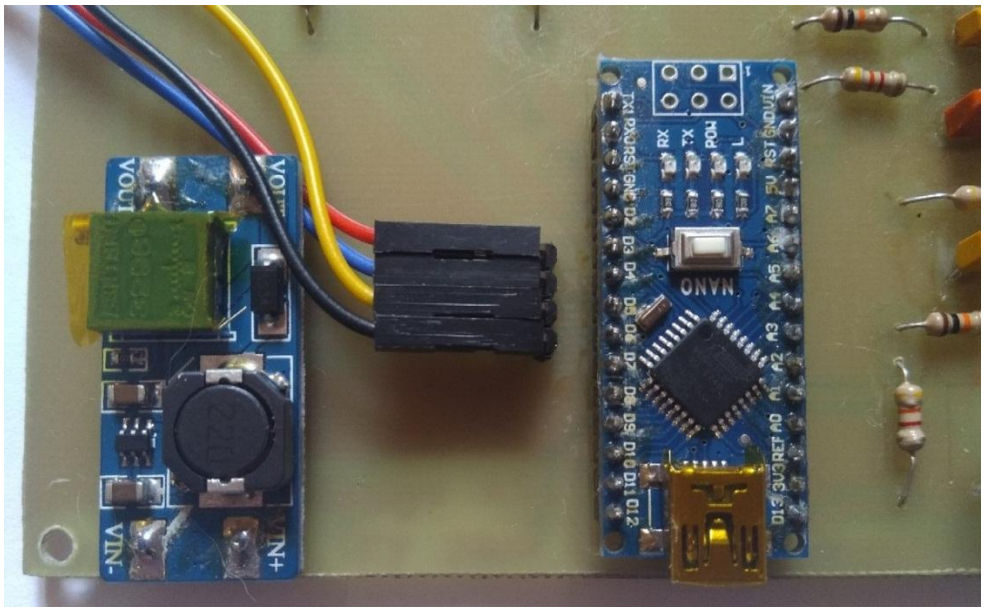

Рисунок 68 – Места, где был наклеен термостойкий скотч.

Датчики Холла установлены на маленькой плате со световой индикацией для наглядности работы. К ним припаян разъём изготовленный из планки PLS, из которой был удалён один вывод что бы данный датчик нельзя было вставить в другие разъёмы. Подключение разъёма к датчику Холла выполнено гибким многожильным проводом типа МГТФ сечением 0,12 квадратных миллиметра. Готовый датчик холла с разъёмом изображён на рисунках 72 и 73.

Рисунок 69 – Собранный велокомпьютер вид сверху.

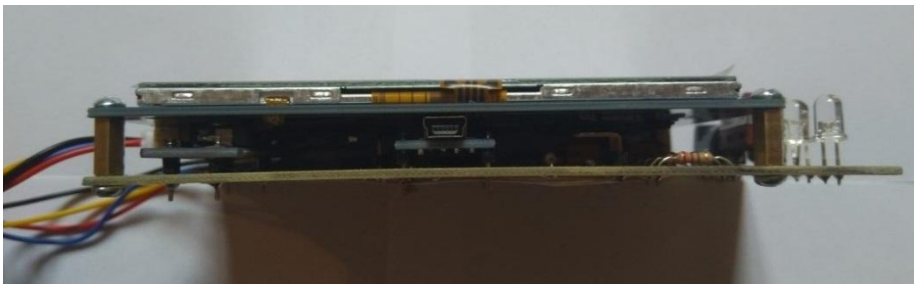

Рисунок 70 – Собранный велокомпьютер вид сбоку.

Датчик температуры подключён к разъёму изготовленный из планки PLS, из которой был удалено два вывода, что бы данный датчик нельзя было вставить в другие разъёмы. Подключение разъёма к датчику температуры

выполнено гибким многожильным проводом типа МГТФ сечением 0,12 квадратных миллиметра. Так же сигнальный вывод подтянут к питанию

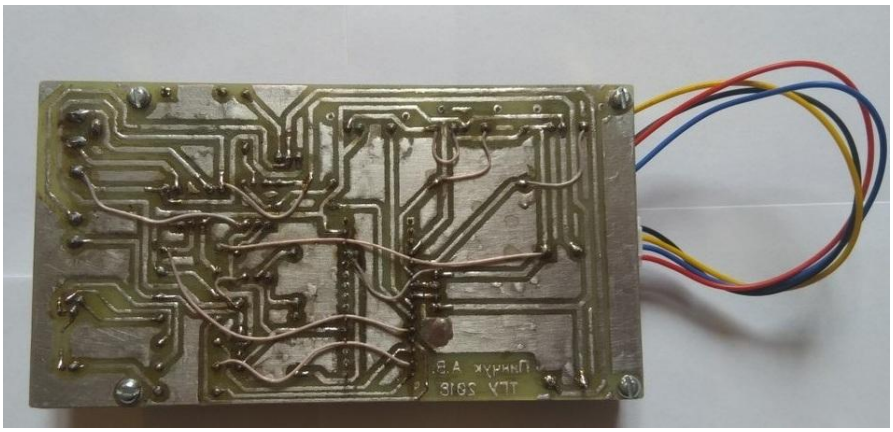

Рисунок 71 – Собранный велокомпьютер вид снизу.

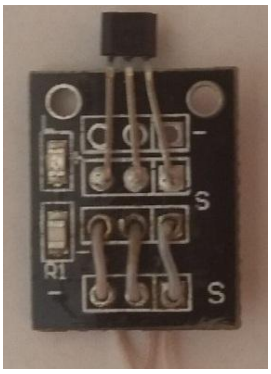

Рисунок 72 – Установленный на плату датчик холла.

через резистор 10 килоом для избегания ложных показания и помех. Данный резистор припаян прямо на датчике. Готовый датчик температуры с разъёмом изображён на рисунках 74 и 75.

Для подключения питания к велокомпьютеру выбранного держателя для аккумуляторов был припаян провод типа PSC (01-6105-6) сечением 1 квадратный миллиметр, а так же подключён разъём типа JR-55 с клеммной колодкой. Готовый держатель для аккумуляторов с разъёмом изображён на рисунках 76 и 77.

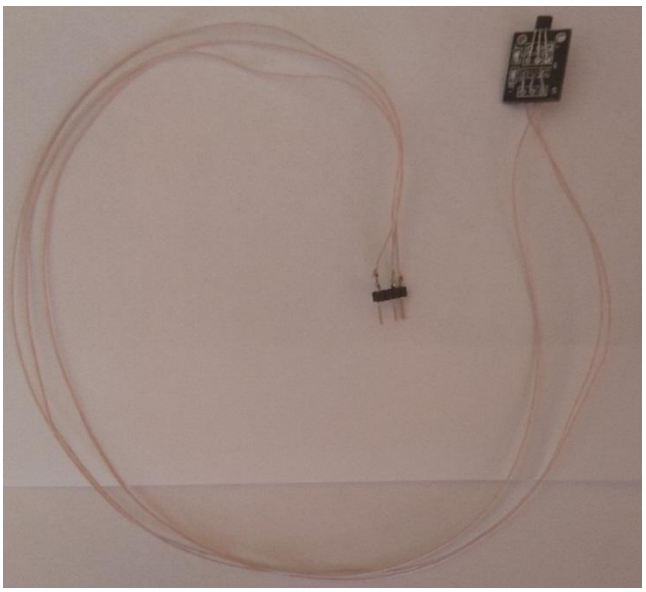

Рисунок 73 – Готовый датчик холла с разъёмом.

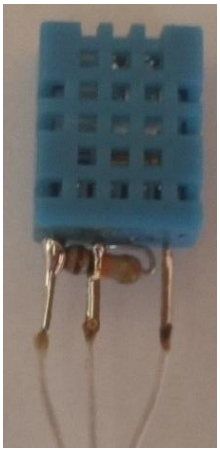

Рисунок 74 – Датчик температуры с установленным резистором.

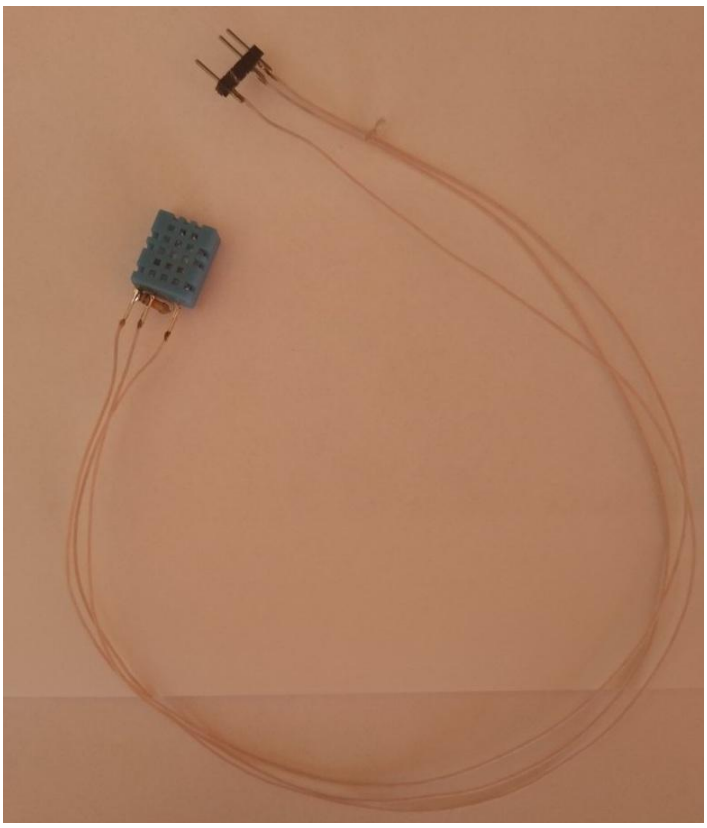

Рисунок 75 – Готовый датчик температуры с разъёмом.

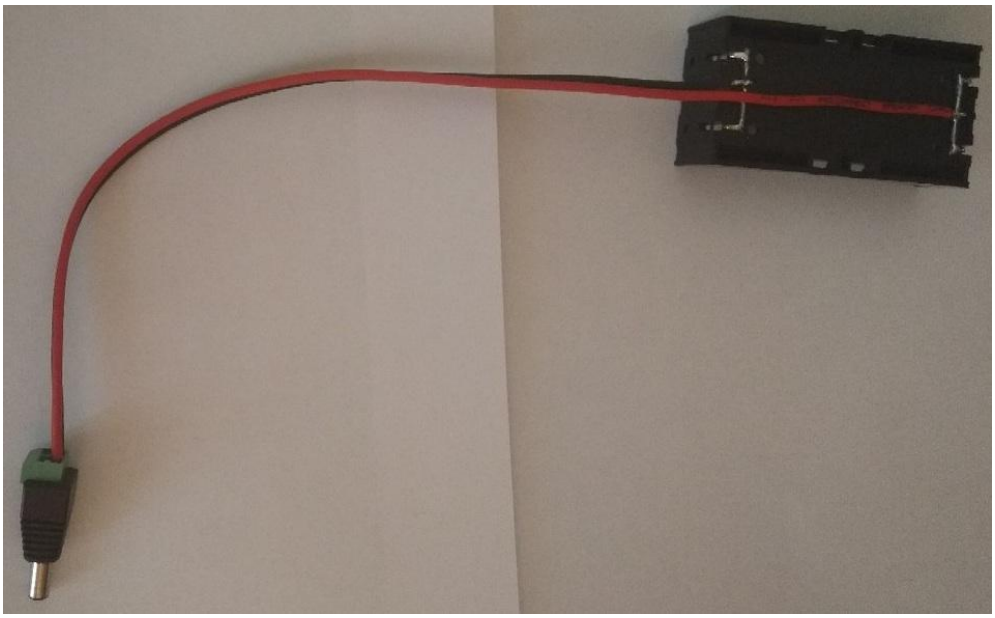

Рисунок 76 – Готовый держатель аккумуляторов с разъёмом.

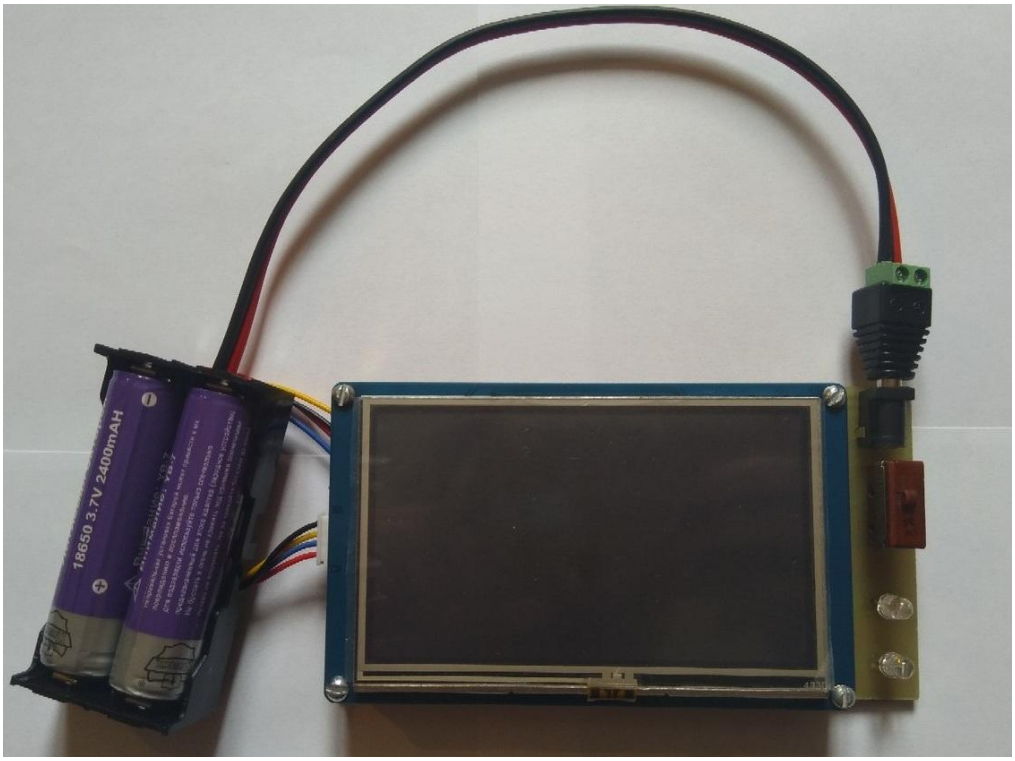

Рисунок 77 – Велокомпьютер с подключёнными аккумуляторами.

5. Отладка и экспериментальные исследования

В ходе работы над устройством не раз вносились изменения в схему устройства. Связанно это с тем, что невозможно предусмотреть сразу всё, всегда что-то будет упущено и потом методом проб и ошибок это исправляется.

Перед тем как был изготовлен велокомпьютер, сначала собирался макет, на котором отрабатывалась вся схема и программа устройства. Общий вид макета изображён на рисунок 78. Но, не смотря на пугающий вид на нём

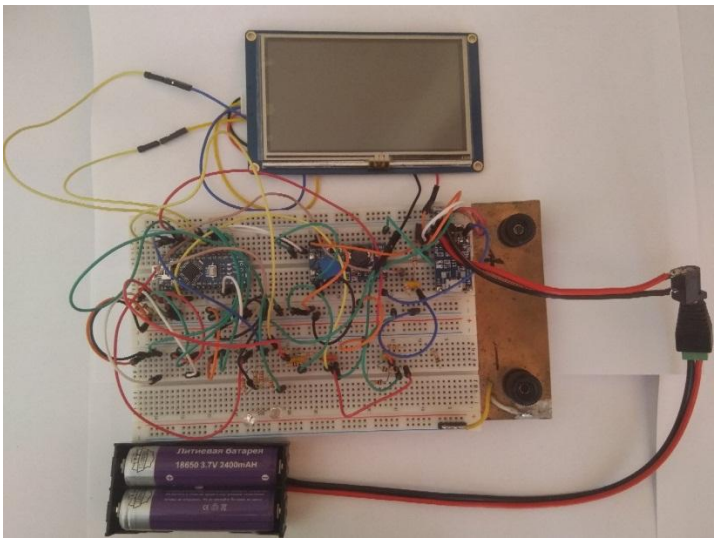

Рисунок 78 – Общив вид макета устройства

удачно удавалось отрабатывать все недочёты схемы. Пример работы макета изображён на рисунке 79.

В ходе работы с макетом были выявлены и устранены следующие недостатки. При отключении датчиков Холла из-за наводок выводились случайные значения скорости и каденса. При том, что в случае со скоростью создавалась и вторая проблема, из-за особенности программы подсчёта пройдённого расстояния, из-за случайных скоростей велокомпьютер так же

считал что он проезжает какое-то расстояние согласно с заданным алгоритмом подсчёта. Соответственно такие недочёты в схеме недопустимы.

Для решения данной проблемы, на входа микроконтроллера к которым подключаются датчики Холла были установлены подтягивающие резисторы на 10 килоом.

Данные резисторы подтягивают входа микроконтроллера на питание, то есть на данных входах есть всегда стабильная логическая единица, а так как когда данная логическая единица присутствует, велокомпьютер ничего не считает. Когда датчик Холла срабатывает, на его выводе появляется минус питания, соответственно на входах появляется логический ноль, после чего и запускается алгоритм, а резисторы исключают короткое замыкание по питанию схемы.

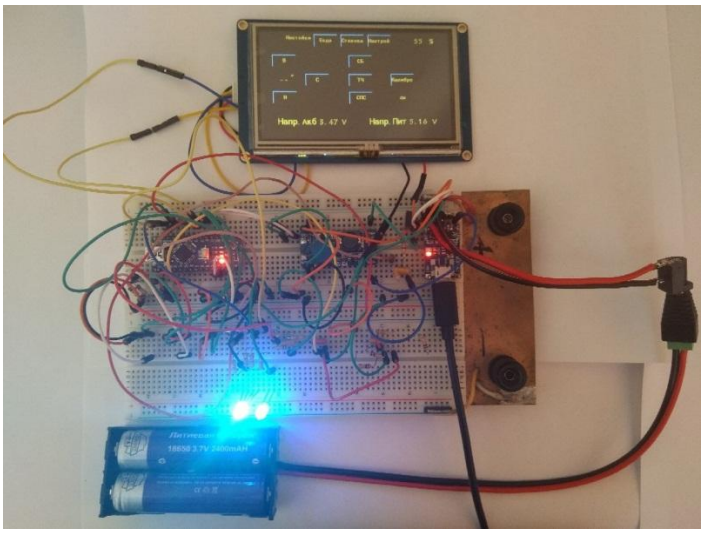

Рисунок 79 – Пример работы с макетом устройства.

Так же изначально для измерения напряжения аккумуляторов, они были подключены напрямую к аналоговому входу микроконтроллера. После чего выяснилось, что при попытке отключить устройство, оно продолжает потреблять напряжение через аналоговый вход и микроконтроллер не отключается, а продолжает функционировать.

После чего в схему был включён транзисторный ключ, что бы при отключении питания, он отключал аккумуляторы от аналогово входа и частично, это помогло, но появились новые проблемы.

После этого велокомпьютер стал неточно измерять напряжение. Но это явная проблема, связанная с тем, что на кремниевом транзисторе всегда присутствует падение напряжения в 0,6 вольта. Для решения данной проблемы, в схему был включён технологический разъём для измерения напряжения до ключа и после и внесения этой разницы в велокомпьютер для корректировки измерения.

Но так же появилась и вторая проблема. Изначально данный транзисторный ключ подключался напрямую к питанию, то есть на выход преобразователя и идея была такова, что при включении питания ключ сразу открывается сам, а при отключении закрывается, и устройство благополучно выключается. Но на деле получилось, так что при попытке выключить его, ключ не успевал закрываться и даже резистор между базой и эмиттером транзистора не помогал и ключ оставался открытым, питая микроконтроллер через аналоговый вход.

Для решения данной проблемы, базовый резистор транзисторного ключа вместо того что бы подключаться к выходу преобразователя, подключился к цифровому выходу микроконтроллера и стал, управляется программно, что и решило данную проблему.

Так же в велокомпьютере присутствует настройка датчиков Холла, а точнее визуальный контроль за их срабатыванием при их установке или замене. Изначально эту функцию выполняло микросхема сдвоенный операционный усилитель. Данная микросхема работала в режиме компаратора, так же для её функционирования использовался делитель напряжения из двух резисторов, а так же керамический конденсатор в качестве фильтра по питанию микросхемы, на выходах операционных усилителей два светодиода с токоограничивающими резисторами.

В последующем стало понятно, что микроконтроллере остается, свободные ресурсы для выполнения этой задачи и данная часть схемы лишь занимает место и уменьшает надёжность устройства, так как повышаются шансы поломки. В итоге несколько строк кода позволило избавиться от части радиоэлементов, оставив лишь два светодиода и токоограничивающих резистора к ним.

Так же после сборки устройства, необходимо провести наладку для его дальнейшего функционирования. Необходимо включить велокомпьютер и подключить двух канальный вольтметр или одноканальный поочерёдно сначала к одному входу разъёма, потом ко второму. После чего необходимо получить разницу между входным и выходным напряжением и внести её в программу велокомпьютера.

Дальнейшие настройки должны производиться уже владельцем велокомпьютера, а именно, ввод значения радиуса колеса, это необходимо для корректного подсчёта скорости и пройденного расстояния велокомпьютером. В этом нет ничего сложного.

Начнём с того что любые манипуляции в настройках велокомпьютера следует проводить зажав на несколько секунд кнопку включения велокомпьютера, пока не начнёт светиться зелёный светодиод, затем кнопку нужно отпустить и проводить необходимые действия. Если кнопку не отпустить, то зелёный светодиод перестанет светиться и начнёт светиться красный сигнализирующий о том что выполняется алгоритм сохранения пробега и выключения велокомпьютера.

После всех проведённых манипуляций следует снова зажать на несколько секунд кнопку включения и зелёный светодиод перестанет светиться, сигнализируя о том, что велокомпьютер вышел из режима настроек и снова готов работать.

Ввод правильного радиуса колеса очень прост. На дисплее располагаются три виртуальные кнопки, одна отвечает за перемотку вперёд, а вторая за перемотку назад. Так как значения радиуса выбирается из уже

готовых самых распространённых значений. После того как вы выбрали нужный радиус, он будет написан между кнопками, необходимо нажать на третью кнопку, которая сохраняет данное значение и записывается в энергонезависимую память. Если вы нажали случайно на выбор радиуса, то пока вы не нажмёте на кнопку сохранить, ничего не изменится.

Так же в будущем, когда накопится определённые значения пробега, его можно сбросить, как полный, так и суточный, правда, если выключить и включить суточный пробег сбрасывается автоматически, но значения после запятой не сбрасывается, для сохранения полного актуального пробега, а данная кнопка позволяет сбросить и это.

На дисплее в середине так же имеется три виртуальные кнопки, она сбрасывает суточный, вторая полный пробег. Третья кнопка посередине подтверждает, что вы сбрасываете пробег, это сделано так же в качестве защиты от случайных нажатий.

И наконец, функция визуально контроля за срабатыванием датчиков Холла. Это необходимо при установке и замене датчиков, а так же если есть необходимость убедиться в их работоспособности. В правой части дисплея находится виртуальная кнопка для включения и выключения данной функции, а ниже надпись off, это говорит о том, что данная функция выключена. При нажатии на кнопку, появится надпись on, что говорит о том, что данная функция включилась, а так же начнут светиться два синих светодиода, что тоже даёт понимание, что данная функция активна. При срабатывании одного или двух датчиков холла, светодиоды будут отключаться, можно непрерывно следить сразу за двумя датчиками холла. После того как цель достигнута, необходимо отключить данную функцию. На этом наладка и настройка велокомпьютера заканчивается.

Заключение

В ходе работы над дипломным проектом был разработан велокомпьютер. Данный компьютер позволяет следить за своим здоровьем при тренировках или туристических поездках на велосипеде, путём показаний каденса, пройденного расстояния, а так же вести анализ тренировок путём мониторинга скорости движения и суточного пробега.

Во время работы был проведён анализ современных аналогов, а так же в специальном пакете DipTrace разработана схема электрическая принципиальная, печатная плата устройства, а так же в среде Arduino программа устройства, а так же выполнена наладка и доработка устройства.

Данная работа соответствует заданию, задача и цели проекта достигнуты.

Список используемой литературы

1 Что такое каденс и каким он должен быть при езде на велосипеде. // VeloFans. 2014—2018. URL: http://velofans.ru/sovety/chto-takoe-kadens-kakim-dolzhen-byt-ezde-velosipede. (Дата обращения: 20.05.2018).

2 Дребезг контактов и способы подавления дребезга. // Электронные печеньки. 2014. URL: https://uscr.ru/drebezg-kontaktov-i-sposoby-podavleniya-drebezga/. (Дата обращения: 20.05.2018).

3 Batteries types, differences and features. // Copyright. 2007-2009. URL: http://www.powerinfo.com/accumulatortype.php. (Дата обращения: 21.05.2018).

4 Основные типы и размеры NiMH и NiCd аккумуляторов. // Компания 2А3А. 2009-2016. URL: http://2a3a.ru/razmer_nimh/. (Дата обращения: 21.05.2018).

5 Методы заряда NiMH аккумуляторов и принципы работы «умных» зарядных устройств. // Компания 2А3А. 2009-2016. URL: http://2a3a.ru/charge_nimh/. (Дата обращения: 21.05.2018).

6 Ni-Cd, Ni-MH, Li-Ion batteries. Overview. // Radio force. 2013. URL: http://www.radiosila.com/lastnews/363-ni-cd-ni-mh-li-ion-akkumulator.html. (Дата обращения: 21.05.2018).

7 Литий-полимерный (Li-Pol) и Литий-ионный аккумулятор (Li-Ion). История, преимущества и недостатки. // Onpro. 2014. URL: https://www.onpro.ru/info/article/litiy-pol-li-ion/. (Дата обращения: 21.05.2018).

8 TP4056 datasheet and tutorial. // Chinagoods 2015. URL: http://chinaguds.ru/goods-with-aliexpress/modul-zaryada-micro-usb-tp4056-5v-1a-s-zashhitoj-plata-kontrolya-zaryada-razryada-li-ion-akkumulyatora-18650.html. (Дата обращения: 23.05.2018).

9 Устройство и принцип работы защитного контроллера Li-ion/polymer аккумулятора. // Go-Radio. 2010. URL: http://go-radio.ru/sxema-kontrollera-litiy-ionnogo-akkumulatora.html. (Дата обращения: 23.05.2018).

10 MT3608 overview and datasheet. // Kit-Shop. 2016. URL: http://www.kit-shop.org/publ/elektronika/obzor_mt3608/443-1-0-51. (Дата обращения: 24.05.2018).

11 Мортон Д. UART-USB на FT232RL. // Электронные устройства. 2011. URL: http://samou4ka.net/page/ft232. (Дата обращения: 24.05.2018).

12 Arduino Nano datasheet. // Arduino.cc. 2009. URL: http://arduino.cc/Hardware/ArduinoBoardNano. (Дата обращения: 26.05.2018).

13 Датчик температуры и влажности. // Амперка. 2018. URL: http://wiki.amperka.ru/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D1%8B:troyka-dht11. (Дата обращения: 26.05.2018).

14 Положительный температурный коэффициент. // Большая энциклопедия нефти и газа. 2018. URL: http://www.ngpedia.ru/id115467p1.html. (Дата обращения:).

15 Боржуев А. Датчики Холла. // RoboCraft. 2011. URL: http://robocraft.ru/blog/electronics/594.html. (Дата обращения: 27.05.2018).

16 Nextion Enhanced NX4827K043 - 4.3" display datasheet, tutorials. // MYSKU. 2016. URL: https://mysku.com/blog/china-stores/44446.html. (Дата обращения: 28.05.2018).

17 Тотоев С. FLProg + Nextion HMI. Урок 1. // Habr. 2016. URL: https://habr.com/company/flprog/blog/392561/. (Дата обращения:).

18 Богданович М.И., Грель И.Н., Прохоренко В.А., Шалимо В.В. Всё о цифровых микросхемах: [Электронный ресурс]. 2009. URL: http://asvcorp.ru/tech/digit/index.html. (Дата обращения: 28.05.2018).

19 Как работает компаратор на операционном усилителе. // HUBSTUB. 2015. URL: https://hubstub.ru/circuit-design/53-kak-rabotaet-komparator-na-operacionnom-usiliteleou.html. (Дата обращения: 29.05.2018).

20 Светодиоды - маркировка, характеристика, подключение. // Светодиодный мир нашего века. 2012. URL: https://svetodiode.blogspot.com/2012/01/blog-post_19.html. (Дата обращения:).

21 DipTrace. // СМИ Сайт-ПАЯЛЬНИК. 2018. URL: http://cxem.net/software/diptrace.php (Дата обращения: 30.05.2018).

22 Расчет ширины дорожки печатной платы в зависимости от силы тока: [Электронный ресурс]. RadioProg. 2017. URL: http://radioprog.ru/post/257. (Дата обращения: 30.05.2018).

23 Шотт Д. Arduino. Секретный вольтметр. Правильное измерение напряжения. // Just For Fun. 2012. URL: http://tim4dev.com/arduino-secret-true-voltmeter/. (Дата обращения: 31.05.2018).

24 **Блум, Д**. Изучаем Arduino: инструменты и методы технического волшебства [Текст] / Пер. с англ. — СПб.: БХВ-Петербург, 2015. — 336 с: ил.; 2000 экз. - ISBN 978-5-9775-3585-4.

Приложение А

Программа велокомпьютера

```
//----------------------------------Библиотеки----------------------------
#include <EEPROM.h>   // Библиотека для работы со внутренней памятью ардуино
#include "DHT.h" // Библиотека для работы с датчиками температуры
#include <SoftwareSerial.h> // Библиотека для программного создания последовательного порта
#include <Nextion.h> // Библиотека для работы с дисплеем

//---------------------Переменные для последовательного порта и дисплея--------------------
SoftwareSerial nextion(4, 5);// Цифровой вывод 4 как TX и 5 как RX
Nextion myNextion(nextion, 9600); // Скорость передачи данных для дисплея 9600bps

//---------------------Переменные для подсчёта оборотов----------------------
unsigned long lastflash; // Переменная хранения времени для подсчёта оборотов
int RPM; // Переменная хранения оборотов в виде целых чисел

//---------------------Переменные для подсчёта скорости----------------------
unsigned long lastturn; // Переменные хранения времени для подсчёта скорости
float SPEED; // Переменная хранения скорости в виде десятичной дроби
float w_length=2.050; // Длина окружности колеса в метрах (изначальная 26")
int flag_circle=0; // Переходное значение для выбора длины окружности

//---------------------Переменные для подсчёта и сохранения расстояния----------------------
float DIST; // Переменная хранения суточного расстояния в виде десятичной дроби
int DIST_T; // Переменная хранения суммарного расстояния в виде целого числа
int dist_past; // Переменная хранения пробега при прошлой остановке
int dist_transit; // Переменная для математических действий, переходное
float dist_comma; // Переменная для хранение значения суточного пробега после запятой
int dist_excess; // Переменная хранения суточного пробега без запятой для математических действий
int cell_hi=6; // Номер ячейки памяти для хранения пробега старшего бита
```

```
int cell_low=7; // Номер ячейки памяти для хранения пробега младшего бита
int cell_comma=8; // Номер ячейки памяти для хранения суточного пробега после запятой
int hi_c; // Переменная для старшего бита значения смены ячеек (addr)
int low_c; // Переменная для младшего бита значения смены ячеек (addr)
int hi_d; // Переменная для старшего бита условия смены ячеек (dist_con)
int low_d; // Переменная для младшего бита условия смены ячеек (dist_con)
int dist_con=500; // Переменная хранения значения для условия смены ячеек памяти
int addr=0; // Значение для смены ячеек памяти
byte hi; // Переменная для старшего бита суммарного пробега (DIST_T)
byte low; // Переменная для младшего бита суммарного пробега (DIST_T)
int flag_sum=0; // Переменная для работы с пробегом
int flag=0; // Переменная для выполнения условия сохранения и выключения

//----------------------Переменные для сброса и сохранения расстояния----------------------
int reset=0; // Переменная для условия сброса пробега
int reset_2=0; // Переменная для условия сброса пробега
int reset_3=0; // Переменная для условия сброса пробега

//----------------------Переменные для измерения температуры и влажности------------------
DHT dht(6, DHT11); // Прописываем порт для работы с датчиком
int wet=0; // Переменная для хранения влажности
int temp=0; // Переменная для хранения температуры

//----------------------Переменные для измерения напряжения----------------------------
const byte debug = 0; // Для отладки через консоль поставить значение 2

const float typVbg = 1.025; // Эта константа для измерения напряжения

float Vcc = 0.0; // Переменная для хранения измеренного напряжения

#define A_PIN 1 // Временное имя для константы (аналоговый вход)
#define COUNT 5 // Временное имя для константы

int i; // переменная для оператора for
float curVoltage; // Переменная для хранения
```

```
int flag_leave=1; // Флаг для разрешения (защита аккумулятора, отсечка)
float cut_off=2.90; // Значение отсечки (защиты аккумулятора)
float cut_notice=3.10; // Значение, при котором будет работать световое оповещение
int shift=0; // Переменная для счёта количества моргания светодиода предупреждения

const int led_r=9; // Вывод контроллера для красного цвета RGB светодиода
const int led_g=10; // Вывод контроллера для зелёного цвета RGB светодиода
const int led_b=11; // Вывод контроллера для синего цвета RGB светодиода
const int led_bb=12; // Вывод контроллера для синего светодиода
const int vt_on=13; // Вывод контроллера для включения транзистора измерения

const int blocking_Vdd=7; // Вывод контроллера для самоблокировки питания контроллера

//----------------------Переменные для выключения и управления с кнопки----------------------
const int button_off=8; // Вывод контроллера для контакта кнопки выключения
int flag_button; // Переменная для чтения статуса кнопки
int flag_off=0; // Флаг для условия выключения или управления
int flag_vk=0; // Флаг для условия передачей данных
int flag_vk_2=0; // Флаг для условия передачей данных
int flag_vk_3=1; // Флаг для условия передачей данных

//----------------------Переменные для диагностики----------------------
int flag_mc=0; // Флаг для выполнения условий
int flag_mc_2=0; // Флаг для выполнения условий
int d_2=0; // Переменная для мониторинга датчиков холла
int d_3=0; // Переменная для мониторинга датчиков холла

//----------------------Цикл настройки----------------------
void setup() // Установим нужные параметры в данном цикле
{
 Serial.begin(9600);  // Открываем порт и устанавливаем скорость передачи данных
 myNextion.init("page0"); // Выводим начальную страницу на дисплее
 attachInterrupt(1,sens_a,RISING); // Подключаем прерывание на 3 пин при повышении сигнала
```

attachInterrupt(0,sens_b,RISING); // Подключаем прерывание на 2 пин при повышении сигнала

pinMode(led_r, OUTPUT); // 9 пин будет работать на выход

pinMode(led_g, OUTPUT); // 10 пин будет работать на выход

pinMode(led_b, OUTPUT); // 11 пин будет работать на выход

pinMode(led_bb, OUTPUT); // 12 пин будет работать на выход

pinMode(vt_on, OUTPUT); // 13 пин будет работать на выход

pinMode(blocking_Vdd, OUTPUT); // 7 пин будет работать на выход

digitalWrite (blocking_Vdd, HIGH); // Подадим логическую единицу на 7 вывод (включаем питание контроллера)

digitalWrite (led_b, LOW); // Подадим логический ноль на 11 вывод (выключаем питание кристала)

digitalWrite (led_bb, LOW); // Подадим логический ноль на 12 вывод (выключаем питание светодиода)

digitalWrite (vt_on, HIGH); // Подадим логическую единицу на 13 вывод (включаем питание транзистора измерения)

analogWrite (led_r, 0); // Установим ШИМ красного кристала светодиода на выводе 9

analogWrite (led_g, 0); // Установим ШИМ зелёного кристала светодиода на выводе 10

w_length=(float)EEPROM.read(1)/100.0; // Читаем ячейку памяти длины окружности колеса

hi_d=EEPROM.read(4); // Читаем ячейку памяти старшего бита условия смены ячеек при запуске системы (dist_con)

low_d=EEPROM.read(5); // Читаем ячейку памяти младшего бита условия смены ячеек при запуске системы (dist_con)

dist_con=word(hi_d,low_d); // Преобразовываем старший и младший бит условия смены ячеек в единое значение

hi_c=EEPROM.read(2); // Читаем ячейку памяти старшего бита значения смены ячеек при запуске системы (addr)

low_c=EEPROM.read(3); // Читаем ячейку памяти младшего бита значения смены ячеек при запуске системы (addr)

addr=word(hi_c,low_c); // Преобразовываем старший и младший бит значения смены ячеек в единое значение

cell_hi=cell_hi+addr; // Суммируем значения начального номера ячейки и значение смены ячеек

cell_low=cell_low+addr; // Суммируем значения начального номера ячейки и значение

```
смены ячеек
 cell_comma=cell_comma+addr; // Суммируем значения начального номера ячейки и значение смены ячеек
 hi=EEPROM.read(cell_hi); // Читаем ячейку памяти старшего бита суммарного пробега при запуске системы
 low=EEPROM.read(cell_low); // Читаем ячейку памяти младшего бита суммарного пробега при запуске системы
 DIST_T=word(hi,low); // Преобразовываем старший и младший бит суммарного пробега в единое значение
 DIST=(float)EEPROM.read(cell_comma)/100.0; // Читаем ячейку памяти суточного пробега при запуске системы (деление на 100 нужно для сохранения сотых долей расстояния)
 dht.begin(); // Запускаем датчик влажности DHT11
// определение опорного напряжения
 analogReference(DEFAULT);  // DEFAULT INTERNAL использовать Vcc как AREF
 delay(100); // Делаем паузу 100 мс.
 Vcc = readVcc(); // присваиваем значение переменной Vcc, переменной readVcc

 if (debug > 1) // Если условие верно, то...
  {
   Serial.println("---"); // Выводим в последовательный порт данный символ
   delay(1000); // Делаем паузу, 1 секунду
  }

  if (debug > 1) // Если условие верно, то...
  {
   Serial.print("Vcc = "); // Выводим в последовательный порт надпись
   Serial.println(Vcc); // Выводим в последовательный порт данное значение переменной
   Serial.println("---"); // Выводим в последовательный порт данный символ
  }
}

//----------------------------Цикл обработки скорости и расстояния----------------------------
void sens_a() // Произведём подсчёт текущей скорости и суточного пробега
{
  if (millis()-lastturn > 80) // Защита от случайных измерений (основано на том, что
```

```
велосипед не будет ехать быстрее 120 кмч)
  {
    SPEED=w_length/((float)(millis()-lastturn)/1000)*3.6;  // Расчет скорости, км/ч
    lastturn=millis();  // Запоминаем время последнего оборота
    DIST=DIST+w_length/1000;   // Прибавляем длину колеса к дистанции при каждом его обороте (делим на 1000 для перевода метров в километры)
    flag_sum=1; // Присваиваем переменной значение 1 для условия вывода данных
    flag_vk=1; //  Присваиваем переменной значение 1 для условия вывода данных
  }
}

//------------------------Цикл обработки оборотов-----------------------
void sens_b() // Произведём подсчёт текущих оборотов педалей
{
  RPM=60/((float)(micros()-lastflash)/1000000);  // Расчёт оборотов, об/мин
  lastflash=micros();  // Запоминаем время последнего оборота
}

  //-------------------------------------Функции-----------------------------------------------
float readVcc() {
  byte i;
  float result = 0.0;
  float tmp = 0.0;

  for (i = 0; i < 5; i++) {
    // Read 1.1V reference against AVcc
    // set the reference to Vcc and the measurement to the internal 1.1V reference
    #if defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
      ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
    #elif defined (__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__)
      ADMUX = _BV(MUX5) | _BV(MUX0);
    #elif defined (__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
```

```
    ADMUX = _BV(MUX3) | _BV(MUX2);
  #else
    // works on an Arduino 168 or 328
    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  #endif

  delay(3); // Wait for Vref to settle
  ADCSRA |= _BV(ADSC); // Start conversion
  while (bit_is_set(ADCSRA,ADSC)); // measuring

  uint8_t low  = ADCL; // must read ADCL first - it then locks ADCH
  uint8_t high = ADCH; // unlocks both

  tmp = (high<<8) | low;
  tmp = (typVbg * 1023.0) / tmp;
  result = result + tmp;
  delay(5);
 }

 result = result / 5;
 return result;
 }

//------------------------Остальная часть программы ------------------------
void loop()
{
//------------------------Цикл обработки скорости и расстояния----------------------
 if ((millis()-lastturn)>1000) // Если сигнал отсутствует более 1 секунд, то...
 {
  SPEED=0;  // Считаем что переменная SPEED равняется 0
  if(flag_sum == 1) // Если равенство переменных равно, то... (помимо подсчёта суммарного пробега, защита от ложных сложений суммарного пробега)
  {
   dist_transit=DIST; // Присваиваем промежуточной переменной значение суточного пробега
```

```
    dist_transit=dist_transit-dist_past; // Вычисляем разность суточного пробега на данный момент и значение суточного пробега при прошлой остановке
    DIST_T=dist_transit+DIST_T; // Разность суточного пробега прибавляем к общему пробегу
    dist_past=DIST; // Присваиваем переменной суточного пробега прошлой остановки, значение суточного пробега последней остановки
    flag_sum=0; // Присваиваем переменной разрешения значение 0 для условия вывода данных
    flag_vk_2=1; //  Присваиваем переменной разрешения значение 1 для условия вывода данных
   }
  }

//------------------------Цикл обработки оборотов-----------------------
  if ((micros()-lastflash)>1000000)// если сигнала нет больше 1 секунды, то...
  {
   RPM=0;  // Считаем, что RPM равняется 0

  }

//------------------------Цикл измерения температуры и влажности-----------------------
  wet = dht.readHumidity(); // Считываем температуру (каждые 250 мс)
  temp = dht.readTemperature(); // Считываем влажность (каждые 250 мс)

//------------------------Цикл сохранения пробега-----------------------
  if (flag == 1) // Если равенство переменных равно, то...
  {
   myNextion.init("page4"); // При выключении переключаем на страницу прощания с пилотом
   delay (5000); // Задержка для того что бы успели прочитать текст перед выключением
// Командами highByte и lowByte разделим значение суммарного пробега на старший и младший бит
   hi = highByte(DIST_T); // Присваиваем переменной старший бит значения пробега
   low = lowByte(DIST_T); // Присваиваем переменной младший байт значения пробега
   EEPROM.write(cell_hi, hi);  // Записываем во внутреннюю ячейку памяти старший байт
```

```
  EEPROM.write(cell_low, low); // Записываем во внутреннюю ячейку памяти младший байт
// Данный алгоритм используется для сохранения значения после запятой суточного пробега
  dist_excess=DIST; // Присваиваем значение суточного пробега временной переменной и избавляемся от запятой
  dist_comma=DIST-dist_excess; // Высчитываем разность целого числа и числа с запятой, в итоге остаётся разность суточного пробега после запятой (разрешение0,00)
  EEPROM.write(cell_comma,(float)dist_comma*100.0);  // Записываем во внутреннюю ячейку памяти значение суточного пробега после запятой (умножение на 100 нужно для сохранения сотых долей расстояния)
  digitalWrite (blocking_Vdd, LOW); //Отключаем питание микроконтроллера
 }
// Данный алгоритм используется для того что бы увеличить ресурс работы внутренних ячеек памяти микроконтроллера
 if (DIST_T == dist_con) // Если равенство переменных равно, то...
 {
  dist_con=dist_con+500; // Прибавим переменной условия значение равное 500 км.
  hi_d = highByte(dist_con); // Присваиваем переменной старший бит значения для условия
  low_d = lowByte(dist_con); // Присваиваем переменной младший байт значения для условия
  EEPROM.write(4, hi_d); // Записываем во внутреннюю ячейку памяти старший байт
  EEPROM.write(5, low_d); // Записываем во внутреннюю ячейку памяти младший байт
  addr=addr+3; // Увеличиваем значение переменной на три для переключения ячеек памяти
// Данный алгоритм нужен для ограничения максимального переключения ячеек памяти ,что бы вернуться к первым ячейкам если программа уже на последних
  if (addr == 1017) // Если равенство переменных равно, то...
  {
   addr=0; // Присваиваем данной переменной для переключения ячеек памяти значения ноль
  }
  hi_c = highByte(addr); // Присваиваем переменной старший бит значения переключения ячеек памяти
```

```
  low_c = lowByte(addr); // Присваиваем переменной младший байт значения переключения ячеек памяти
  EEPROM.write(2,  hi_c); // Записываем во внутреннюю ячейку памяти старший байт
  EEPROM.write(3, low_c); // Записываем во внутреннюю ячейку памяти младший байт
  cell_hi=6; // При смене ячеек возвращаем начальное значение переменной
  cell_low=7; // При смене ячеек возвращаем начальное значение переменной
  cell_comma=8; // При смене ячеек возвращаем начальное значение переменной
  cell_hi=cell_hi+addr; // Считаем новое значение переменных значения ячеек памяти
  cell_low=cell_low+addr; // Считаем новое значение переменных значения ячеек памяти
  cell_comma=cell_comma+addr; // Считаем новое значение переменных значения ячеек памяти
 }

//-----------------------Цикл выбора длины окружности колеса-----------------------
  String message = myNextion.listen(); // Ожидаем нужной комбинации символов из последовательного порта для выполнения условий
  if (message == "65 3 6 1 ffff ffff ffff") // Если условие верно, то...
  {
   flag_circle=flag_circle+1; // Инкрементируем значение переменной на 1
  }
  if (message == "65 3 7 1 ffff ffff ffff") // Если условие верно, то...
  {
   flag_circle=flag_circle-3; // Декрементируем значение переменной на 1
  }
  if (flag_circle == 19) // Если условие верно, то...
  {
   flag_circle=1; // Присваиваем значение переменной равное 1 для того что бы оказаться в начале списка
  }
  if (flag_circle == -1) // Если условие верно, то...
  {
   flag_circle=17; // Присваиваем значение переменной равное 1 для того что бы оказаться в конце списка
  }
// Данные условия служат для выбора длины окружности колеса
```

```
  if (flag_circle == 1) // 12,5"
  {
   w_length=0.99; // Присваиваем значение переменной служащей для определения скорости
   myNextion.setComponentText("t22", String(12.5)); // Выводим на дисплей значение выбранной на данный момент длины окружности
   flag_circle=2; // Присваиваем данное значение переменной для дальнейшего ожидания
  }

   if (flag_circle == 3) // 16"
  {
   w_length=1.27; // Присваиваем значение переменной служащей для определения скорости
   myNextion.setComponentText("t22", String(16)); // Выводим на дисплей значение выбранной на данный момент длины окружности
   flag_circle=4; // Присваиваем данное значение переменной для дальнейшего ожидания
  }

   if (flag_circle == 5) // 20"
  {
   w_length=1.59; // Присваиваем значение переменной служащей для определения скорости
   myNextion.setComponentText("t22", String(20)); // Выводим на дисплей значение выбранной на данный момент длины окружности
   flag_circle=6; // Присваиваем данное значение переменной для дальнейшего ожидания
  }

   if (flag_circle == 7) // 24"
  {
   w_length=1.91; // Присваиваем значение переменной служащей для определения скорости
   myNextion.setComponentText("t22", String(24)); // Выводим на дисплей значение выбранной на данный момент длины окружности
   flag_circle=8; // Присваиваем данное значение переменной для дальнейшего ожидания
  }
```

```
  if (flag_circle == 9) // 26"
  {
  w_length=2.07; // Присваиваем значение переменной служащей для определения скорости
  myNextion.setComponentText("t22", String(26)); // Выводим на дисплей значение выбранной на данный момент длины окружности
  flag_circle=10; // Присваиваем данное значение переменной для дальнейшего ожидания
  }

  if (flag_circle == 11) // 27"
  {
  w_length=2.11; // Присваиваем значение переменной служащей для определения скорости
  myNextion.setComponentText("t22", String(27)); // Выводим на дисплей значение выбранной на данный момент длины окружности
  flag_circle=12; // Присваиваем данное значение переменной для дальнейшего ожидания
  }

  if (flag_circle == 13) // 27,5"
  {
  w_length=2.19; // Присваиваем значение переменной служащей для определения скорости
  myNextion.setComponentText("t22", String(27.5)); // Выводим на дисплей значение выбранной на данный момент длины окружности
  flag_circle=14; // Присваиваем данное значение переменной для дальнейшего ожидания
  }

  if (flag_circle == 15) // 28"
  {
  w_length=2.33; // Присваиваем значение переменной служащей для определения скорости
  myNextion.setComponentText("t22", String(28)); // Выводим на дисплей значение выбранной на данный момент длины окружности
  flag_circle=16; // Присваиваем данное значение переменной для дальнейшего ожидания
```

```
  }

   if (flag_circle == 17) // 29"
  {
   w_length=2.31; // Присваиваем значение переменной служащей для определения скорости
   myNextion.setComponentText("t22", String(29)); // Выводим на дисплей значение выбранной на данный момент длины окружности
   flag_circle=18; // Присваиваем данное значение переменной для дальнейшего ожидания
  }
// Данная часть кода служит для сохранения значения длины окружности колеса
  if (message == "65 3 8 1 ffff ffff ffff") // если условие верно, то...
  {
   EEPROM.write(1,(float)w_length*100.0); // Записать в ячейку памяти значение длины окружности
   w_length=(float)EEPROM.read(1)/100.0; // Читаем ячейку памяти значения длиныокружности
   flag_circle=0; // Присваиваем значение переменной для отключения выбора после сохранения
   myNextion.setComponentText("t22", "Ok"); // Выводим на дисплей сообщение о том, что настройка длины окружности успешно сохранена
  }

//------------------------Цикл сброса пробега-----------------------
 if (message == "65 3 a 1 ffff ffff ffff") // Если условие верно, то...
  {
   reset=1; // Присваиваем значение переменной 1 для условия сброса (подтверждение)
  }
 if (message == "65 3 9 1 ffff ffff ffff") // Если условие верно, то...
  {
   reset_2=1; // Присваиваем значение переменной 1 для условия сброса (полный пробег)
  }
if (message == "65 3 16 1 ffff ffff ffff") // Если условие верно, то...
  {
   reset_3=1; // Присваиваем значение переменной 1 для условия сброса (суточный пробег)
```

```
  }
if (reset == 1 && reset_2 == 1) // Если условие верно, то...
 {
  DIST=0; // Присваиваем значение переменной равное 0
  DIST_T=0; // Присваиваем значение переменной равное 0
  dist_past=0; // Присваиваем значение переменной равное 0
  dist_transit=0; // Присваиваем значение переменной равное 0
  dist_comma=0; // Присваиваем значение переменной равное 0
  dist_excess=0; // Присваиваем значение переменной равное 0
  cell_hi=6; // Присваиваем значение переменной равное 0
  cell_low=7; // Присваиваем значение переменной равное 0
  cell_comma=8; // Присваиваем значение переменной равное 0
  dist_con=500; // Присваиваем значение переменной равное 500
  addr=0; // Присваиваем значение переменной равное 0
  delay(1000); // Делаем паузу в 1 секунду для стабильности
  hi = highByte(DIST_T); // Присваиваем переменной старший бит значения пробега
  low = lowByte(DIST_T); // Присваиваем переменной младший байт значения пробега
  EEPROM.write(cell_hi, hi);  // Записываем во внутреннюю ячейку памяти старший байт
  EEPROM.write(cell_low, low); // Записываем во внутреннюю ячейку памяти младший байт
  EEPROM.write(cell_comma,(float)dist_comma*100.0);     // Записываем во внутреннюю ячейку памяти значение суточного пробега после запятой (умножение на 100 нужно для сохранения сотых долей расстояния)
  hi_d = highByte(dist_con); // Присваиваем переменной старший бит значения для условия
  low_d = lowByte(dist_con); // Присваиваем переменной младший байт значения для условия
  EEPROM.write(4, hi_d); // Записываем во внутреннюю ячейку памяти старший байт
  EEPROM.write(5, low_d); // Записываем во внутреннюю ячейку памяти младший байт
  hi_c = highByte(addr); // Присваиваем переменной старший бит значения переключения ячеек памяти
  low_c = lowByte(addr); // Присваиваем переменной младший байт значения переключения ячеек памяти
  EEPROM.write(2,  hi_c); // Записываем во внутреннюю ячейку памяти старший байт
  EEPROM.write(3, low_c); // Записываем во внутреннюю ячейку памяти младший байт
  reset=0; // Присваиваем значение переменной 0 для завершения цикла
```

```
 reset_2=0; // Присваиваем значение переменной 0 для завершения цикла
 }
 if (reset == 1 && reset_3) // Если условие верно, то...
 {
  DIST=0; // Присваиваем значение переменной равное 0
  dist_transit=0; // Присваиваем значение переменной равное 0
  dist_past=0; // Присваиваем значение переменной равное 0
  delay (1000); // Делаем паузу в 1 секунду для стабильности
  myNextion.setComponentText("t3", String(DIST)); // Отправляем на дисплей значение суточного пробега
  reset=0; // Присваиваем значение переменной 0 для завершения цикла
  reset_3=0; // Присваиваем значение переменной 0 для завершения цикла
 }

//-----------------------Цикл измерения напряжения-----------------------
  Vcc = readVcc(); // Считываем точное напряжение с A1, где будет находиться наш вольтметр
  curVoltage = 0.0; // Обнуляем значение curVoltage
  for (i = 0; i < COUNT; i++) // Запускаем цикл for
  {
     curVoltage = curVoltage + analogRead(A_PIN); // Читаем значение с аналогово входа и суммируем
     delay(10); // Делаем паузу 10 мс для стабильности
  }
  curVoltage = curVoltage / COUNT; // Делим полученное значение на константу напряжения питания (5 вольт)
  float v  = (curVoltage * Vcc) / 1024.0; // Получаем измеренное напряжение путём математического вычисления

  if (debug > 1) // Если условие верно, то...
  {
    Serial.print("V = "); // Выводим в последовательный порт надпись
    Serial.print(v); // Выводим в последовательный порт данное значение переменной
  }
// В данном условии мы показываем пилоту что аккумулятор близок к заданному разряду
```

```
  if (v <= cut_notice) // Если условие верно, то...
  {
    analogWrite (led_r, 255); // Установим ШИМ красного кристалла светодиода на выводе 9
    analogWrite (led_g, 125); // Установим ШИМ зелёного кристалла светодиода на выводе
10
    delay (150); // Делаем паузу, 1 секунду
    analogWrite (led_r, 0); // Установим ШИМ красного кристалла светодиода на выводе 9
    analogWrite (led_g, 0); // Установим ШИМ зелёного кристалла светодиода на выводе 10
    delay (150); // Делаем паузу, 1 секунду
  }
// В данном условии мы показываем пилоту что аккумулятор разряжен
  if (v <= cut_off) // Если условие верно, то...
  {
    flag_leave=0; // Присваиваем переменной значение
    analogWrite (led_r, 255); // Установим ШИМ красного кристалла светодиода на выводе 9
    delay (250); // Делаем паузу, 150 мс
    analogWrite (led_r, 0); // Установим ШИМ красного кристалла светодиода на выводе 9
    delay (300); // Делаем паузу, 300 мс
    shift=shift+1; // Инкрементируем значение переменной
  }
// Данный алгоритм отключает питание контроллера после достаточного числа
предупреждений пилота
  if (shift == 10) // Если условие верно, то...
  {
    flag = 1; // Присваиваем значение переменной 1 для выполнения условия выключения
  }

//-----------------------Цикл выключения и управления с кнопки-----------------------
  flag_button = digitalRead (button_off); // Читаем статус кнопки...
  if (flag_button == 1) // Если условие верное, то...
  {
    flag_off=flag_off+1; // Инкрементируем переменную на 1
    delay(500); // Делаем паузу 25 мс
  }
  // При таком значении выключаем велокомпьютер
```

```
  if (flag_off == 5 && flag_button == 1) // Если условие верно, то...
  {
    flag_off = constrain(flag_off, 0, 5); // Ограничим данную переменную, что бы точно выключить и не промахнуться
    digitalWrite (led_g, LOW); // Выключаем зелёный светодиод
    digitalWrite (led_r, HIGH); // Включаем красный светодиод
    flag = 1; // Присваиваем переменной данное значение для выполнения условия
  }
  // При таком значении показываем, что можно включить режим настройки
  if (flag_off == 3) // Если условие верное, то...
  {
    digitalWrite (led_g, HIGH); // Включаем зелёный светодиод
  }
  // При таком значении включаем режим настройки
  if (flag_off >= 3 && flag_button == 0) // Если условие верное, то...
  {
    flag_off=10; // Присваиваем переменной данное значение для выполнения условия
    flag_vk_3=0; // Присваиваем переменной данное значение для выполнения условия
  }
  // При таком значении выключаем режим настройки
  if (flag_off == 13) // Если условие верное, то...
  {
    flag_off = constrain(flag_off, 10, 13); // Ограничим данную переменную, что бы точно выключить и не промахнуться
    digitalWrite (led_g, LOW); // Выключаем зелёный светодиод
    flag_off=0; // Присваиваем переменной данное значение для выполнения условия
    flag_vk_3=1; // Присваиваем переменной данное значение для выполнения условия
  }


//-----------------------Цикл диагностики-----------------------
  if (message == "65 3 b 1 ffff ffff ffff") // Если условие верно, то...
  {
    flag_mc=flag_mc+1;  // Присваиваем переменной данное значение для выполнения условия
```

```
 }

 if (flag_mc == 1) // Если условие верно, то...
 {
  myNextion.setComponentText("t30", "On"); // Отправляем на дисплей текстовую информацию о включении
  flag_mc=2; // Присваиваем переменной данное значение для выполнения условия
 }
 if (flag_mc == 2) // Если условие верно, то...
 {
  digitalWrite (led_g, LOW); // Выключаем зелёный светодиод
  d_2=digitalRead (2); // Считываем состояние цифрового входа
  d_3=digitalRead (3); // Считываем состояние цифрового входа
  if (d_2 == 1) // Если условие верно, то...
  {
   digitalWrite (led_b, LOW); // Подадим логический  ноль на 11 вывод (выключаем питание кристала)
  }
   if (d_3 == 1) // Если условие верно, то...
  {
   digitalWrite (led_bb, LOW); // Подадим логический  ноль на 12 вывод (выключаем питание светодиода)
  }
  if (d_2 == 0) // Если условие верно, то...
  {
   digitalWrite (led_b, HIGH); // Подадим логический  ноль на 11 вывод (выключаем питание кристала)
  }
   if (d_3 == 0) // Если условие верно, то...
  {
   digitalWrite (led_bb, HIGH); // Подадим логический  ноль на 12 вывод (выключаем питание светодиода)
  }
 }
 if (flag_mc == 3) // Если условие верно, то...
```

```
  {
   digitalWrite (led_b, LOW); // Подадим логический  ноль на 11 вывод (выключаем питание кристала)
   digitalWrite (led_bb, LOW); // Подадим логический  ноль на 12 вывод (выключаем питание светодиода)
   digitalWrite (led_g, HIGH); // Включаем зелёный светодиод
   myNextion.setComponentText("t30", "Off"); // Отправляем на дисплей текстовую информацию о выключении
   flag_mc=0; // Присваиваем переменной данное значение для выполнения условия
  }

//------------------------Цикл отправки информации на дисплей-----------------------
  if (flag_vk_2 == 1) // Если условие верно, то...
  {
   myNextion.setComponentText("t1", String(SPEED)); // Отправляем на дисплей значение скорости велосипеда
   myNextion.setComponentText("t2", String(RPM)); // Отправляем на дисплей значение каденса
   myNextion.setComponentText("t3", String(DIST)); // Отправляем на дисплей значение суточного пробега
   flag_vk=0; // Присваиваем переменной данное значение для выполнения условия
   flag_vk_2=0; // Присваиваем переменной данное значение для выполнения условия
   flag_vk_3=1; // Присваиваем переменной данное значение для выполнения условия
  }
  if (flag_vk == 1) // Если условие верно, то...
  {
   myNextion.setComponentText("t1", String(SPEED)); // Отправляем на дисплей значение скорости велосипеда
   myNextion.setComponentText("t2", String(RPM)); // Отправляем на дисплей значение каденса
   myNextion.setComponentText("t3", String(DIST)); // Отправляем на дисплей значение суточного пробега
  }
  if (flag_vk_3 == 1) // Если условие верно, то...
  {
```

```
  myNextion.setComponentText("t11", String(temp)); // Отправляем на дисплей значение температуры
  myNextion.setComponentText("t12", String(wet)); // Отправляем на дисплей значение влажности
  myNextion.setComponentText("t13", String(DIST_T)); // Отправляем на дисплей значение полного пробега
  myNextion.setComponentText("t24", String(v)); // Отправляем на дисплей значение напряжения аккумулятора
  myNextion.setComponentText("t25", String(Vcc)); // Отправляем на дисплей значение напряжения питания велокомпьютерной сети
 }
}
```