СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире микропроцессорные системы, позволяющие автоматизировать различные процессы и управлять ими, очень широко применяются в разных сферах деятельности. Такое развитие связано с их массовым распространением и усовершенствованием технологий.

Изобретение микропроцессоров привело к тому, что люди стали создавать автоматические устройства, которые сами выполняют некоторые операции по заложенной в них программе, – роботов. На заводах и предприятиях появились промышленные манипуляторы, выполняющие запрограммированный набор действий, который позволяет сократить время производственного цикла. Но большинство роботов имеют большие габариты и низкую манёвренность. Поэтому использование самобалансирующих роботов, передвигающихся на двух колёсах, а также имеющих более высокую подвижность, позволит обеспечить наиболее качественное выполнение всевозможных задач в различных видах деятельности.

Актуальность выпускной квалификационной работы обусловлена тем, что в наше время вопрос балансировки неуравновешенных предметов является одним из наиболее важных в дисциплинах, базирующихся на теории автоматического управления. Для инженеров, занимающихся в этой области, важны как теоретические знания, описывающие идеальную балансирующую модель, так и практические, которые позволят наглядно продемонстрировать принцип стабилизации на реальном объекте. Выпускная работа послужит примером, опираясь на который можно будет решать дальнейшие задачи, связанные с самобалансировкой. Также важно, что данный прототип, обладающий манёвренностью, можно будет использовать в качестве демонстрационного варианта, чтобы понять логику для сборки более мощного сегвея или гироборда. Так как он даёт представление о системе управления и балансировки. Практическая значимость заключается в том, что данная работа может применяться на предприятиях, производящих

бюджетные гироскутеры, а также её можно использовать в будущих проектах, связанных с вопросом стабилизации.

Объект исследования – Роботы на базе микроконтроллеров.

Предмет исследования – Самобалансирующий робот с микроконтроллерным управлением.

Цель выпускной квалификационной работы – проектирование самобалансирующего робота на базе микроконтроллера.

Задачи, которые необходимо решить:

- выполнить анализ предметной области исследования;
- разработать требования к проекту и выбрать элементную базу устройства;
- разработать аппаратную часть и листинг программы;
- рассмотреть вопросы экологической безопасности и охраны труда;
- произвести расчёты себестоимости и эффективности проекта.

1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 Анализ предметной области исследования

1.1.1 Микроконтроллеры и их семейства

Микроконтроллер – это микросхема, применяемая для организации контроля и управления электронными системами. Первый микроконтроллер i8048, который представлен на рисунке 1, был создан в 1976 году американской фирмой Intel.

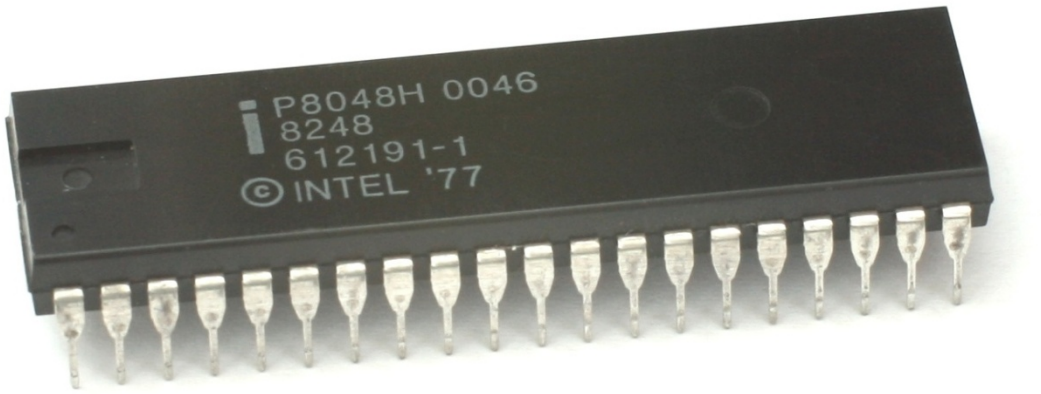


Рисунок 1 – Микроконтроллер i8048

Микроконтроллер состоит из микропроцессора, постоянного запоминающего устройства, а также дополнительных вспомогательных модулей, например портов ввода-вывода, аналогово-цифровых преобразователей [1]. Внешним видом он похож на обычную микросхему. Изготовление микроконтроллеров каждый год превосходит выпуск процессоров, а спрос на них не уменьшается.

Низкое энергопотребление и малые размеры микроконтроллеров позволяют собирать на их основе многие устройства, такие как цифровые термометры, метеостанции, микроволновые печи, различные роботы, системы «умный дом».

Принцип работы микроконтроллера заключается в принятии двух сигналов разного уровня: высокого и низкого. Получая данные сигналы, в памяти устройства происходит запись определённого кода, соответствующего выполняемой команде. Микроконтроллер, имеющий в себе встроенные базовые команды, обрабатывает поступающий код, тем самым выполняя его. Связав необходимые команды друг с другом, получится написать алгоритм работы какой-либо программы [2].

Плюс микроконтроллера состоит в том, что во время его функционирования команды, приходящие из запоминающего устройства либо порта ввода, обрабатываются и сразу же выполняются, следовательно написание программы менее трудоёмко. Также можно легко изменить программный код, загрузив необходимую прошивку.

Микроконтроллеры обычно не функционируют по отдельности. Они взаимосвязано подключаются со всеми датчиками в одну схему.

Микроконтроллеры объединяются в семейства. В каждое семейство входят устройства, имеющие одинаковую базу, например набор команд, режимы прерываний, структуру памяти, входящий набор периферии [3]. Семейства могут классифицироваться по разрядности шины данных используемого процессора: 4-битные, 8-битные, 16-битные и 32-битные. Микроконтроллеры, относящиеся к одному семейству, могут иметь разные функциональные характеристики.

Существуют разные семейства микроконтроллеров, например ARM, AVR, PIC, ESP32, каждый из которых обладает разными функциональными возможностями.

Например, микроконтроллеры семейства ARM потребляют мало электроэнергии, поэтому применяются в изготовлении энергоэффективных устройств.

AVR (Atmel) контроллеры обладают единой архитектурой. В них применяется большой набор развитых команд. Производительность

микроконтроллеров этого семейства высокая, так как при выполнении первой команды сразу начинается подготовка ко второй [4].

У семейства PIC применяется минимальный набор команд. К этому семейству относится большое количество микроконтроллеров с разной разрядностью, которые имеют свою программную среду. Они обладают низким энергопотреблением и имеют всевозможную периферию.

На сегодняшний момент микроконтроллеры ESP32, разрабатываемые компанией Espressif, набирают популярность и используются в разных странах благодаря своей дешевизне и функциональности. ESP32 – серия микроконтроллеров, представленная в 2016 году. Преимуществом ESP32 перед другими микроконтроллерами является наличие встроенных Bluetooth и Wi-Fi модулей, а также развитой периферии.

В независимости от принадлежности к какому-либо семейству все микроконтроллерные устройства программируются. Для написания программ используются разные языки программирования, например Ассемблер, Си, а также Си-подобные языки. Код можно писать в любых текстовых редакторах, но это не эффективно, так как определять ошибки придётся самому. Поэтому наиболее часто используются редакторы кода, включающие в себя разнообразные библиотеки и функциональные возможности, а также компиляторы – программы, позволяющие переводить текст, написанный на формальном языке, предназначенном для программирования, в набор системных команд, понятных определённым устройствам [5]. Соответственно операция, заключающаяся в конвертировании кода, называется компиляцией. Принцип работы компилятора продемонстрирован на рисунке 2.

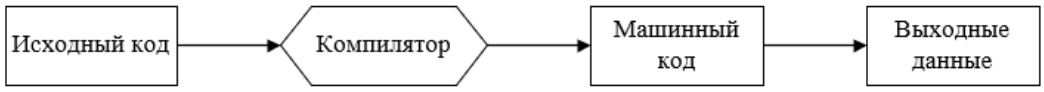


Рисунок 2 – Процесс компиляции

Существует большое количество программ, как бесплатных, так и платных, которые подходят для написания программного кода. К первым можно отнести ArduinoIDE, MPLAB, Atmel Studio, ко вторым – Atollic TrueSTUDIO и PIC Simulator IDE. Все эти среды разработки позволяют выполнять компиляцию, отладку – нахождение ошибок, а также загрузку программы в микроконтроллер. Они предназначены для некоторых семейств микроконтроллеров, например MPLAB используется только для серии PIC. Однако в ArduinoIDE есть возможность работы с контроллерами разных производителей.

1.1.2 Понятие роботов, их разновидности и применение

Роботы для получения сведений об окружающем мире используют различные датчики, которые являются необходимой частью их функционирования. По форме и конструкции данные автоматические устройства могут быть любыми, но зачастую при их изготовлении применяют части тел каких-либо живых существ. Роботы способны заменить трудовую деятельность человека, выполняя всевозможные операции – от прикладных до производственных. Они могут управляться как оператором, так и выполнять задачи автоматически, по заданному алгоритму.

Прототипом роботов были играющие на музыкальных инструментах механические фигуры, изготовленные изобретателем Аль-Джазари, жившем в 12 веке нашей эры. Первые зарисовки роботизированного человека были выполнены Леонардо да Винчи в XV веке. У этого деятеля также были обнаружены подробные рисунки механического воина, который мог сидеть и шевелиться, во второй половине XX века. В это же время в Америке фирмы по производству робототехники выпустили и стали продавать первых промышленных роботов, которые были нужны для выполнения технологических операций [6].

Производство роботов начало расти с появлением систем управления на основе микропроцессоров и языка программирования, разработанного для робототехнических устройств в 1982 году. Первый робот Scara,

потребляющий электроэнергию, был изготовлен в 1984 году [7]. В 1986 году роботы были применены для ликвидации аварии на Чернобыльской АЭС. Одним из них был «Клин-1», который выполнял большой объём работ [8].

Наибольшее развитие робототехники произошло в XXI веке. Эта наука стала использоваться в разных целях человека. Применение ей стали находить в быту и даже космосе.

На протяжении всей истории робототехники можно отметить 3 этапа её развития:

1. создание первых версий роботов, которые использовались в 1960-х в промышленности. У них была определённая программа, которая не учитывала факторы внешней среды;
2. изобретение робототехнических систем 2-го поколения. С этого времени для них разрабатывались программы, которые могли с необходимой точностью решать различные задачи. Также был выбор нужного режима работы в зависимости от факторов окружающей среды;
3. производство интеллектуальных роботов, в которых используются технологии распознания образов, а также адаптации к необходимым условиям. На этом этапе начинается освоение искусственного интеллекта.

В наш век разрабатывается много разновидностей роботов, которые применяются для решения различных задач и операций. По назначению их можно классифицировать на следующие виды:

- промышленные, которые применяются на производстве;
- медицинские. Некоторые из них способны ухаживать за больными, а другие – приготавливать лекарства;
- бытовые, используемые для решения проблем человека в ежедневных делах;
- боевые, которые применяются в военных действиях;

– исследовательские, предназначенные для проведения исследований в труднодоступных и опасных местах для человека.

К преимуществам роботов можно отнести то, что они не утомляются, им не нужно время на отдых и пищевые ресурсы. Обеспечив автоматическим устройствам необходимое энергопотребление, их можно успешно применять на производстве. Например, они активно используются на заводах, специализирующихся на изготовлении автомобилей. На данный момент у предприятий автомобильной промышленности есть концепции развития, заключающиеся в автоматизации сборки и производства деталей с помощью роботов [9]. Стоит отметить, что спрос на роботов ежегодно только увеличивается. Денежные средства, потраченные на закупку необходимой техники, быстро окупаются, так как растёт качество, и увеличивается темп выпуска продукции.

По методу передвижения роботы подразделяются на следующие группы:

– использующие колёса, количество которых может варьироваться;
– с гусеничным передвижением, которое обычно используется в военной технике;
– имитирующие движение человека посредством шагов с помощью двух ног;
– летающие, например дроны;
– плавающие, способные передвигаться в воде как рыбы;
– имитирующие походку животных, например кошки или собаки.

В роботах также могут комбинироваться несколько способов, например они могут обладать одновременно колёсами и гусеницами.

Все робототехнические устройства имеют какую-либо систему управления. В зависимости от того, принимает ли участие человек в их контроле, они делятся на 3 вида [10]:

1. биотехнические;

2. интерактивные;
3. автоматические.

Биотехнические роботы выполняют действия, когда с ними взаимодействует оператор. Он может управлять ими как с помощью различных пультов дистанционного управления, так и на основе своих движений, которые они будут повторять.

Интерактивные роботы содержат в себе устройства памяти, в которые записываются определённые команды для автоматической реализации некоторых операций. Такие устройства могут управляться по очереди, как оператором, так и автоматически.

Автоматические роботы функционируют самостоятельно, выполняя заранее заданный программный алгоритм либо приспосабливаясь под определённые условия для решения необходимых задач.

Несмотря на тип управления, роботы по своему внешнему виду и способу выполнения определённых функций делятся на следующие группы [11]:

- декартовы роботы. Данные устройства служат для работы в трёхмерном пространстве (прямоугольной системе координат Декарта). У них есть 3 оси – x, y и z, по которым они двигаются и выполняют необходимые операции с высокой точностью;
- цилиндрические роботы. Они имеют 3 оси – круговую и 2 линейные, позволяющие осуществлять движения рукой устройства в вертикальном и горизонтальном положении. Такие роботы применяются при сварке деталей или их транспортировке;
- параллельные роботы. Это устройства, которые поддерживают какую-либо платформу с помощью применения управляемых компьютером механизмов. Данные системы используются в лётных и автомобильных тренажёрах;
- шарнирные роботы. Они состоят из нескольких элементов, последний из которых выполняет необходимые подвижные задачи, как по

основным осям, так и имеющимся в любом количестве дополнительным. Благодаря этому достигается высокая манёвренность, позволяющая использовать эти устройства во многих областях, например в качестве погрузчика или дозатора;

- сферические роботы. Это устройства с внутренним приводным блоком, внешней оболочкой которых выступает шар. Они обладают мобильностью, а также способны передвигаться под водой. Их могут использовать в наблюдении за разными процессами;
- роботы-гуманоиды. Данные устройства спроектированы в виде человека. Они имитируют его движение, а в некоторых случаях даже речь. В таких роботах применяются различные датчики вместо органов чувств человека, считывающие необходимые показания;
- роботы-гексаподы, использующие для передвижения 6 ног, прикреплённых к корпусу. Внешне они похожи на паукообразных. Их ноги позволяют обеспечить нужную устойчивость и гибкость;
- трёхногие и четвероногие роботы. Данный класс устройств основан на реализации походки с помощью трёх или четырёх ног. Такие роботы могут копировать движение собак или кошек;
- одноколёсные и двухколёсные роботы. Создание робота, способного передвигаться и балансировать на одном колесе, является сложной задачей для робототехников. Инженерам необходимо учитывать много различных факторов, чтобы устройство удерживало баланс. Поэтому чаще разрабатываются двухколёсные роботы. В них принцип балансировки достигается легче. Устанавливаются с разных сторон 2 двигателя с колёсами, которые всегда должны находиться в активном состоянии для сохранения баланса, а также датчики-гироскопы, которые предназначаются для определения пространственных углов тела;
- плавательные роботы. Эти устройства могут принимать внешность рыб, водных змей, и предназначены для плавания в воде;

– летающие роботы. Полёт в них обеспечивается за счёт применения различных двигателей и приводов.

1.1.3 Понятие самобалансирующего устройства

В современном мире всё большую популярность набирают самобалансирующие устройства. Гироскутеры относятся к разновидностям данных устройств. Это электрические транспортные средства, изготовленные в виде двух соединённых платформ, на которые встаёт человек, с колёсами, находящимися с двух сторон. Они осуществляют самобалансировку с помощью электродвигателей, которые получают питание от электроаккумуляторов, и датчиков гироскопов. Данное изобретение берёт начало в 1990-е годы, когда зарождались подобные механизмы, имеющие в своём строении системы автоматической балансировки. Балансировка – это процесс уравновешивания каких-либо вращающихся тел или частей, например колёс автомобиля. Смысл балансировки заключается в нахождении углов отклонения и установки регулирующих масс для стабилизации ротора [12]. Понятие «самобалансирующее устройство» означает процесс нахождения равновесного состояния самим устройством с помощью использования датчиков гироскопов, способных подстраиваться под изменение углов находящегося в пространстве тела, на котором они установлены, и микроконтроллера либо микропроцессора, которые обрабатывают сигналы с датчиков и подают на двигатели управляющие импульсы, приводящие их в движение. Двигатели, в свою очередь, реагируя на изменение равновесия устройства, придают вращение каждому колесу или другому механизму, который может использоваться, тем самым обеспечивая балансировку как при нахождении в одном положении (на месте), так и в движении.

1.1.4 Достижение балансировки на программном уровне

На программном уровне реализация стабильного положения проектируемого устройства будет достигаться за счёт компенсации отклонения углов.

Гироскоп на основе данных об угловой скорости позволяет найти положение, в котором он окажется спустя какое-то время. Но из-за ошибок вычислений скорости, а также влияния внешних сил на устройство, появляется не соответствующий действительности результат в измерении пространственных углов. Поэтому гироскопы не дают возможности точно определить положение тела, которое нужно для стабилизации. Для обеспечения наиболее приближённых значений в ориентации к ним применяют различные фильтры, такие как Калмана или Маджвика.

В разрабатываемом прототипе будет использоваться последний вариант. Это наиболее новый метод фильтрации сигналов, который позволяет достичь удержания баланса с меньшим применением вычислительных ресурсов [13]. Он представлен двумя способами – AHRS (Attitude and Heading Reference Systems) и MARG (Magnetic, Angular Rate and Gravity). В первом используются только показания акселерометра – устройства, служащего для определения проекции ускорения, и гироскопа, во втором к ним добавляются значения магнетометра – прибора, предназначенного для считывания значений магнитного поля. Расчёт ориентации в фильтре Маджвика происходит за счёт применения кватернионов – совокупности гиперкомплексных чисел, которые нужны для получения значений положения тела в трёхмерном пространстве.

Для примера взяты 2 системы координат – A и B, которые представлены на рисунке 3. Чтобы определить положение системы B к A, нужно повернуть на нулевой угол систему A вокруг вектора $A_{\hat{r}}$.

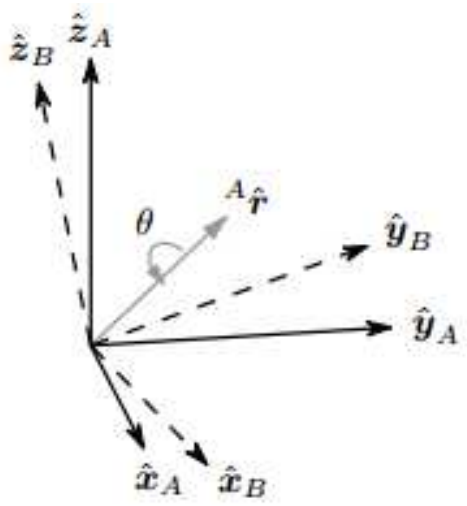


Рисунок 3 – Системы координат

Сделать это можно с помощью кватерниона, применяющегося в фильтре и описывающего положение системы координат B к A (1) [13]:

$$ {}^A_B\hat{q} = [q_1\ q_2\ q_3\ q_4] = \left[\cos\frac{0}{2} - r_x \sin\frac{0}{2} - r_y \sin\frac{0}{2} - r_z \sin\frac{0}{2}\right], \quad (1) $$

где q_1 – нормированный кватернион, равный единице;

$q_2\ q_3\ q_4$ – кватернионы координатных осей;

r_x, r_y, r_z – вектора в координатной плоскости.

Чтобы получить значение ориентации тела, находящегося в трёх координатных осях, нужно перевести гиперкомплексные числа в углы Эйлера, продемонстрированные на рисунке 4 и имеющие обозначения α, β, γ или φ, θ, ψ. Это можно сделать с помощью формул (2), (3), (4) [13]:

$$ \psi = \operatorname{Atan2}(2q_2\ q_3 - 2q_1 q_4, 2q_1^2 + 2q_2^2 - 1); \quad (2) $$

$$ \theta = -\sin^{-1}(2q_2 q_4 + 2q_1 q_3); \quad (3) $$

$$ \varphi = \operatorname{Atan2}(2q_3 q_4 - 2q_1 q_2, 2q_1^2 + 2q_4^2 - 1). \quad (4) $$

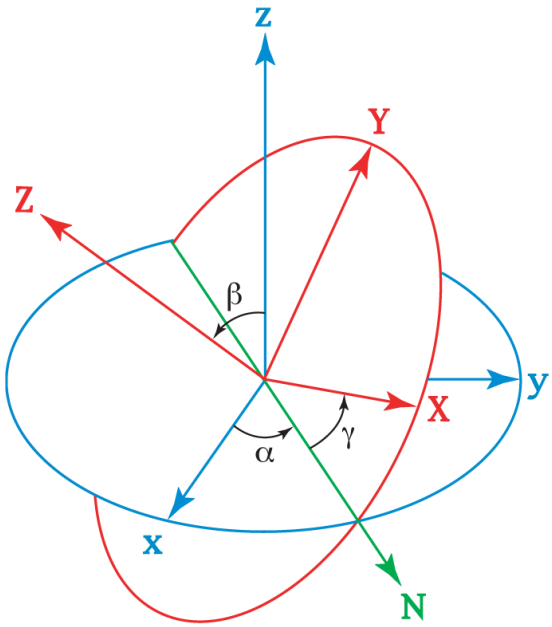


Рисунок 4 – Углы Эйлера

Таким образом, формулы, реализующие Фильтр Маджвика, будут применяться при написании программного кода к устройству.

1.2 Сравнительный обзор программно-аппаратных комплексов данной системы

Самобалансирующий робот – это робот, способный балансировать на месте и при езде благодаря использованию электродвигателей и гироскопических датчиков.

На рынке представлены различные варианты реализации данных роботов. Были рассмотрены некоторые из них, их цена, вид и характеристики.

Yahboom Balance Robot – это робот, балансирующий на двух колёсах и способный двигаться по любой поверхности. На нём есть специальная площадка, на которую можно поставить какой-либо предмет.

Устройство благодаря конструкции шасси способно совершать разворот и маневрировать. Оно основано на микроконтроллере ATmega328p. Конструктор содержит следующие детали: два мотора-редуктора с их драйверами, плату расширения, колёса, датчик-гироскоп MPU6050, Bluetooth-модуль. При необходимости можно перепрограммировать робота.

Чтобы подключиться к устройству, нужно установить на смартфон специальное мобильное приложение. Питание робот получает от трёх литий-ионных аккумуляторов 18650 с напряжением по 3,7 В.. На платформу устройства, корпус которого выполнен из органического стекла, можно класть груз массой до 2 кг. Робот может работать до 8 часов, сохраняя при этом все свои функции. Размеры данного самобалансирующего устройства – 220х72х142 мм, а вес составляет 900 г. Его цена – 9240 рублей[1].

Внешний вид рассматриваемого робота представлен на рисунке 5.

Рисунок 5 – Yahboom Balance Robot

KiB RC Robot – это балансирующий робот, изготовленный из пластика. Он собран на основе контроллера TPS2231PWRG4 с гироскопом и сенсорными датчиками. С помощью двух колёс устройство может ездить и перевозить грузы. Запуск робота осуществляется с пульта на радиоуправлении, которым можно взаимодействовать на дистанции.

Особенностью данного самобалансирующего робота является его способность двигаться по заданному алгоритму и проводить спарринг по боксу со вторым таким же устройством. KiB RC Robot питается от 4 батареек формата AAA с напряжением равным 1,5 вольт, а пульт управления от 2 таких же элементов питания. Аккумуляторы в комплекте с роботом не идут.

[1] Балансирующий робот Yahboom Balance Robot [Электронный ресурс] URL: https://supereyes.ru/catalog/konstruktory_robot_kit_rrobototehnicheskie_nabory/balansiruyushchiy_robot_arduino_balance_robot/ (дата обращения: 22.02.2022)

Устройство может бесперебойно функционировать в течение 2 часов. Робот весит 750 г, а его габариты равны 210x80x150 мм. Цена устройства – 5157 рублей[2].

Данный самобалансирующий робот продемонстрирован на рисунке 6.

Рисунок 6 – KiB RC Robot

AT-Smart Robot – это интерактивный робот с колёсной платформой. В нём применяется контроллер LPC1768. Одна часть данного устройства выполнена из АБС-пластика, а другая из металла. Робот может выполнять следующие команды:

- определять в какую сторону ехать;
- двигаться по заданному алгоритму;
- записывать и повторять сообщения.

Устройство также реагирует на касание и голос человека. У него есть инфракрасные датчики, с помощью которых он выполняет свои функции. В роботе присутствует возможность записи голосовых сообщений. В нём можно выбрать 12 мелодий со световыми эффектами, так как есть

[2] Радиоуправляемый робот JXD KiB [Электронный ресурс] URL: https://rc-like.ru/radioupravlyaemyj-robot-jxd-kib-2-4g-1016a (дата обращения: 22.02.2022)

кнопки, которые позволяют выбрать необходимое действие. Громкость музыки регулируется с помощью сенсоров. Благодаря режиму автоматического программирования в AT-Smart Robot можно задать разные направления движения. Устройство способно автономно работать около часа. Питание в нём осуществляется с помощью 3 гальванических элементов типа AAA с напряжением 1,5 вольт, которые не входят в комплект. Вся конструкция весит 458 г. Размеры робота – 140x160x200 мм. Его цена – 4550 рублей[3].

Обозреваемое устройство можно увидеть на рисунке 7.

Рисунок 7 – AT-Smart Robot

B-Robot – это самобалансирующийся робот на микроконтроллере ATmega32u4, созданный из частей, изготовленных на 3D-принтере. Программный код устройства разработан в среде Arduino IDE и его можно менять. Имея всего два колеса, робот может постоянно поддерживать равновесие, используя свои внутренние датчики и приводя в движение двигатели. Управлять устройством можно, отправляя команды по технологии Wi-Fi с мобильного устройства.

[3] Интерактивный AT Smart Robot [Электронный ресурс] URL: https://smartselect.kz/product/interaktivnyj-at-smart-robot-russkij-yazyk/ (дата обращения: 22.02.2022)

Этот робот считывает данные с датчиков, встроенных в микросхему MPU6000, со скоростью 200 раз в секунду. Он вычисляет угол своего положения и сравнивает его с базовым углом, равным 0º. На основе различия угловых значений, B-Robot отправляет необходимые команды двигателям, чтобы сохранить свой баланс. Его корпус имеет следующие размеры – 150x100x50 мм и весит 300 г. Цена робота – 7992 рубля[4].

Реализация этого самобалансирующего робота представлена на рисунке 8.

Рисунок 8 – B-Robot

Самобалансирующих роботов на рынке представлено не так много, но из тех вариантов, которые есть, многофункционального и относительного дешёвого устройства – нет.

1.3 Формирование требований к проектируемому устройству

Самобалансирующий робот должен ездить в любую сторону и останавливаться. Для этого он должен быть связан с мобильным устройством через приложение Bluetooth RC Car, интерфейс которого представлен на рисунке 9, с помощью Bluetooth модуля.

[4] B-robot EVO KIT [Электронный ресурс] URL: https://www.jjrobots.com/product/b-robot-kit-plug-and-play-robot-version/ (дата обращения: 22.02.2022)

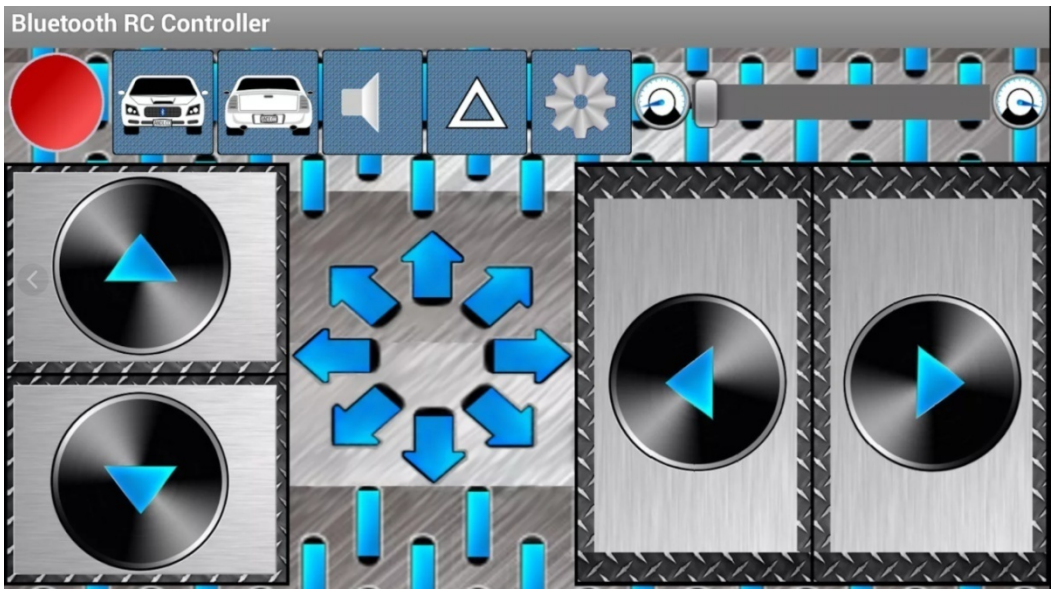


Рисунок 9 – Интерфейс мобильного приложения

При нажатии кнопок в приложении сигнал будет передаваться устройству, которое обеспечивает выполнение заданных команд. Если робот будет находить впереди себя препятствие в виде стены, то он развернётся. Для защиты от падений в устройстве должна быть дуга, на которую оно будет опираться. Самобалансирующий робот с помощью своей подвижной руки должен подниматься с поверхности. Также с помощью кнопок проектируемое устройство сможет менять свою скорость. Все датчики, позволяющие выполнять команды, в системе должны соединяться с одним микроконтроллером.

Устройство должно иметь 2 режима работы:

1. основной, в котором система поддерживает режим работы и выполняет все свои функции;
2. выключенный, в котором система отключена.

В проектируемом устройстве будут представлены следующие функциональные блоки:

– блок измерения угла наклона и балансировки;
– блок определения расстояния до преграды;
– блок питания;
– блок связи с устройством.

Данный проект должен выполнять следующие функции:

- самобалансировать на месте и при езде с помощью гироскопа и колёс;
- измерять расстояние до препятствия с использованием ультразвукового датчика, чтобы развернуться;
- подключаться по Bluetooth к приложению на смартфоне для управления.

К устройству предъявляется ряд требований для обеспечения безопасности:

- включенное устройство должно находиться в защищённом от влаги месте;
- выполнять разборку и проверку соединения контактов на плате только при отключённом питании.

Требования к хранению и эксплуатации самобалансирующего робота заключаются в следующем. Перед хранением из устройства рекомендуется достать элементы питания, чтобы обеспечить им сохранение большего уровня заряда. Робот должен храниться и эксплуатироваться в помещении с температурой в установленных пределах – от 0 до +35 °С. При этом он не должен располагаться близко к системам отопления. Также для обеспечения работоспособного состояния устройства относительная влажность места хранения должна быть больше 35%, но ниже 80%. Самобалансирующий робот предназначается для использования на ровной поверхности.

1.4 Выводы по исследовательскому разделу

В выпускной квалификационной работе выбран вариант разработки самобалансирующего двухколёсного робота. Данный вид устройств обеспечивает самобалансировку как за счёт своей аппаратной части, так и программной. Математическая модель этих роботов представлена обратным маятником – механизмом с центром тяжести, располагающимся выше своего основания. Он всегда неуравновешен и должен сохранять баланс, чтобы не упасть.

2 КОНСТРУКТОРСКИЙ РАЗДЕЛ

2.1 Анализ требований к проектируемому устройству

При разработке технического задания самобалансирующего робота (Приложение А) были предъявлены требования к надёжности, к эргономике и технической эстетике, к эксплуатации, к безопасности, к функциям и к информационному обеспечению. Все требования, приведённые в техническом задании, подобраны для достижения максимального удобства пользования устройством за небольшую сумму, а также обеспечения надёжной работы в процессе эксплуатации.

Самобалансирующий робот будет состоять из 4 подсистем:

1. подсистема измерения расстояния;
2. подсистема измерения угла наклона и балансировки;
3. подсистема питания;
4. подсистема Bluetooth-соединения.

Измерение расстояния у робота будет происходить с помощью ультразвукового датчика, который отправляет сигналы. При нахождении преграды они отражаются и возвращаются, тем самым давая устройству определённые команды, для предотвращения столкновений.

В подсистему измерения угла наклона и балансировки будут входить гироскоп со встроенным акселерометром и шаговые моторы с их драйверами. На основании показаний первых двух датчиков будет происходить управление шаговыми моторами, которые позволят роботу балансировать.

Питание устройства будет осуществляться от четырёх литий-ионных аккумуляторов, которые обеспечат микроконтроллеру, а значит и всем датчикам, подключённым к нему, необходимое напряжение. Некоторым датчикам будет необходимо меньшее напряжение, чем другим. Для понижения напряжения от батарей будут использоваться понижающие DC-

DC преобразователи. Также будут задействованы конденсаторы для сглаживания пульсаций тока.

Подсистема Bluetooth-соединения будет основана на микроконтроллере, содержащем в себе Bluetooth-модуль, и смартфоне, на котором установлено приложение для связи с роботом. Данный способ взаимодействия удобен, так как он позволяет удалённо управлять устройством без дополнительного оборудования.

Требования к технической эстетике будут реализованы за счёт удобного, компактного размещения всех разъёмов и остальных элементов на плате устройства, а также наличия в мобильном приложении интуитивно понятного интерфейса и обмена команд, основанного на Bluetooth-модуле.

2.2 Выбор элементной базы устройства

Для разработки самобалансирующего робота необходимо подобрать элементную базу, которая будет сочетать в себе следующие свойства: материальную доступность, относительную лёгкость применения, соответствующие проектированию характеристики и универсальность.

Для управляющей программы всех датчиков потребуется микроконтроллер с поддержкой Bluetooth, имеющий не менее 15 линий ввода/вывода.

Микроконтроллер будет выбран из вариантов, указанных в таблице 1.

Таблица 1 – Сравнение микроконтроллеров

Критерии\Микроконтроллер	AVR (AVR32DA28-I/SO)	PIC (PIC16F628A)	ESP32 (ESP32-D0WDQ6)
1. Напряжение питания	1,8-5,5 В.	3-5,5 В.	2,3-3,6 В.
2. Разрядность данных	8	8	32
3. Частота процессорного ядра	24 МГц	20 МГц	240 МГц
4. Объём ОЗУ	4 Кб	1,75 Кб	520 Кб
5. Кол-во линий I/O	28	16	48
6. Цена	260 руб.	100 руб.	250 руб.

Сравнивая данные микроконтроллеры, можно прийти к выводу, что ESP32-D0WDQ6, который представлен на рисунке 10, при своей низкой

стоимости обладает развитым функционалом и превосходит свои аналоги. Его структурная схема продемонстрирована на рисунке 11.

Рисунок 10 – Микроконтроллер ESP32-D0WDQ6

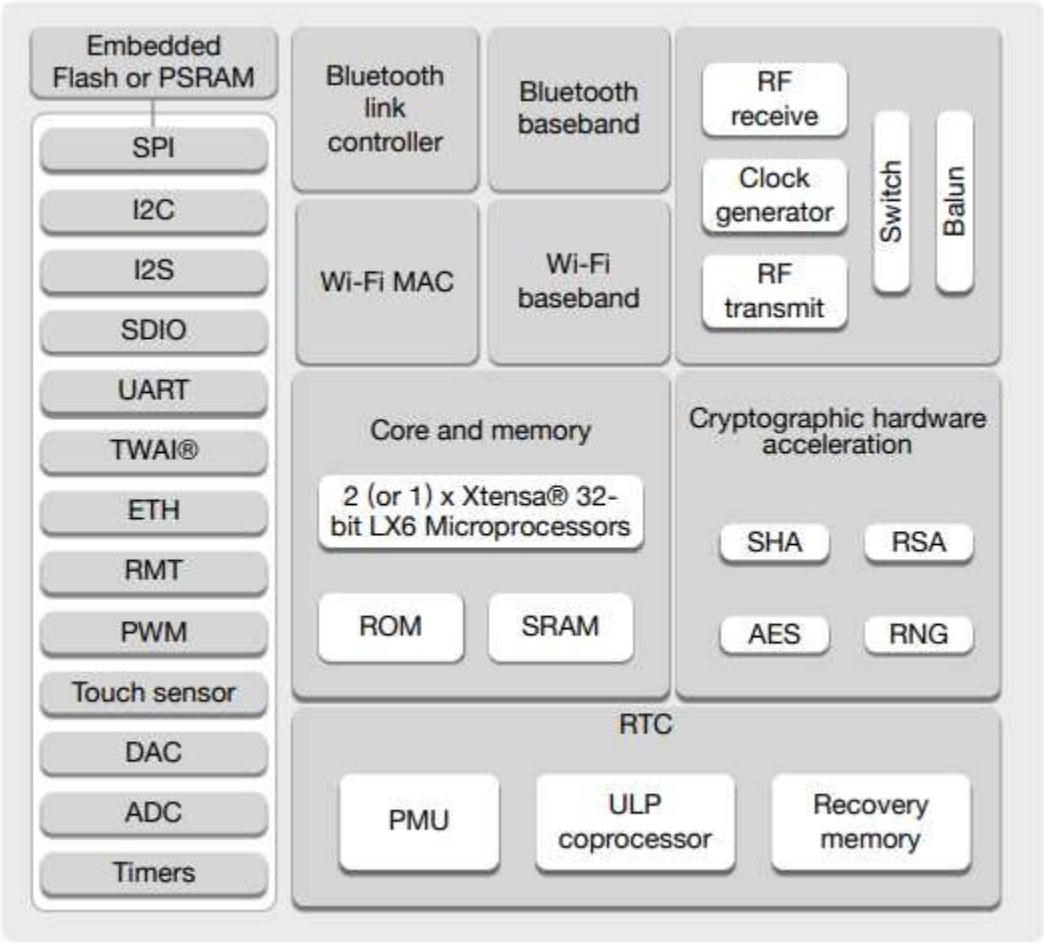


Рисунок 11 – Структурная схема используемого микроконтроллера

Этот микроконтроллер из семейства ESP32 содержит в себе следующие функциональные возможности и параметры[5]:

- процессорная часть основана на двухъядерном 32-разрядном Tensilica Xtensa LX6;
- объём памяти ПЗУ – 448 Кб (для загрузчика и функций ядра), внутренний статической памяти – 520 Кб (для данных и команд);
- беспроводные интерфейсы, такие как Wi-Fi и Bluetooth;
- часы и таймеры:
  1) внутренний генератор 8 мГц;
  2) внутренний RC-генератор;
  3) внешний генератор 40 мГц;
  4) внешний генератор 32 кГц для RTC (часы реального времени);
  5) 2 группы таймеров, включая 2x64 бит таймеры;
  6) RTC таймер;
- периферийные интерфейсы:
  1) 34 программируемых универсальных портов ввода/вывода;
  2) 12-ти разрядный АЦП, до 18 каналов;
  3) два 8-битных ЦАП;
  4) 10 портов для подключения емкостных датчиков;
  5) 4 последовательных интерфейсов SPI;
  6) 2 интерфейса I^2S;
  7) 2 интерфейса I^2C;
  8) 3 интерфейса передачи данных UART;
  9) Ethernet MAC интерфейс;
  10) ИК-порт;

[5] ESP32 Series Datasheet [Электронный ресурс] URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата обращения 24.02.2022)

11) ШИМ для двигателей и для светодиодов до 16-ти каналов.

Также есть поддержка работы с внешней флэш-памятью.

Рабочее напряжение питания чипа обычно 3,3 В. Оно подаётся на контакты. Средний потребляемый ток – 80 мА.

Почему же для устройства был выбран именно микроконтроллер ESP32-D0WDQ6? Микроконтроллеры семейства ESP32 со своей достаточно развитой периферией обладают низкой стоимостью, а также кроссплатформенностью. Выбранный вариант имеет все характеристики, а именно объём памяти, количество контактов, нужные для разработки проекта, а также модуль Bluetooth, благодаря которому нет необходимости в отдельном датчике. Можно было бы использовать микроконтроллер из семейства AVR – AVR32DA28-I/SO, но он не содержит в себе Bluetooth-модуль и уступает в функционале. Также есть несколько параметров, по которым ESP32 превосходит распространенные контроллеры AVR: скорость опроса датчиков, скорость взаимодействия с различными двигателями.

Следовательно, выбранный микроконтроллер подходит для построения самобалансирующего робота.

Устройство ездит с помощью двух колёс. Для управления вращением колёс используются шаговые двигатели, которые обеспечивают плавность управления. Шаговый двигатель – это электродвигатель, на который поступает напряжение, и в результате этого происходит перемещение его вращающейся части на определённый угол шагами. Для разработки устройства необходимо 2 шаговых мотора. На рынке представлено большое разнообразие различных моделей. Характеристики наиболее подходящих из них указаны в таблице 2[6, 7].

[6] 17HS4401S datasheet [Электронный ресурс] URL: https://datasheetspdf.com/pdf-file/1310364/Handson/17HS4401S/1 (дата обращения 28.02.2022)

Таблица 2 – Сравнение шаговых двигателей

Критерии\Шаговый двигатель	Nema 17HS4401	28BYJ-48
Номинальный ток	1,7 А.	1,7 А.
Напряжение питания	12-36 В.	5-24 В.
Крутящий момент	5,5 кг х см.	0,3 кг х см.
Угловой шаг	1,8°±5%, 200 шагов	5,625°, 64 шага
Число фаз	2	4
Разъём	4 PIN	4 PIN
Диаметр вала	5 мм	5 мм
Длина вала	24 мм	8 мм
Габариты	42 х 42 х 48 мм	25 х 18 х 25 мм
Вес	280 г	40 г
Цена	600 руб.	70 руб.

Из данных шаговых моторов по крутящему моменту и количеству шагов подходит Nema 17HS4401, продемонстрированный на рисунке 12. Это распространённый шаговый двигатель, который широко используется в разных проектах за счёт своего точного позиционирования, надёжности, а также доступной цены.

Рисунок 12 – Шаговый двигатель Nema 17HS4401

Для управления Nema 17HS4401 нужны два шаговых драйвера. Это платы, которые принимают входящие импульсы STEP или DIR, имеющие соответственно низкий и высокий уровень постоянного напряжения 5 В. И

[7] 28BYJ-48 Stepper Motor [Электронный ресурс] URL: https://components101.com/sites/default/files/component_datasheet/28byj48-step-motor-datasheet.pdf (дата обращения 28.02.2022)

регулирующие угол вращения вала. В таблице 3 представлено сравнение двух драйверов[8, 9].

Таблица 3 – Сравнение драйверов

Критерии\Драйвер шагового двигателя	A4988	DRV8825
Напряжение питания	8-35 В.	8,2-45 В.
Установка шага	1, 1/2, 1/4, 1/8, 1/16	1, 1/2, 1/4, 1/8, 1/16, 1/32
Напряжение логики	3,3 В.	3-5,5 В.
Защита от перегрева	Есть	Есть
Максимальный ток на фазу	1 А. без радиатора, 2 А. с радиатором	1,5 А. без радиатора, 2,5 А. с радиатором
Габариты	20 х 15 х 10 мм	20 х 15 х 10 мм
Цена	72 руб.	90 руб.

Среди рассмотренных вариантов наиболее подходящим является DRV8825, продемонстрированный на рисунке 13, так как в нём можно наиболее точно настроить шаг.

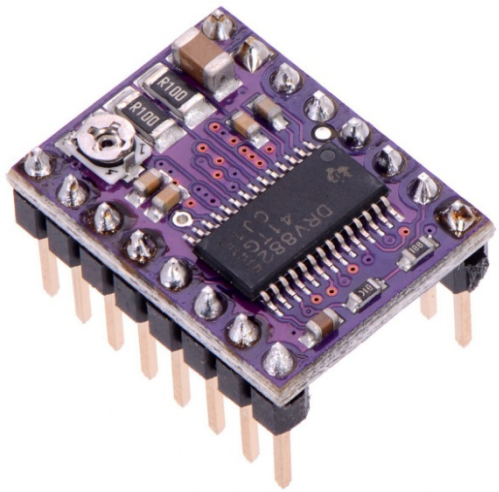


Рисунок 13 – Драйвер двигателя DRV8825

Самобалансирующий робот должен предотвращать столкновения с препятствиями. Ультразвуковой датчик применяется для нахождения различных тел, и определения их расстояния относительно него, что в свою

[8] A4988 Driver [Электронный ресурс] URL: https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf (дата обращения 02.03.2022)

[9] DRV8825 Stepper Motor [Электронный ресурс] URL: https://www.ti.com/lit/ds/symlink/drv8825.pdf?ts=1646807282047 (дата обращения 02.03.2022)

очередь позволяет избежать аварийных ситуаций. В датчике находится излучатель, создающий волну, движущуюся в пространстве со скоростью звука. При попадании на какие-либо тела ультразвуковая волна, отражаясь, приходит к датчику. Он обрабатывает отражённый сигнал и сохраняет время его возвращения. Благодаря этому можно найти расстояние, которое преодолела волна, с помощью микроконтроллера. Сравнение некоторых датчиков представлено в таблице 4[10, 11, 12].

Таблица 4 – Сравнение датчиков ультразвукового излучения

Критерии\Датчик ультразвукового излучения	US-025	HC-SR04	HY-SRF05
Рабочее напряжение	3-5,5 В.	5 В.	5 В.
Рабочий ток	5,3 мА.	15 мА.	20 мА.
Дальность измерения	2-600 см.	2-400 см.	2-450 см.
Точность измерения	0,2 см.	0,3 см.	0,2 см.
Угол измерения	15°	15°	15°
Габариты	45 х 20 х 15 мм	43 х 20 х 15 мм	50 х 30 х 20 мм
Цена	47 руб.	50 руб.	69 руб.

Из данных вариантов наиболее подходящим для проекта является US-025, продемонстрированный на рисунке 14. Он обладает наибольшей дальностью измерения, а также наименьшей ценой.

Рисунок 14 – Датчик ультразвукового излучения US-025

[10] US-025 ультразвуковой датчик расстояния [Электронный ресурс] URL: https://www.chipdip.ru/product0/8009337863 (дата обращения 07.03.2022)
[11] HC-SR04 [Электронный ресурс] URL: https://static.chipdip.ru/lib/092/DOC001092302.pdf (дата обращения 07.03.2022)
[12] HY-SRF05 Datasheet [Электронный ресурс] URL: https://datasheet-pdf.com/PDF/HY-SRF05-Datasheet-ETC-813041 (дата обращения 07.03.2022)

Для балансировки и нахождения углов, характеризующих наклон устройства, в проекте используется гироскоп-акселерометр, выполненный в одном корпусе. Используя данные датчики по одному, не будет возможности точно измерить угол. Поэтому в самобалансирующем роботе они применяются вместе. Характеристики некоторых таких датчиков представлены в таблице 5[13].

Таблица 5 – Сравнение гироскопов-акселерометров

Критерии\Гироскоп-акселерометр	MPU6050	GY-9250
Рабочее напряжение	3,5-6 В.	3-5 В.
Ток потребления	До 4 мА.	До 4 мА.
Диапазон измерений	± 2 ± 4 ± 8 ± 16g	± 2 ± 4 ± 8 ± 16g
Режимы для гироскопа	±250°, ±500°, ±1000°, ±2000°	±250°, ±500°, ±1000°, ±2000°
Количество осей	6 осей (3 оси гироскопа, 3 оси акселерометра)	9 осей (3 оси гироскопа, 3 оси акселерометра, 3 оси магнитометра)
Габариты	20 x 10 x 3 мм	25 x 16 x 3 мм
Цена	83 руб.	464 руб.

Данные устройства отличаются количеством осей и соответственно ценой. MPU6050, имеющий 6 осей, подходит для проекта, так как обладает таким же функционалом, что и GY-9250, и стоит дешевле. Выбранный датчик продемонстрирован на рисунке 15.

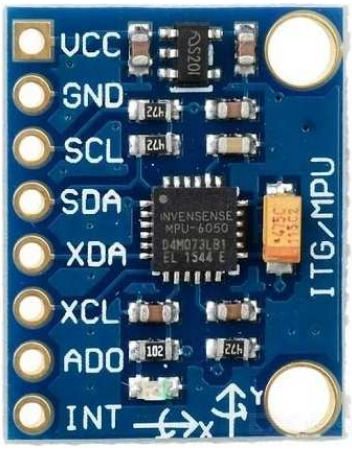


Рисунок 15 – Гироскоп-акселерометр MPU6050

[13] MPU-6000 Datasheet [Электронный ресурс] URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf (дата обращения 11.03.2022)

В самобалансирующем роботе применяется подвижная рука, которая обеспечивает поднятие устройства при падении. Чтобы управлять рукой, нужен сервомотор. Сервомотор – это механизм, который позволяет вращать с заданной точностью какие-либо элементы. Характеристики рассмотренных сервомоторов представлены в таблице 6[14, 15, 16].

Таблица 6 – Сравнение сервомоторов

Критерии\Сервомотор	MTR-SERVO-FS5106M	MG995	MS-6-40
Рабочее напряжение	4,8-6 В.	4,8-7,2 В.	4,8-6 В.
Угол поворота	180°	180°	120°
Скорость вращения	60° за 0,18 сек (при 4,8 В.), 60° за 0,16 сек (при 6 В.)	60° за 0,20 сек (при 4,8 В.), 60° за 0,16 сек (при 6 В.)	60° за 0,18 сек (при 4,8 В.), 60° за 0,16 сек (при 6 В.)
Крутящий момент	6 кг/см (при 4.8 В.), 7,5 кг/см (при 6 В.)	8,5 кг/см (при 4,8 В.), 10 кг/см (при 6 В.)	5 кг/см (при 4.8 В.), 6 кг/см (при 6 В.)
Материал шестерней	Металл	Металл	Нейлон
Габариты	40,8 x 20,1 x 38 мм	40 x 20 x 42 мм	40,8 x 20,1 x 38 мм
Цена	680 руб.	304 руб.	500 руб.

Сравнивая эти варианты, можно прийти к выводу, что MG995, продемонстрированный на рисунке 16, обладая своей низкой ценой и имея похожие характеристики, превосходит аналоги. Данная модель даёт возможность поворота в двух направлениях в пределах от 0 до 180°. Поэтому этот сервомотор подойдёт, чтобы поднять робота из любого положения.

Рисунок 16 – Сервомотор MG995

[14] FS5106M specs [Электронный ресурс] URL: https://www.rhydolabz.com/documents/24/FS5106M_specs.pdf (дата обращения 11.03.2022)
[15] MG995 – Tower-Pro [Электронный ресурс] URL: https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf (дата обращения 11.03.2022)
[16] MS6N40 datasheet [Электронный ресурс] URL: https://datasheetspdf.com/pdf-file/1004964/Bruckewell/MS6N40/1 (дата обращения 14.03.2022)

Для обеспечения питания устройства необходимо четыре литий-ионных аккумулятора типа 18650, которые должны располагаться в батарейном отсеке. Аккумулятор – это устройство, которое хранит в себе энергию и использует её в дальнейшем. Аккумулятор 18650, продемонстрированный на рисунке 17, представляет собой батарейку цилиндрической формы. Характеристики аккумулятора 18650 представлены в таблице 7[17].

Рисунок 17 – Аккумулятор 18650

Таблица 7 – Характеристики аккумулятора 18650

Ёмкость	2600 мАч
Рабочее напряжение	3,5-4 В.
Интервал рабочих температур	От -20 до +600 °С
Наличие защиты от переразряда и перезарядки	Есть
Ресурс	От 1000 и более циклов
Срок службы при активной эксплуатации	От 7 до 10 лет
Габариты	Диаметр 18 мм, длина 65 мм
Цена за 4 шт.	690 руб.

Батарейный отсек продемонстрирован на рисунке 18.

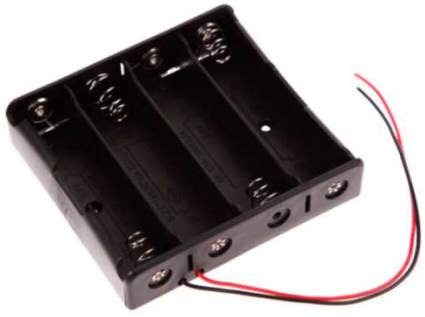


Рисунок 18 – Батарейный отсек для аккумуляторов 18650

[17] 18650H-2600mAh [Электронный ресурс] URL: https://www.ineltro.ch/media/downloads/SAAItem/45/45958/36e3e7f3-2049-4adb-a2a7-79c654d92915.pdf (дата обращения 16.03.2022)

Некоторым датчикам необходимо меньшее напряжение, чем другим. Для понижения напряжения от батарей используются два понижающих миниатюрных DC-DC преобразователя. Понижающий DC-DC преобразователь – это устройство, преобразующее меньшее напряжение из получаемого более высокого напряжения. Сравнение некоторых преобразователей представлено в таблице 8[18, 19].

Таблица 8 – Сравнение понижающих DC-DC преобразователей

Критерии\DC-DC преобразователь	MH-Mini-360	MP2315
Входное напряжение	4,75-23 В.	4,5-24 В.
Выходное напряжение	1-17 В. (регулируемое)	1-12 В. (регулируемое)
Выходной ток	До 3 А.	До 3 А.
Эффективность преобразования	96%	96%
Частота преобразования	340 кГц	500 кГц
Рабочая температура	От -40 до +85 °C	От -40 до +125 °C
Защита от перегрева	Есть	Есть
Габариты	17 x 11 x 3,8 мм	20 x 11 x 5 мм
Цена	190 руб.	41 руб.

Из данных вариантов наиболее подходящим для проекта является MP2315, продемонстрированный на рисунке 19. У него низкая цена и он обладает такими же характеристиками, как аналог MH-Mini-360. На нём есть контакт EN, который позволяет включать и выключать его. Также в данном модуле есть возможность установки определённого выходного напряжения.

Рисунок 19 – Понижающий DC-DC преобразователь MP2315

[18] MINI-360 [Электронный ресурс] URL: https://www.matts-electronics.com/wp-content/uploads/2018/06/MINI-360.pdf (дата обращения 17.03.2022)

[19] MP2315S Datasheet [Электронный ресурс] URL: https://static.chipdip.ru/lib/683/DOC011683595.pdf (дата обращения 17.03.2022)

Для включения самобалансирующего робота используется кнопочный переключатель KCD11, представленный на рисунке 20. Его цена – 20 рублей. Переключатель с двумя положениями может замыкать электрическую цепь, а также размыкать её при нажатии на клавишу.

Рисунок 20 – Кнопочный переключатель KCD11

В результате был проведен анализ элементной базы устройства, были подобраны необходимые компоненты, обоснован их выбор.

2.3 Проектирование устройства

2.3.1 Выбор инструментального обеспечения проекта

Построить структурно-функциональную схему проекта и блок-схему, демонстрирующую алгоритм работы программы, можно с помощью такого ПО, как Microsoft Visio, LucidChart и yEd Graph Editor. Во всех этих программах есть блоки с разными типами данных и возможностью выбора оформления. Из данного ПО была выбрана Microsoft Visio. Это программа, в которой есть возможность создавать, изменять, а также экспортировать блок-схемы в любом удобном формате.

Для разработки принципиальной схемы проекта можно использовать такие сервисы, как EasyEDA, EAGLE и TinkerCAD. В них имеется редактор принципиальных схем и есть возможность создания печатных плат. Для проектируемого устройства был выбран бесплатный онлайн сервис EasyEDA. Он позволяет автоматически переводить схемы в режим проектирования

печатной платы. Также имеется возможность предварительного просмотра печатной платы в виде 3D модели.

Чтобы спроектировать корпус устройства, можно воспользоваться программами для трёхмерного моделирования. К их числу относятся AutoCAD, Fusion 360, FreeCAD и другие системы проектирования. Для разрабатываемого устройства была выбрана FreeCAD. Это бесплатная программа, в которой можно смоделировать любой объект. В ней имеется менеджер дополнений, позволяющий установить необходимые функциональные расширения, созданные пользователями. Также есть возможность работы с чертежами.

Для написания управляющей программы существуют такие интегрированные среды разработки, как ArduinoIDE, PlatformIO IDE. В них есть возможность проверки кода на ошибки, а также они позволяют загрузить его в микроконтроллер. Для проекта была выбрана программа ArduinoIDE. В ней встроен компилятор для написания прошивок, которые затем можно загрузить в микроконтроллер. Её преимуществом является то, что в ней есть поддержка контроллеров разных производителей.

2.3.2 Разработка структурной схемы устройства

Необходимо составить структурную схему, которая будет наглядно показывать связи между блоками и элементами устройства, а также как распределяется информация и питание.

В проекте главным элементом управления является микроконтроллер ESP32-D0WDQ6. Он со встроенным Bluetooth модулем входит в блок центрального контроллера, где должен посылать и принимать информационные сигналы от блока датчиков, в котором находятся датчик ультразвукового излучения и гироскоп-акселерометр, и исполнительных устройств, содержащих в себе сервомотор, драйверы шагового двигателя и сами шаговые двигатели. Также микроконтроллер должен обмениваться данными с мобильным приложением.

Питание приходит в устройство от четырёх литий-ионных аккумулятора типа 18650. Драйверы шагового двигателя получают напряжение 14 В.. С помощью понижающих DC-DC преобразователей оно становится равным 5 В. Затем питание подаётся на сервомотор и микроконтроллер, в котором напряжение стабилизируется до 3,3 В., и от него питаются датчики.

Полученный вариант структурной схемы представлен на рисунке 21.

Рисунок 21 – Структурная схема

В результате была разработана структурная схема устройства.

2.3.3 Разработка принципиальной схемы устройства

Принципиальная схема устройства предназначена для наглядного представления связей в частях электрических устройств. Принципиальная схема самобалансирующего робота представлена на рисунке 22.

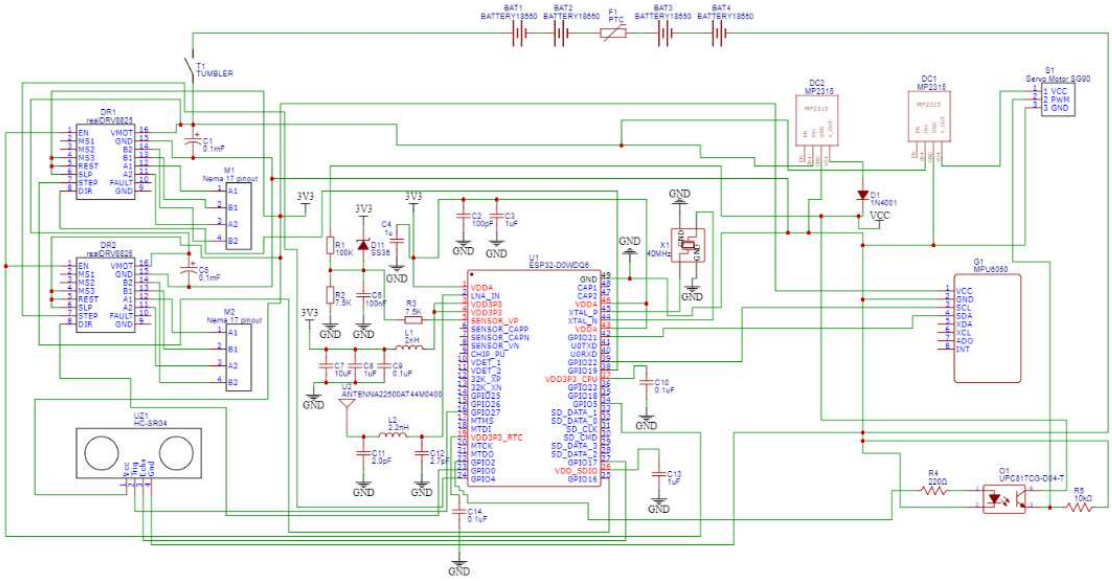


Рисунок 22 – Принципиальная схема

Для сглаживания импульсов от драйверов необходимо 2 конденсатора ёмкостью 0,1 mF и напряжением 16 В.. Для защиты схемы от пробоя установлен диод серии 1N4001, расположенный на выходе понижающего преобразователя, напряжение от которого идёт к микроконтроллеру. Чтобы исключить перегрев, в устройстве используется токовый предохранитель на 5 А., идущий от аккумуляторов. Также в схеме присутствует оптопара PC817, соединяющая выход микроконтроллера с сигнальным контактом сервомотора. Она нужна для обеспечения гальванической развязки, которая используется для защиты оборудования от повреждений электрическим током, между высоковольтной и низковольтной частями. К оптопаре от микроконтроллера идёт токоограничивающий резистор на 220 Ом. Чтобы не было напряжения при выключении питания, около выхода оптрона устанавливается стягивающий резистор на 10 кОм, позволяющий ограничить ток.

2.3.4 Разработка печатной платы устройства

Печатная плата – это пластина, изготовленная из диэлектрического материала, на которой имеются электропроводящие дорожки. Она предназначена для соединения различных электронных звеньев. Их контакты припаивают к элементам проводящего рисунка.

Печатная плата проектируемого устройства была получена с помощью встроенной функции преобразования принципиальной схемы в EasyEDA. Лицевая сторона платы проекта представлена на рисунке 23, а обратная сторона на рисунке 24. Её 3D-модель продемонстрирована на рисунке 25.

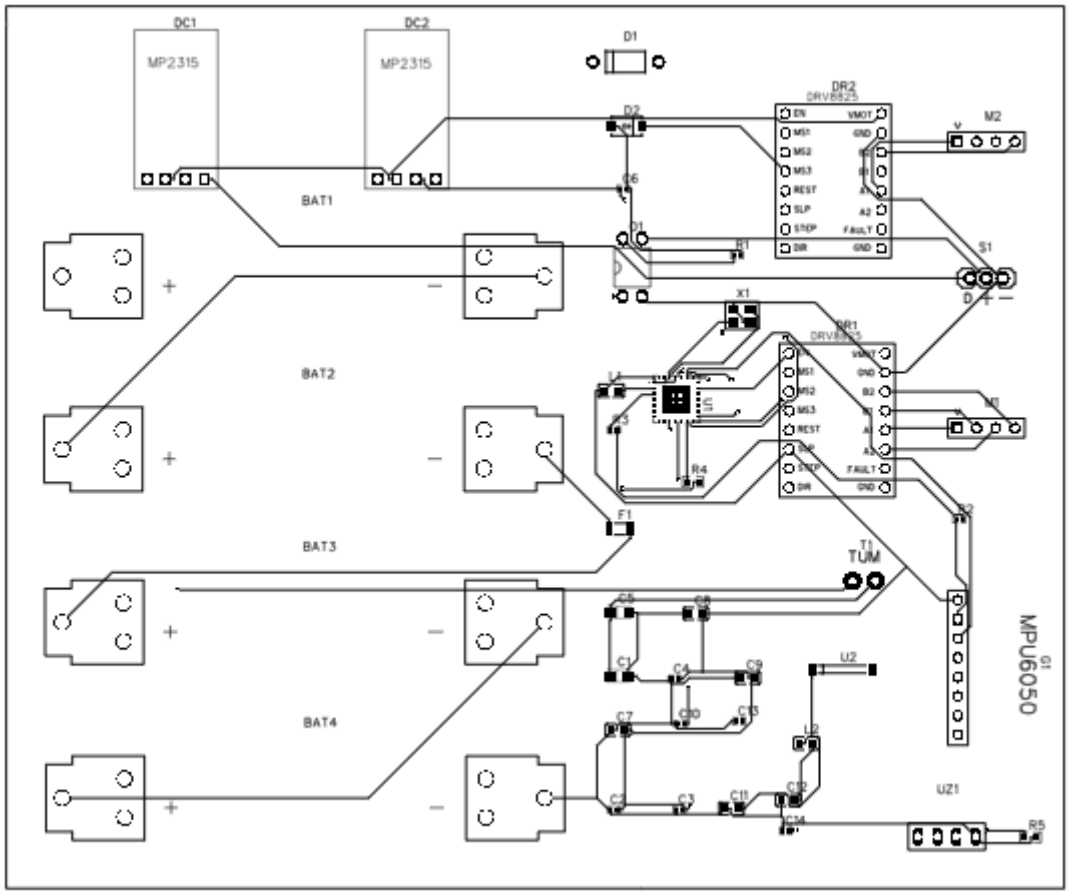


Рисунок 23 – Лицевая сторона печатной платы

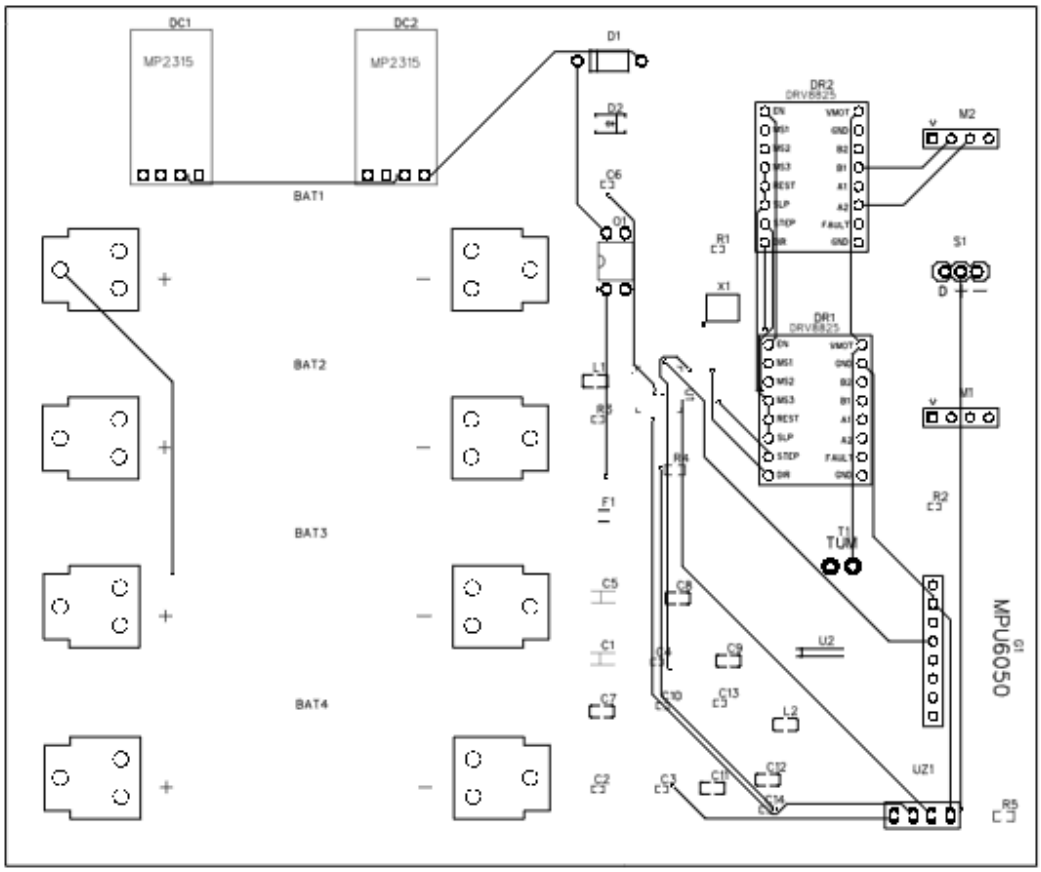


Рисунок 24 – Обратная сторона печатной платы

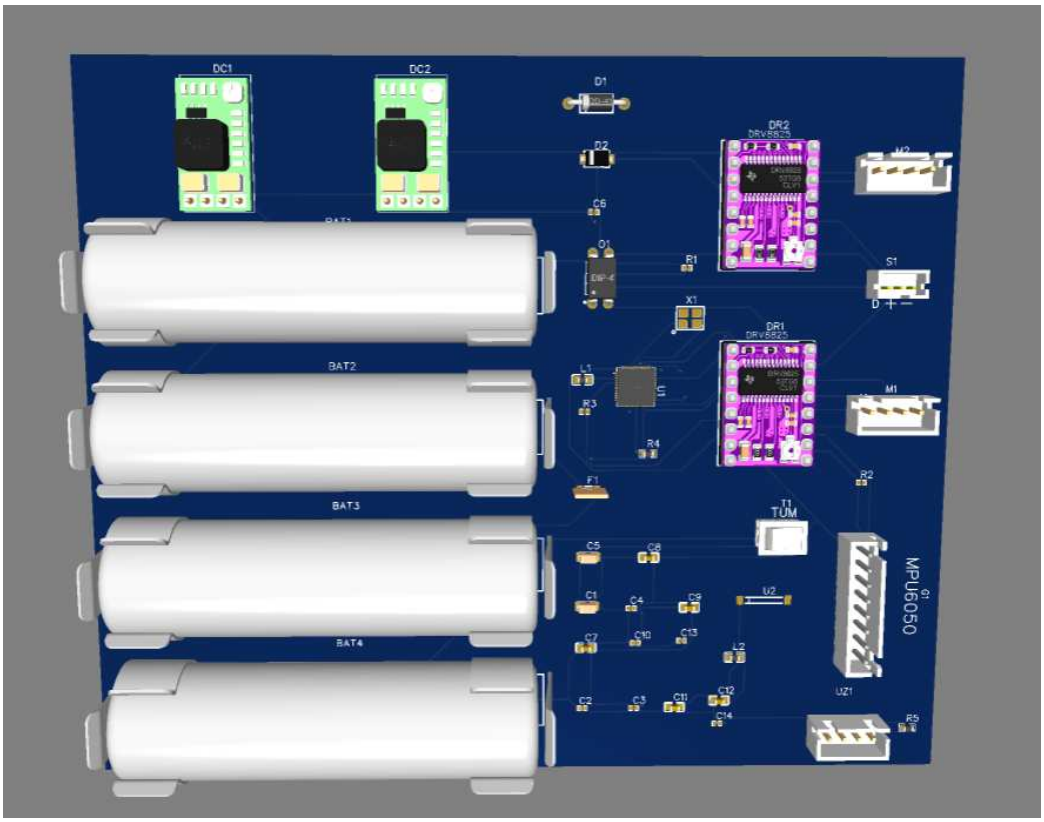


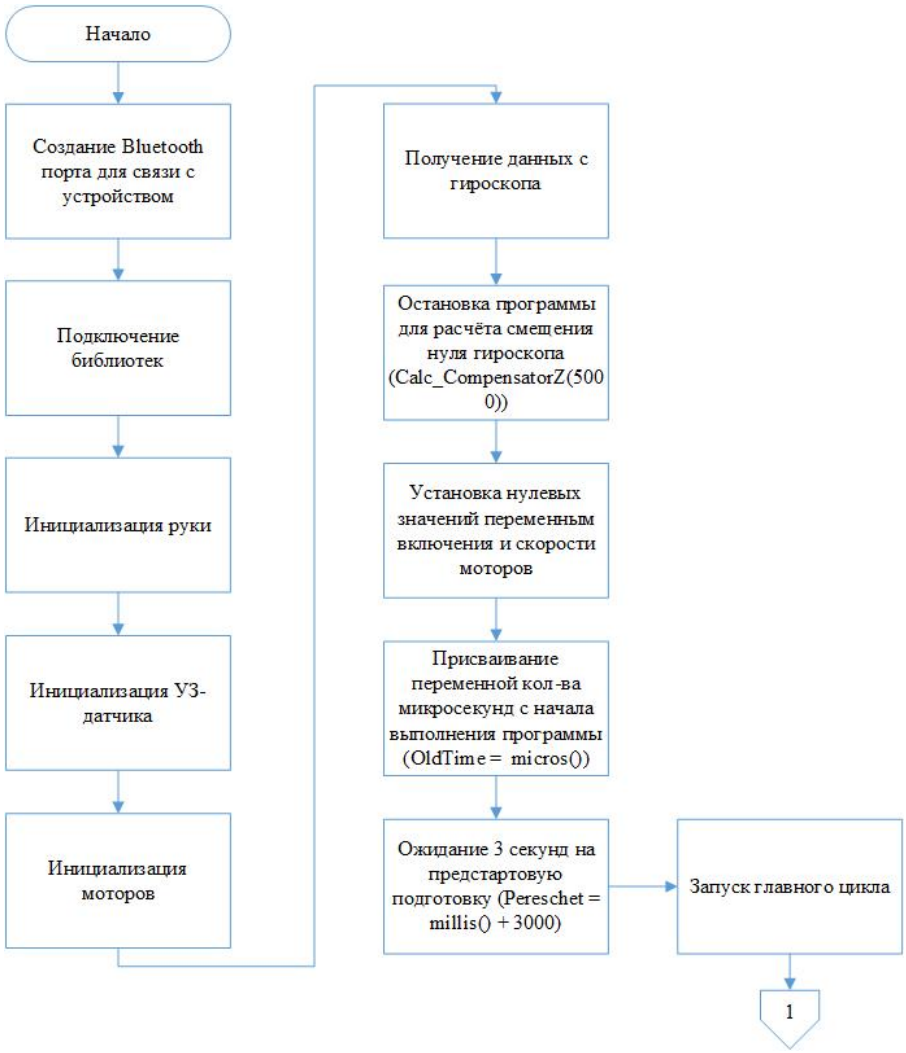
Рисунок 25 – 3D-модель печатной платы

В результате была разработана печатная плата устройства, выполнена её трассировка и получена 3D-модель.

2.3.5 Разработка алгоритма работы программы устройства

Для написания управляющего кода сначала необходимо разработать алгоритм работы программы. Это можно сделать с помощью блок-схемы, которая представляет собой последовательные шаги в виде блоков. К её преимуществам можно отнести наглядность выполнения всех действий, а также простоту реализации.

В первую очередь необходимо подключить библиотеки, позволяющие работать с датчиками, и внутренние программные файлы, в которых отдельно прописан код для каждой задачи устройства. После этого можно приступать к написанию цикла. Весь алгоритм работы программы представлен на рисунке 26.



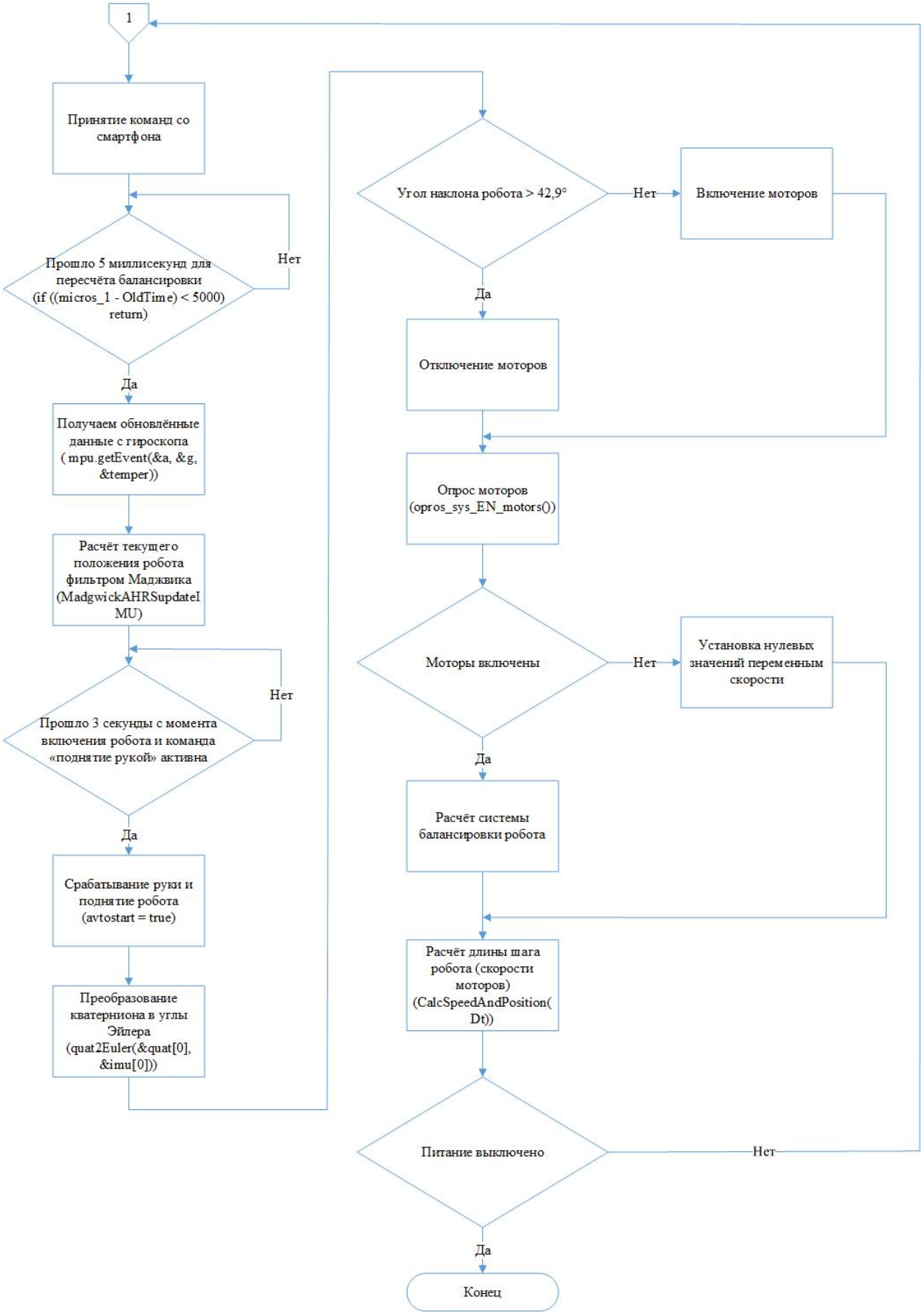


Рисунок 26 – Блок-схема алгоритма работы программы

В результате был разработан алгоритм работы управляющей программы.

2.3.6 Разработка программного обеспечения

Для написания кода к проектируемому устройству потребуются вспомогательные внутренние файлы, написанные на Wiring, в которых будут реализованы отдельные функции. Главный цикл будет вызывать участки кода из них в нужный момент. Для этого в начале программы необходимо подключить следующие файлы:

- ZnachPerem.h, в котором находятся фиксированные значения величин и переменные;
- MadgwickAHRS.h, содержащий в себе расчёт фильтра Маджвика;
- OpisMotor.h, описывающий инициализацию моторов и определяющий их порты с помощью файла MotPin.h;
- NastroikaSonara.h, в котором расписаны функции ультразвукового датчика;
- ServoRuka.h, предназначенный для работы с функциями поднятия и поворота руки;
- Prerivaniya.h, нужный для обеспечения прерываний в цикле;
- RaschetShag.h, необходимый для расчёта шагов (скорости) моторов;
- ObmenBL.h, содержащий обмен командами по Bluetooth со смартфоном;
- Hranenie.h, в котором хранятся настройки, связанные с положением робота.

Библиотека MadgwickAHRS.h представлена 3 функциями:

1. MadgwickAHRSupdate;
2. MadgwickAHRSupdateIMU;
3. quat2Euler.

В проекте для реализации расчёта фильтра используется вторая, так как она обходится без применения магнетометра, основываясь на данных

гироскопа и акселерометра. Для преобразования кватернионов в углы Эйлера применяется функция quat2Euler.

Файл NastroikaSonara.h включает в себя инициализацию ультразвукового датчика, а также 4 этапа анализа измеряемого им расстояния:

1. перевод максимально возможного значения расстояния из сантиметров в микросекунды и генерация импульса;
2. отражение сигнала;
3. пересчёт времени в расстояние;
4. получение значения в миллиметрах.

Файл Prerivaniya.h реализует режим прерываний каждые 10 микросекунд. Это значит, что в течение установленного времени на микроконтроллер будут приходить команды, связанные с проверкой выполнения шагов моторов, работой ультразвукового датчика и руки, а также проверяться на выполнение.

В файле RaschetShag.h есть функции опрашивания моторов (включены ли они) и нахождения длины их шага. Для этого ограничивается скорость двигателей и длительность шагов с помощью команды constrain. Из данного файла берётся командная скорость, настраиваемая в мобильном приложении, а также реальная на текущий момент.

Файл ObmenBL.h содержит в себе функцию BT_input и оператор выбора, которые принимают команды, поступающие во время взаимодействия с устройством, и на основе этого выполняются какие-либо действия, например остановка робота.

В главном цикле после подключения всех библиотек и файлов устанавливается пересчёт балансировки каждые 5 миллисекунд. После этого прописываются значения углов наклона робота, при которых моторы можно включать. Если они включены, то начинается расчёт балансировки робота и скорости моторов.

Для взаимодействия с роботом было выбрано приложение Bluetooth RC Car. Чтобы запрограммировать на его кнопки определённые функции устройства, необходимо использовать оператор выбора с командами, продемонстрированными на рисунке 27.

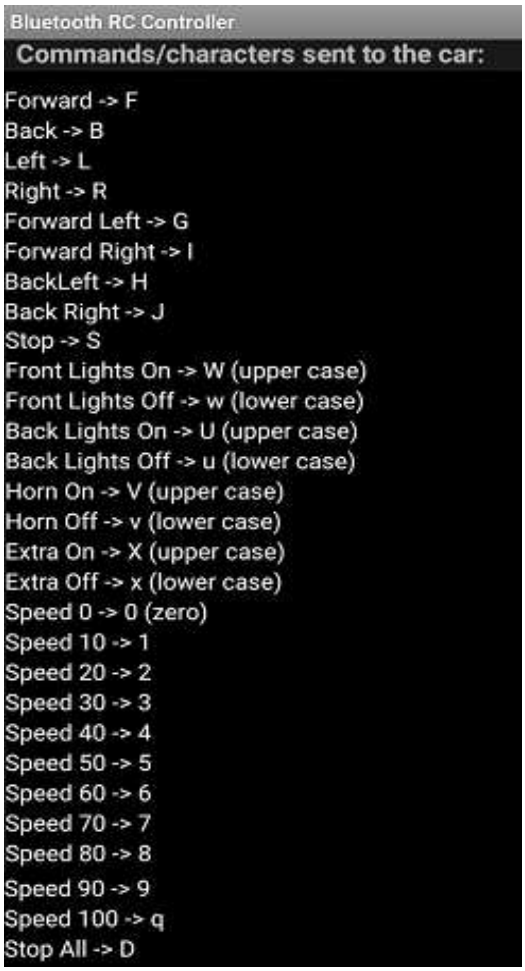


Рисунок 27 – Команды в приложении

Каждая команда предназначена для определённого действия.

Forward (F) – это движение вперёд. Принимая эту команду, робот едет вперёд, пока не обнаружит на своём пути препятствие. Если в зоне действия ультразвукового датчика появится объект, мешающий проезду, то устройство развернётся налево.

При использовании Back (B) робот едет назад.

Left (L) позволяет поворачиваться налево на месте.

Right (R) – поворот направо.

Forward Left (G) предоставляет возможность повернуть влево при движении.

Forward Right (I) – отвечает за поворот вправо на ходу.

Команда Back Left (H) отправляет роботу сигнал, что необходимо повернуть направо с помощью заднего хода.

Back Right (J) – поворот влево задним ходом.

Stop (S) – остановка.

Horn On (V) – поднятие робота с помощью руки.

Horn Off (v) – возвращение руки в исходное положение.

Speed 0-90 (0-9) и Speed 100 (q) – скорости, устанавливаемые роботу в процентах.

Остальные команды не используются.

Полный листинг программы продемонстрирован в приложении Б.

2.3.7 Разработка корпуса устройства

В ходе работы был спроектирован корпус самобалансирующего робота, представленный на рисунке 28, и подготовлены чертежи устройства, продемонстрированные в приложении В на рисунках В.1 – В.7.

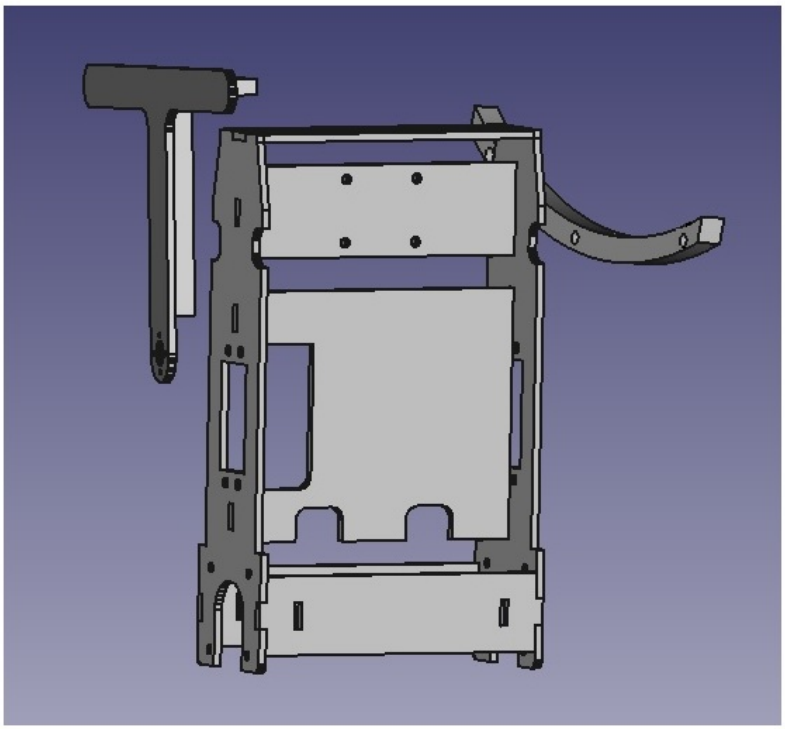


Рисунок 28 – Корпус самобалансирующего робота

Изготовленное устройство со всеми установленными элементами представлено на рисунке 29.

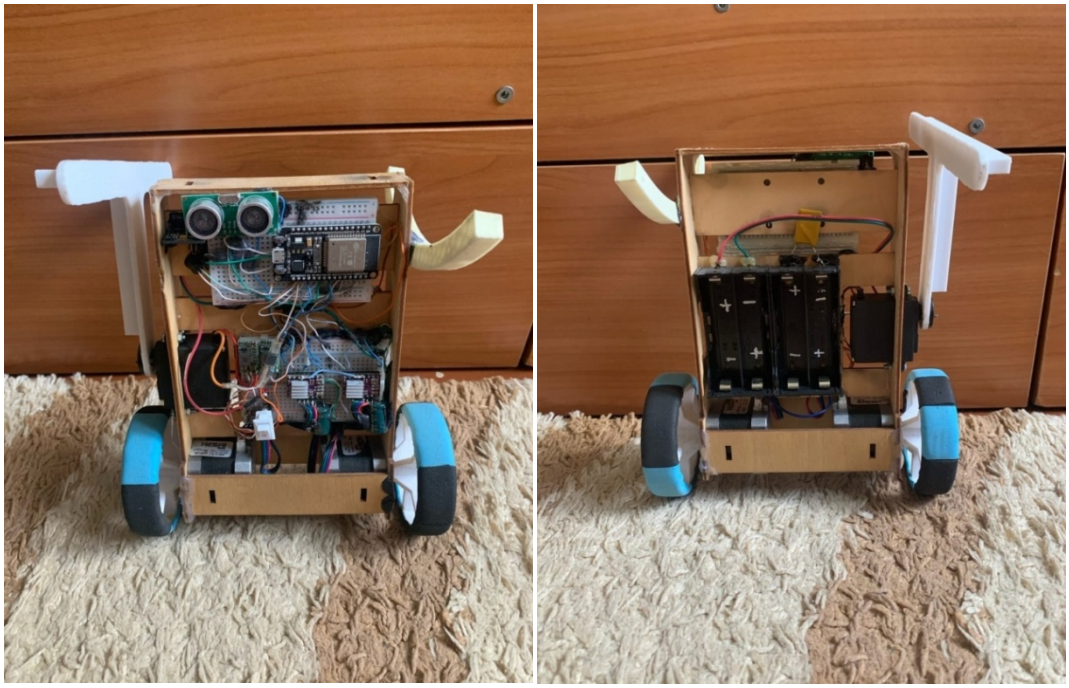


Рисунок 29 – Изготовленное устройство

У робота корпус выполнен из нескольких частей фанеры, соединённых термоклеем. Внутри него расположен центральный контроллер с блоком питания, блоком исполнительных устройств и датчиков. В его основание вмонтированы 2 шаговых двигателя, к которым прикреплены колёса, изготовленные из пластика. Их 3D-модель продемонстрирована на рисунке 30.

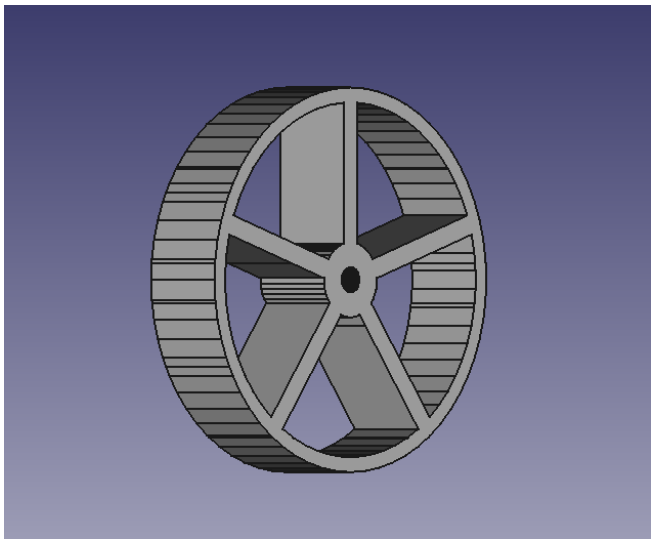


Рисунок 30 – Колесо робота

Слева лицевой стороны на корпусе устройства есть отверстие под сервомотор, на котором зафиксирована подвижная рука в исходном положении, справа – дуга, защищающая его от падения. Датчик ультразвукового излучения находится в верхней части робота. Таким образом, он позволяет обнаружить больше объектов. Под ним располагается гироскоп, определяющий положение устройства.

На плате расположен переключатель, позволяющий включить робота. При нажатии на клавишу загорается красный диод микроконтроллера, и в это время в приложении можно подключиться к устройству по Bluetooth. Для этого нужно открыть настройки, нажав на значок шестерёнки, и выбрать функцию «Connect to car», представленную на рисунке 31.

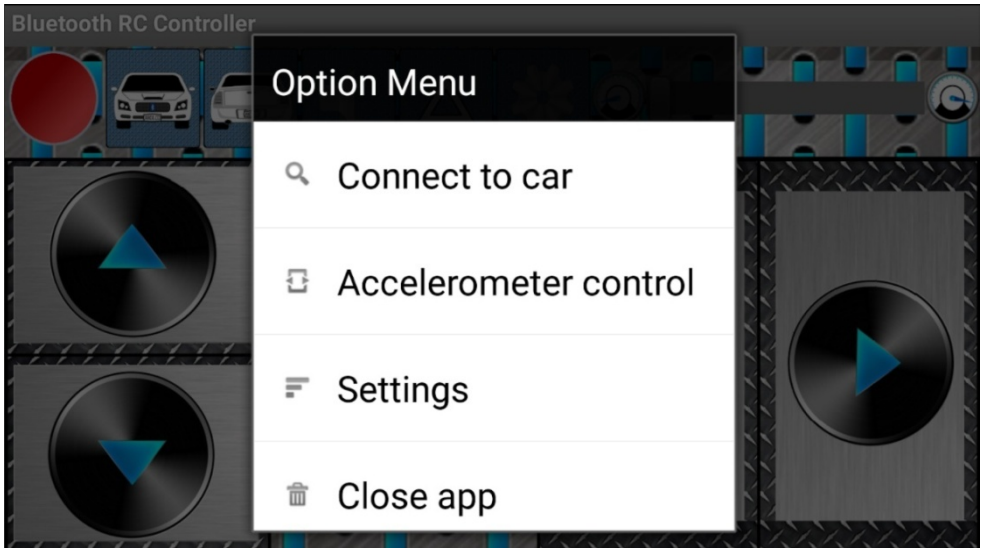


Рисунок 31 – Функция подключения к роботу

Когда связь установится, в приложении будет показан зелёный индикатор, нужно нажать на значок динамика, который продемонстрирован на рисунке 32, чтобы с помощью руки поднять самобалансирующего робота.

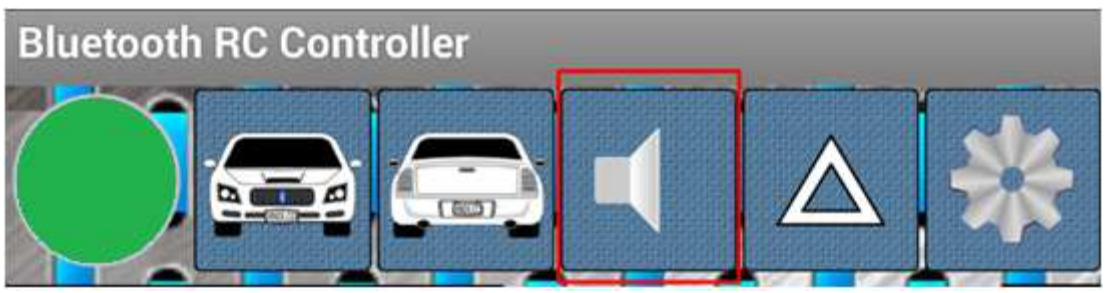


Рисунок 32 – Значок динамика

После этого можно управлять им, нажимая на кнопки в приложении. Скорость робота отображается и регулируется за счёт перемещаемого ползунка. Устройство выполняет все функции в соответствии с техническим заданием.

2.4 Выводы по конструкторскому разделу

В данном разделе были рассмотрены вопросы, связанные с анализом требований, выбором элементной базы и проектированием устройства.

Обеспечение самобалансировки потребовало связанного комплекса датчиков, регулирующих положение робота. Было предложено решение, как и с помощью чего будут выполняться функции устройства, а также подобраны необходимые элементы для этого. Была составлена структурная и принципиальная схемы проектируемого робота, получена его печатная плата. После этого были разработаны блок-схема алгоритма работы программы устройства и сам управляющий код.

3 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

3.1 Расчёт затрат на разработку программы для микроконтроллера

Рассчитать затраты на разработку программы для микроконтроллера можно по следующей формуле:

$$З_{РПР} = З_{ФОТР} + З_{ОВФ} + З_{ЭВМ} + З_{СПП} + З_{ХОН} + Р_{Н}, \quad (5)$$

где $З_{ФОТР}$ – общий фонд оплаты труда разработчиков программы;

$З_{ОВФ}$ – начисления на заработную плату разработчиков программы во внебюджетные фонды;

$З_{ЭВМ}$ – затраты, связанные с эксплуатацией техники;

$З_{СПП}$ – затраты на специальные программные продукты, необходимые для разработки программы для микроконтроллера;

$З_{ХОН}$ – затраты на хозяйственно-операционные нужды (бумага, литература, носители информации и т.п.);

$Р_{Н}$ – накладные расходы (РН= 30% от ЗФОТР).

При разработке программы для микроконтроллера общее время разработки составило 3 месяца. Из них машинное время составляет 2 месяца.

Фонд оплаты труда за время работы над программой для микроконтроллера:

$$З_{ФОТР} = \sum_{j=1}^{m} О_{Рj} \cdot Т_{РПРj} \cdot (1 + k_{Д}) \cdot (1 + k_{У}), \quad (6)$$

где $О_{рj}$ – оклад j-ого разработчика. В разработке участвовал 1 человек, его оклад составляет 22000 руб.;

$Т_{РПРj}$ – общее время работы над программой в месяцах, $Т_{РПР} = 3$;

$k_{Д}$ – коэффициент дополнительной заработной платы, $k_{Д} = 20\% = 0{,}2$;

$k_{У}$ – районный коэффициент, $k_{У} = 0{,}15$.

Таким образом,

$$З_{ФОТР} = 22000 \cdot 3 \cdot (1 + 0{,}2) \cdot (1 + 0{,}15) = 91080 \text{ руб.}$$

Страховой взнос во внебюджетные фонды складываются из взносов на обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний.

Страховые взносы складываются из обязательного пенсионного страхования (ОПС), отчислений в фонд социального страхования и отчислений в фонд обязательного медицинского страхования.

Тариф на обязательное пенсионное страхование 22%, на медстрахование – 5,1%, на социальное страхование – 2,9% и на защиту от несчастных случаев – 0,2%.

В сумме значение страховых взносов составляет 30,2%.

Сумма начислений на заработную плату во внебюджетные фонды составляет:

$$З_{ОВФ} = 0{,}302 \cdot З_{ФОТР}, \quad (7)$$

$$З_{ОВФ} = 0{,}302 \cdot 91080 = 27506{,}16 \text{ руб.}$$

Затраты, связанные с использованием вычислительной и оргтехники:

$$З_{ЭВМ} = Т_{МРПР} \cdot k_r \cdot n \cdot С_{М-ч}, \quad (8)$$

где k_r– коэффициент готовности ЭВМ, k_r= 0,95;

n – количество единиц техники, равно 1;

$С_{М-Ч}$ – себестоимость машинно-часа, $С_{М-Ч}$= 10 руб.;

$Т_{МРПР}$ – машинное время работы над программой, равно 2 мес.

Перевод рабочего временив часы осуществляется по формуле:

$$Т_{час} = Т_{мес} \cdot Ч_{РД} \cdot Т_{см} \cdot К_{см}, \quad (9)$$

где $Т_{час}$ – рабочее время, ч;

$Т_{мес}$ – рабочее время, мес., ($Т_{мес}$= 2);

$Ч_{РД}$ – число рабочих дней, ($Ч_{РД}$=22);

$Т_{см}$ – продолжительность рабочей смены, ($Т_{см}$= 8 ч);

$К_{см}$ – количество рабочих смен, ($К_{см}$=1).

Таким образом, время на разработку программы для микроконтроллера с использованием ЭВМ составляет:

$$Т_{час} = 2 \cdot 22 \cdot 8 \cdot 1 = 352 \text{ часа},$$

$$З_{ЭВМ} = 352 \cdot 0{,}95 \cdot 1 \cdot 10 = 3\,344 \text{ руб.}$$

Чтобы найти затраты на специальные программные продукты, необходимые для разработки программы для микроконтроллера, нужно воспользоваться следующей формулой:

$$З_{СПП} = \sum_{р=1}^{n} Ц_р, \quad (10)$$

где $Ц_р$ – цена р-го специального программного продукта.

Перечень программных продуктов специального назначения приведен в таблице 9.

Таблица 9 – Программные продукты специального назначения

№	Название ПП	Цена, руб.
1	ArduinoIDE	0
2	EasyEDA	0
	Итого:	0

Для написания программы применялась среда разработки ArduinoIDE стоимостью 0 рублей.

Использованные программные продукты бесплатны, поэтому:

$$Зспп = 0 \text{ руб.}$$

Затраты на хозяйственно-организационные нужды приведены в таблице 10 и вычисляются по формуле:

$$З_{ХОН} = \sum_{\tau=1}^{n} Ц_{\tau} \cdot К_{\tau}, \quad (11)$$

где Цτ - цена τ-ого товара, руб.;

Кτ - количество τ-ого товара.

Таблица 10 – Затраты на хозяйственно-организационные нужды

Наименование	**Цена за единицу (руб.)**	**Кол-во (шт.)**	**Всего (руб.)**
USB-флеш-накопитель	600	1	600
Бумага	0,9	100	90
		Итого:	690

$$З_{ХОН} = 600 \cdot 1 + 0{,}9 \cdot 100 = 690 \text{ рублей}$$

Накладные расходы:

$$Р_{Н} = З_{ФОТР} \cdot k_{НР}, \quad (12)$$

$$Р_{Н} = 91080 \cdot 0{,}302 = 27506{,}16 \text{ руб.}$$

Таким образом, затраты на разработку программы для микроконтроллера, рассчитанные по формуле 5, составят:

$$З_{РПР} = 91080 + 27506{,}16 + 3344 + 0 + 690 + 27506{,}1 = 150126{,}32 \text{ руб.}$$

3.2 Расчёт затрат на внедрение программы для микроконтроллера

Формула для рассчета затрат на внедрение программы для микроконтроллера:

$$З_{ВПР} = З_{М} + З_{КТС} \cdot (1 + k_{тун}) + З_{ПО} + З_{ФОТВ} + З_{ОВФ} + З_{ЭВМ} + Р_{ком} + Р_{Н}, \quad (13)$$

где $З_{М}$ – затраты на преобретение материалов,руб.;

$З_{КТС}$ – затраты на приобретение комплекса технических средств, руб.;

$З_{ПО}$ – затраты на приобретение программного обеспечения, руб.;

$З_{ФОТВ}$ – зарплаты на оплату труда работников, занятых внедрением проекта, руб.;

$З_{ОВФ}$ – отчисления в страховые фонды с заработной платы работников, занятых внедрением проекта, руб.;

$З_{ЭВМ}$ – затраты, связанные с эксплуатацией ЭВМ при внедрении проектного решения, руб.;

$Р_{ком}$ – командировочные расходы, руб.;

$Р_{н}$ – накладные расходы, руб.;

$К_{тун}$ – коэффициент транспортирования, установки и наладки комплекса технических средств.

Затраты на приобретение материалов (ЗМ) приведены в таблице 11.

Таблица 11 – Затраты на приобретение материалов

Наименование	**Цена за единицу (руб.)**	**Кол-во (шт.)**	**Всего (руб.)**
Фанера для изготовления корпуса (2300х2100 мм.)	540	1	540
ESP32-D0WDQ6	250	1	250
Nema 17HS4401	600	2	1200
DRV8825	90	2	180
US-025	47	1	47
MPU6050	83	1	83
MG995	304	1	304
Аккумулятор 18650	172,5	4	690
Батарейный отсек	123,82	1	123,82
Пластик PETG	658руб./267м.	10 м.	24,7
MP2315	41	2	82
KCD11	20	1	20
Крепёжные элементы	2,25	23	51,75
Кабель USB для прошивки	106	1	106
Провода кабеля Aopen UTP (2 м.)	169	1	169
Соединительные провода типа папа-папа	4,9	8	39,2

Продолжение таблицы 11

Итого:	3910,47

Дополнительного приобретения компьютеров или других КТС не требуется, следовательно, $З_{КТС} = 0$.

Затраты на приобретение программного обеспечения в данном случае равны затратам на разработку и составляют $З_{ПО} = 150126,32$ руб.

Внедрением программы занимается один специалист. Его оклад составляет 18000 руб. Время внедрения – 0,5 месяца.

Формула для расчёта затрат на оплату труда и отчисления в страховые фонды:

$$З_{ФОТВ} = 18000 \cdot 0,5 = 9000 \text{ руб.}$$

$$З_{ОВФ} = 9000 \cdot 0,302 = 2718 \text{ руб.}$$

Затраты, связанные с эксплуатацией ЭВМ при внедрении проектного решения составят:

$$З_{ЭВМ} = 0,5 \cdot 22 \cdot 8 \cdot 10 = 880 \text{ руб}$$

Командировочные расходы не предусмотрены, поэтому $Р_{ком} = 0$.

Коэффициент накладных расходов по данным организации составляет $k_{НР} = 0,302$, следовательно, величина накладных расходов равна 2718 руб.

Затраты на внедрение программы составят:

$$З_{ВПР} = 3910,47 + 0 + 150126,32 + 9000 + 2718 + 880 + 0 + 2718 =$$

$$= 169352,79 \text{ руб.}$$

3.3 Расчёт эксплуатационных текущих затрат по программе для микроконтроллера

Годовые затраты на обработку результатов до внедрения разработанной программы для микроконтроллера рассчитываются по формуле:

$$С_1 = ЗП_1 + ОТ_{ВН1} + З_{ЭВМ1} + М_{з1} + НР_1, \quad (14)$$

где $ЗП_1$ – затраты на оплату труда сотрудника на выполнение функций до внедрения проекта;

$ОТ_{ВН1}$ – отчисления в страховые фонды;

$З_{ЭВМ1}$ – затраты, связанные с эксплуатацией ЭВМ;

$М_{з1}$ – годовые материальные затраты на сопровождение программы для микроконтроллера составляют 1500 руб.;

$НР_1$ – накладные расходы.

Формула для расчёта временных затрат работы сотрудника в месяцах:

$$Т_{1мес} = \frac{Т_{1час}}{Ч_{рд} \cdot Ч_{рч}}, \quad (15)$$

где $Т_{1мес}$, $Т_{1час}$ – время, затрачиваемое сотрудником на обработку результатов, в месяцах и часах соответственно ($Т_{1час}$= 616 часов);

$Ч_{рд}$ – число рабочих дней в месяц;

$Ч_{рч}$– число рабочих часов в день.

$$Т_{1мес} = \frac{616}{22 \cdot 8} = 3{,}5 \text{ мес},$$

Затраты на оплату труда сотрудника составят:

$$ЗП_1 = О_с \cdot Т_{1мес} \cdot (1 + k_Д) \cdot (1 + k_У), \quad (16)$$

где $О_с$ – оклад сотрудника (составляет 18000 руб.);

$$ЗП_1 = 18000 \cdot 3{,}5 \cdot (1 + 0{,}2) \cdot (1 + 0{,}15) = 86940 \text{ руб.}$$

Страховой взнос до внедрения вычисляют по формуле:

$$ОТ_{ВН1} = ЗП_1 \cdot 0{,}302, \quad (17)$$

$$ОТ_{ВН1} = 86940 \cdot 0{,}302 = 26255{,}88 \text{ руб.}$$

Формула для расчёта затрат, связанных с эксплуатацией ЭВМ до внедрения:

$$З_{ЭВМ1} = Т_{1час} \cdot С_{м-ч}, \quad (18)$$

$$З_{ЭВМ1} = 616 \cdot 10 = 6160 \text{ руб.}$$

После подстановки соответствующих значений в формулу 14 было получено:

$$С_1 = 86940 + 26255{,}88 + 6160 + 1500 = 120855{,}88 \text{ руб.}$$

Формула для расчёта годовых затрат на эксплуатацию системы после внедрения программы для микроконтроллера:

$$С_2 = ЗП_2 + ОТ_{ВН2} + З_{ЭВМ2} + М_{з2} + НР_2, \quad (19)$$

где $ЗП_2$ – затраты на оплату сотрудника после внедрения;

$ОТ_{ВН2}$ – отчисления в страховые фонды;

$З_{ЭВМ2}$ – затраты, связанные с эксплуатацией ЭВМ после внедрения;

$М_{з2}$ – годовые материальные затраты на сопровождение программы для микроконтроллера составляют 2000 руб.;

$НР_2$ – накладные расходы.

Временные затраты работы сотрудника в месяцах рассчитываются по формуле:

$$Т_{2мес} = \frac{Т_{2час}}{Ч_{рд} \cdot Ч_{рч}}, \quad (20)$$

где $Т_{2мес}$, $Т_{2час}$ – время, затрачиваемое сотрудником на обработку результатов, в месяцах и часах соответственно ($Т_{2час}$= 212 часов);

$Ч_{рд}$ – число рабочих дней в месяц;

$Ч_{рч}$ – число рабочих часов в день;

$$Т_{2мес} = \frac{212}{22 \cdot 8} = 1{,}2 \text{ мес.}$$

Затраты на оплату труда сотрудника составят:

$$ЗП_2 = О_с \cdot Т_{2мес} \cdot (1 + k_Д) \cdot (1 + k_У), \quad (21)$$

где $О_с$ – оклад сотрудника (составляет 18000 руб.);

$$ЗП_2 = 18000 \cdot 1{,}2 \cdot (1 + 0{,}2) \cdot (1 + 0{,}15) = 29808 \text{ руб.}$$

Отчисления на социальные нужды после внедрения вычисляются по формуле:

$$ОТ_{ВН2} = ЗП_2 \cdot 0{,}302, \quad (22)$$

$$ОТ_{вн2} = 29808 \cdot 0{,}302 = 9002{,}02 \text{ руб.}$$

Формула для расчёта затрат, связанных с эксплуатацией ЭВМ после внедрения:

$$З_{ЭВМ2} = Т_{2час} \cdot С_{м-ч}, \quad (23)$$

$$З_{ЭВМ2} = 212 \cdot 10 = 2120 \text{ руб.}$$

Отсюда,

$$С_2 = 29808 + 9002{,}02 + 2120 + 2000 = 42930{,}02 \text{ руб.}$$

Таким образом, текущие затраты на содержание системы до внедрения программы составляют 120855,88 руб., после внедрения 42930,02 руб.

3.4 Расчёт экономической целесообразности разработки и внедрения информационных технологий

Для разрабатываемого проекта расчёт экономической эффективности производится исходя из следующих условий:

– годовые текущие затраты до внедрения автоматизированной системы, C_1= 120855,88 руб.;

– годовые текущие затраты после внедрения системы, C_2 = 42930,02 руб.;

– горизонт расчёта принимается исходя из срока использования разработки, Т = $T_н$ = 3 годам;

– шаг расчёта равен одному году, t =1 году;

– капитальные вложения равны затратам на создание системы, К=169352,79 руб.;

– норма дисконта равна норме дохода на капитал, Е = 12%.

Ожидаемая условно-годовая экономия от внедрения системы рассчитывается по формуле:

$$Э_{уг} = C_1 - C_2 + \sum Э_i, \quad (24)$$

где $Э_{уг}$ – величина экономии, руб.;

C_1 – годовые текущие затраты до внедрения автоматизированной системы, руб.;

C_2 – годовые текущие затраты после внедрения системы, руб.;

$\sum Э_i$ – ожидаемый дополнительный эффект от различных факторов, руб.

Так как основным фактором, по которому производится расчёт экономической эффективности от внедрения программы для микроконтроллера, является уменьшение времени обработки результатов тестирования и дополнительный эффект не учитывается, то $\sum Э_i = 0$.

После подстановки вычисленных выше значений в формулу было получено:

$$Э_{уг} = 120855{,}88 - 42930{,}02 = 77925{,}86 \text{ руб.}$$

где $Э_{уг}$ – ожидаемая условно-годовая экономия, руб.

Формула для расчёта ожидаемого годового экономического эффекта от внедрения системы:

$$Э_г = Э_{уг} - К \cdot Е_н, \quad (25)$$

где $Э_г$ – ожидаемый годовой экономический эффект, руб.;

$Э_{уг}$ – ожидаемая условно-годовая экономия, руб.;

К – капитальные вложения (равны затратам на создание системы), руб.;

$Е_н$ – нормативный коэффициент экономической эффективности капитальных вложений.

Нормативный коэффициент экономической эффективности капитальных вложений определяется по формуле:

$$Е_н = \frac{1}{Т_н}, \quad (26)$$

где $Т_н$ – нормативный срок окупаемости капитальных вложений, лет.

После подстановки вычисленных выше значений в формулу было получено:

$$Э_г = 77925{,}86 - 169352{,}79 \cdot 0{,}33 = 22039{,}43 \text{ руб.}$$

Расчётный коэффициент экономической эффективности капитальных вложений находится по формуле:

$$Е_р = \frac{Э_{уг}}{К}, \quad (27)$$

где $Е_р$ – расчётный коэффициент экономической эффективности капитальных вложений;

$Э_{уг}$ – ожидаемая условно-годовая экономия, руб.;

К – капитальные вложения на создание системы, руб.

Отсюда,

$$E_p = \frac{77925,86}{169352,79} = 0,46$$

Расчётный срок окупаемости капитальных вложений составляет:

$$T_p = \frac{1}{E_p}, \quad (28)$$

где Ер – коэффициент экономической эффективности капитальных вложений.

После подстановки вычисленных выше значений в формулу было получено:

$$T_p = \frac{1}{0,46} = 2,2 \text{ года.}$$

Срок окупаемости без дисконтирования составляет 2,2 года.

Чистый дисконтированный доход (ЧДД) определяется как сумма текущих эффектов за весь расчётный период, приведенная к начальному шагу, или как превышение интегральных результатов над интегральными затратами.

При условии, что в течение расчётного периода не возникает инфляция или расчёт производится в базовых ценах, то величина ЧДД для постоянной нормы дисконта вычисляется по формуле:

$$ЧДД = \sum_{t-1}^{T} (P_t - З_t) \cdot \frac{1}{(1+E)^t} - К, \quad (29)$$

где P_t – ожидаемые результаты от внедрения предложений системы, руб.;

$З_t$ – ожидаемые затраты (капитальные и текущие) на создание и эксплуатацию системы, руб.;

$Э_t = (Р_t - З_t)$ – эффект, достигаемый на t-м шаге расчёта;

К – капитальные вложения;

t – номер шага расчёта (t = 1, 2, 3);

Т – горизонт расчёта;

Е – постоянная норма дисконта, 12%.

$Э_t = (Р_t - З_t) = Э_{уг}$ = 77925,86 руб. В том случае, если текущие затраты ($З_t$) на весь срок использования разработки равны 0.

t = 1, 2, 3 год, т.к. предполагается, что результат от внедрения предложенной системы будет с текущего года её внедрения.

Если ЧДД инвестиционного проекта положителен, то проект является эффективным при данной норме дисконта.

Тогда суммарный дисконтированный доход за весь горизонт расчёта равен:

$$ЧДД = Э_1 \cdot \frac{1}{(1+E)} + Э_2 \cdot \frac{1}{(1+E)^2} + Э_3 \cdot \frac{1}{(1+Е)^3}, \quad (30)$$

$$ЧДД = \frac{77925,86}{(1+0,12)} + \frac{77925,86}{(1+0,12)^2} + \frac{77925,86}{(1+0,12)^3} - 169352,79 = 17811,98.$$

Положительное значение чистого дисконтированного дохода, ЧДД> 0, свидетельствует о том, что инвестирование целесообразно и данная система может приносить прибыль в установленном объеме.

Индекс доходности (ИД) представляет собой отношение суммы приведенных эффектов к величине капитальных вложений и вычисляется по формуле:

$$ИД = \frac{1}{К}\sum_{t-1}^{Т}(P_t - З_t) \cdot \frac{1}{(1+E)^t}, \quad (31)$$

где К – величина капиталовложений или стоимость инвестиций.

$$ИД = \frac{187164,77}{169352,79} = 1,11.$$

Инвестиции считаются эффективными, если индекс доходности выше единицы, ИД>1, следовательно, инвестиции в данную систему эффективны.

Внутренняя норма доходности (ВНД):

при $E_1 \rightarrow ЧДД_1 > 0$

$E_2 \rightarrow ЧДД_2 < 0$

$$ВНД = E_1 + \frac{ЧДД_1}{ЧДД_1 - ЧДД_2} \cdot (E_2 - E_1), \quad (32)$$

при $E_1 \rightarrow ЧДД_1 > 0$

$E_2 \rightarrow ЧДД_2 > 0$

$$ВНД = E_1 + \frac{ЧДД_1}{ЧДД_1 + ЧДД_2} \cdot (E_2 - E_1), \quad (33)$$

$E_1 = 0,10$

$$ЧДД_1 = \frac{77925,86}{(1+0,1)} + \frac{77925,86}{(1+0,1)^2} + \frac{77925,86}{(1+0,1)^3} - 169352,79 = 24437,30 \text{ руб.}$$

$E_2 = 0,13$

$$ЧДД_2 = \frac{77925,86}{(1+0,13)} + \frac{77925,86}{(1+0,13)^2} + \frac{77925,86}{(1+0,13)^3} - 169352,79 =$$

$$= 14642,06 \text{ руб.}$$

$E_1 \rightarrow ЧДД_1 > 0$ $\qquad$ $E_2 \rightarrow ЧДД_2 > 0$

$$ВНД = 0,10 + \frac{24437,30}{24437,30 + 14642,06} \cdot (0,13 - 0,10) = 0,119.$$

Таким образом, норма дисконта должна быть в пределах 10% – 13%.

Показатели экономической целесообразности разработки и внедрения программы для микроконтроллера сведены в таблицу 12.

Таблица 12 – Показатели экономической целесообразности разработки и внедрения программы для микроконтроллера

Наименование показателя	**Значения**
Затраты на разработку и внедрение ПП, руб.	169352,79
Ожидаемая экономия от внедрения ПП, руб. в год	77925,86
Чистый дисконтированный доход, руб.	17811,98
Индекс доходности	1,11
Внутренняя норма доходности	0,119
Дисконтированный срок окупаемости, года	2,2
Срок морального старения, года	3

3.5 Выводы по организационно-экономическому разделу

Произведенные расчёты доказывают, что внедрение разработанной в ВКР программы для микроконтроллера приведет к сокращению годовых текущих затрат на 77925,86 рублей. Также можно сделать вывод о том, что разработка и внедрение предлагаемой программы является экономически обоснованной и целесообразной.

4 ОХРАНА ТРУДА И ПРОМЫШЛЕННАЯ ЭКОЛОГИЯ

4.1 Анализ вредных и опасных производственных факторов при пайке деталей, узлов и наладке электронных устройств

Охрана труда – это комплекс средств и способов обеспечения безопасности рабочих на производстве. Среда, в которой они работают, характеризуется таким свойством, как опасность, влияние которого может по-разному отразиться на людях. Они могут получить травму, какое-либо заболевание или умереть.

В условиях трудовой деятельности на специалистов всегда влияют различные производственные факторы, которые негативно воздействуют на них и приносят вред или представляют опасность. Поэтому должны быть созданы необходимые условия труда, позволяющие комфортно и безопасно работать [14].

Вредные факторы на производстве – это неблагоприятные факторы, возникающие в ходе работы, которые могут навредить здоровью и снизить работоспособность человека. Также они приводят к болезням.

Опасные факторы, появляющиеся в трудовой деятельности – это факторы, воздействия которых проявляются в травмах, плохом самочувствии или смерти работника.

В соответствии с [15] данные факторы по способу действия классифицируются на:

- физические, например, различные движущиеся устройства, острые инструменты, электричество и т.д.;
- химические, заключающиеся в синтезе различных веществ, а также их применении;
- биологические, возникающие в результате влияния всевозможных микроорганизмов, которые способны привести к заболеванию;

– психофизиологические, появляющиеся в ходе работы индивидуально у каждого человека, например, физическое утомление, перенапряжение, вызванное умственной активностью работника.

Инженер-электроник – это персонал, занимающийся организацией безошибочной и безотказной эксплуатации, работы электронных устройств.

На людей данной профессии влияют разные негативные факторы.

К ним можно отнести:

– плохую освещённость рабочего места;
– повышенные температуры используемых материалов;
– выделение вредных веществ при пайке;
– воздействие электричества;
– перенапряжение органов зрения;
– влияние шума.

Освещённость помещения, в котором выполняются работы, должна быть обеспечена в соответствии [16]. Освещение может быть естественным и искусственным. Естественное освещение происходит благодаря свету, проникаемому в помещение от солнца или неба. Искусственное освещение осуществляется с помощью использования источников, созданных людьми. Естественное освещение можно грамотно организовать, установив рабочее место на определённом расстоянии от окна, не заграждённого мебелью. Недостаток света может привести к понижению качества выполняемых операций, нарушению зрения, появлению усталости. Поэтому необходимо учитывать этот фактор при организации трудового процесса.

Нагретые поверхности, например, при пайке – установлении контакта между несколькими деталями с помощью расплавленного припоя, могут послужить причиной для получения ожога. Чтобы обеспечить безопасность при выполнении такого вида работы, к пайке допускаются специалисты II группы по электробезопасности, которые прошли обучение, освоили необходимые методы и способы выполнения работ, а также соблюдают

внутренние распорядки организации. При осуществлении паяльных работ выделяются вредные пары, например, олова, канифоли. Они оседают на коже человека и могут попасть в организм, вызвав отрицательные последствия для здоровья, например, пневмокониоз. Для ликвидации вредных веществ из помещения нужно использовать вытяжную вентиляцию. В случае если она отсутствует, то необходимо применять респиратор или защитную маску.

Чтобы избежать воздействия электрического напряжения на специалиста, всё оборудование должно быть изолированным. Если произошёл пробой изоляции оборудования, то человек может получить удар током. Поэтому работающий не должен соприкасаться с частями приборов, на которых воздействует напряжение, либо должен иметь защитные диэлектрические перчатки. В помещении также обязан находиться автоматический выключатель, позволяющий отключать питание устройств при прохождении значения тока, больше установленного и в случае короткого замыкания.

Инженер-электроник работает с различным оборудованием, к числу которого относятся и мониторы. Чтобы не ухудшилось зрение, специалист должен сидеть на определённом расстоянии от этих устройств, и они должны располагаться под установленным углом. Он может использовать специальные защитные очки от излучаемого компьютером синего цвета.

На инженера-электроника также влияет шум, возникающий при работе с компьютерной техникой. Причиной этого может быть работа жёстких дисков и принтеров. Поэтому при выборе оборудования рекомендуется сразу рассматривать варианты, генерирующие минимальный шум. Например, вместо матричного принтера эксплуатировать лазерный. Также повышение уровня шума происходит из-за работающих станков, мостовых кранов и различных передвигающихся механизмов, находящихся в непосредственной близости с рабочим местом специалиста. Чтобы обеспечить снижение шума применяют звукопоглощающий материал, который прикрепляют к стенам и

потолку, выполняют ограждение рабочего места, путём установления различных перегородок.

Специалист также подвергается длительной сидячей работе. У него идёт нагрузка на позвоночник, которая может привести к различным заболеваниям костей. Ему не хватает физических нагрузок. Поэтому нужно проводить зарядку в течение трудовой деятельности, менять рабочую позу, а также делать перерывы, как для личных целей, так и обязательные, установленные руководством, например на уборку помещения. Всё это даёт возможность увеличения производительности труда и минимизации отрицательного влияния на здоровье.

Помещение, в котором работает инженер-электроник, должно быть оборудовано необходимой техникой для обеспечения пожарной безопасности, например, огнетушителями, датчиками дыма, сигнализацией и т.д. Рабочий при выполнении пайки использует паяльник, нагревающийся до высокой температуры. Неосторожное обращение с ним может вызвать не только ожог у человека, но и возгорание каких-либо объектов. Также паяльник при нагревании и остывании нужно класть на специальную подставку, где он не будет соприкасаться своим жалом с находящимися рядом предметами. Специалист должен соблюдать последовательность действий при пожаре и требования безопасности.

В течение работы инженера-электроника также должна проводиться диагностика всего оборудования. Перед началом работы техника проверяется на наличие неисправностей. При обнаружении необходимо выполнить комплекс действий для их устранения. В случае отсутствия неисправностей специалист допускается к работе. Диагностика является важным процессом трудовой деятельности. Она позволяет избежать возможные производственные травмы.

Создание комфортных и не представляющих опасность условий рабочей среды требует применения современных технических средств.

Для защиты от вредного воздействия факторов на предприятиях используют различные средства защиты, позволяющие ослабить или устранить их негативное влияние на человека [17].

Они делятся на 2 группы:

1. коллективные;
2. индивидуальные.

К первой относятся средства защиты, которые находятся в непосредственной близости с рабочим местом и связаны с технологическим оборудованием, используемым специалистами, например, изоляция техники, различные предохранительные устройства.

Средства индивидуальной защиты применяются отдельно для каждого работника, когда техника не предусматривает требуемого уровня безопасности для выполнения работ.

4.2 Расчёт технических средств обеспечения безопасности труда на рабочем месте инженера-электроника

Рабочее место сотрудника, представленное на рисунке 33, должно быть оснащено всеми необходимыми инструментами и техническими средствами. Это позволит сократить время на их поиск, а также повысить продуктивность специалиста.

Рисунок 33 – Рабочее место инженера-электроника

У инженера-электроника на рабочем месте должны присутствовать:

- стол;
- кресло или стул;
- устройства ввода и вывода;
- измерительные приборы;
- паяльник и паяльные принадлежности;
- шкаф для инструментов.

У стола должны быть выдвижные ящики, в которых хранятся необходимые материалы либо часто используемые инструменты. Его поверхность должна быть покрыта плотным материалом, чтобы исключить скатывание маленьких деталей. Для обеспечения комфортных условий работы сотрудника стол должен соответствовать определённым параметрам [18]:

- его высота должна позволять свободно сидеть;
- ширина стола должна быть не менее 1000 мм;
- пространство для ног должно быть высотой не менее 600 мм, шириной – не менее 500 мм.

Монитор должен располагаться напротив специалиста на расстоянии около 450 мм, чтобы минимизировать нагрузку на глаза. Клавиатура должна находиться на поверхности высотой примерно 650 мм. Принтеры, сканеры и другую оргтехнику рекомендуется устанавливать справа либо слева от сотрудника, чтобы он мог в любой момент беспрепятственно взаимодействовать с ней.

Основное время при работе инженер-электроник проводит сидя. Его кресло либо стул должны быть прочными. Форма сиденья должна позволять сидеть прямо с помощью опирания на спинку, а поверхность должна быть выполнена из мягкого материала. Высота кресла или стула над полом должна быть не менее 450 мм.

К измерительным приборам относятся мультиметры, осциллографы и другие устройства, позволяющие определить значение каких-либо параметров у техники. Их часто располагают на полке над столом, либо в шкафу с инструментами, где всё распределено по пластмассовым контейнерам.

Паяльник и паяльные принадлежности, например припой, флюс, должны храниться в специально отведённом месте, например в выдвижном ящике. Для паяльника есть подставка, на которой он нагревается и остывает, чтобы специалист не получил ожог. Во время пайки должно быть включено искусственное освещение, а также должна работать вентиляция для ликвидации испаряемых вредных веществ.

При пайке оловом или свинцом вытяжка должна обеспечивать скорость движения воздуха не менее 0,6 м/с. Воздухоприёмные устройства должны быть расположены рядом с местом пайки.

Рассчитать необходимый воздухообмен при пайке припоем с канифолью можно по формуле:

$$G = \frac{K \cdot B}{(q_2 - q_1)}, \quad (34)$$

где К = 1,7;

В – количество свинца, выделяющегося на рабочем месте за 1 час, В = 0,45 мг/ч;

q_2 – концентрация свинца в удаляемом воздухе, принимается равной предельно допустимой, q_2 = 0,01 мг/м^3;

q_1 – концентрация свинца в приточном воздухе, составляет 30% от ПДК, q_1 = 0,003 мг/м^3.

Таким образом,

$$G = \frac{1{,}7 \cdot 0{,}45}{(0{,}01 - 0{,}003)} = 79 \text{ м}^3/ч.$$

Вентиляция осуществляется с помощью отсасывающей панели.

Чтобы найти расход воздуха при этом нужно использовать следующую формулу:

$$G_п = C \cdot Q_K^{\frac{1}{3}} \cdot (H + B)^{\frac{5}{3}}, \quad (35)$$

где C – коэффициент, зависящий от конструкции панели и ее расположения относительно источников тепла;

Q_K – конвективная составляющая источника тепла,

$Q_K = 0{,}25Q_{чел} + Q_п = 0{,}25 \cdot 77 + 9 = 28{,}25$ Вт;

H – расстояние от верха плоскости источника до центра всасывающих отверстий панели, H = 0,38 м;

B – ширина источника тепла, B = 0,2 м.

Коэффициент C применяется равным:

$$C = 228 \cdot F \cdot \left(\frac{I}{(H+B)}\right)^{\frac{2}{3}}, \quad (36)$$

где F – площадь источника тепловыделения, F= 0,4 м^2;

I – максимальное удаление источника от панели, I = 0,9 м.

$$C = 228 \cdot 0{,}4 \cdot \left(\frac{0{,}9}{(0{,}38+0{,}2)}\right)^{\frac{2}{3}} = 122{,}2.$$

После подстановки соответствующих значений в формулу 35 было получено:

$$G_п = 122{,}2 \cdot 28{,}25^{\frac{1}{3}} \cdot (0{,}38 + 0{,}2)^{\frac{5}{3}} = 150{,}1 \text{ м}^3/\text{ч}.$$

$G_п > G$, следовательно, использовать отсасывающую панель можно.

4.3 Утилизация компьютерной и оргтехники

В наше время компьютерная и оргтехника используются с каждым днём все чаще. На замену устаревшей или нерабочей техники приходит новая. Вопрос утилизации списанного оборудования крайне важен для общества, так как он связан с экологией.

Приборы запрещено выбрасывать на свалку, это контролируется Федеральным законом от 24 июня 1998 г. N 89-ФЗ "Об отходах производства и потребления". Неисправная техника содержит в себе различные драгоценные металлы, а также химически вредные элементы, например, свинец, которые под влиянием внешних условий проявляют свои токсичные свойства [19]. Именно поэтому её требуется правильно утилизировать.

Согласно [20] в переработке нуждаются компьютеры, факсы, принтеры и другая техника.

Перед утилизацией её необходимо списать. Составляется акт, в котором устанавливается, что оборудование не пригодно, и его можно вычесть из имущества предприятия. Это делается для того, чтобы не было ошибок в бухгалтерской отчётности и штрафов.

Чтобы обеспечить переработку пластмассы и металлов, из которых изготовлено оборудование, для дальнейшего использования в производстве, обычно обращаются к организациям, решающим эти вопросы. Они обладают всеми необходимыми лицензиями и техническими средствами для проведения утилизации. В их состав входит высококвалифицированный персонал, способный качественно выполнить работу. Сначала подписывается договор на утилизацию. Затем технику перерабатывают с помощью следующих этапов:

- транспортировка её с организации;
- демонтаж техники для распределения по материалам;
- вывоз сырья на производство для повторного использования;
- передача деталей, содержащих драгметаллы, на аффинаж;
- устранение отходов, представляющих опасность;
- заключение акта о завершении утилизации.

В итоге заказчик получает все документы о проведённой переработке.

4.4 Выводы по разделу «Охрана труда и промышленная экология»

В данном разделе были рассмотрены вопросы, связанные с организацией рабочего места специалиста, расчётом технических средств для обеспечения безопасности при работе и утилизацией техники.

На инженера-электроника во время работы воздействует большое количество негативных факторов. Для обеспечения его безопасности применяются различные средства защиты, позволяющие полностью либо частично ослабить их влияние.

Списанную технику необходимо правильно утилизировать. Утилизация позволит избежать экологической проблемы, а материалы, применяемые в оборудовании, будут повторно использованы на производстве.

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы был разработан проект самобалансирующего робота с микроконтроллерным управлением, и были получены следующие результаты:

- выполнен анализ предметной области исследования, установлено, что на рынке представлено не так много самобалансирующих роботов, но они отличаются как по функциональным возможностям, так и внешнему виду, а также имеют разные цены;
- разработаны технические требования к проектируемому устройству, в которых были обозначены его функции, взаимодействие с мобильным приложением, требования безопасности, условия хранения и эксплуатации, а также выбрана элементная база, позволяющая реализовать представленные функциональные возможности;
- разработана аппаратная часть устройства, включая – проектирование принципиальной схемы, печатной платы и деталей корпуса, а также листинг программы и блок-схема её работы;
- рассмотрены вопросы экологической безопасности и охраны труда, а именно проанализированы вредные и опасные факторы, влияющие на инженера-электроника, индивидуальные и коллективные средства защиты от них, способы утилизации компьютерной и оргтехники, также проведён расчёт технических средств обеспечения безопасности труда на рабочем месте специалиста;
- выполнены расчёты себестоимости и эффективности проекта, в ходе которых были определены показатели окупаемости и доходности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Нидилько Марина Васильевна Назначение и область применения, архитектура микроконтроллера // Достижения науки и образования. 2018. №6 (28). URL: https://cyberleninka.ru/article/n/naznachenie-i-oblast-primeneniya-arhitektura-mikrokontrollera (дата обращения: 19.02.2022)

2. Микроконтроллер | Центральный процессор (CPU) [Электронный ресурс] URL: https://auto-metal.ru/spravka/mikrokontroller.html (дата обращения: 19.02.2022)

3. Семейства микроконтроллеров (Лекция) | МК [Электронный ресурс] URL: http://mc-plc.ru/mk/semeystva-mikrokontrollerov.htm (дата обращения: 19.02.2022)

4. AVR. Учебный курс. Архитектура. | Электроника для всех [Электронный ресурс] URL: http://easyelectronics.ru/avr-uchebnyj-kurs-arxitektura.html (дата обращения 19.02.2022)

5. Что такое компилятор? | Hexlet Guides [Электронный ресурс] URL: https://guides.hexlet.io/compiler/ (дата обращения: 19.02.2022)

6. История создания робототехники. Робототехника: история и современность [Электронный ресурс] URL: https://knhb.ru/istoriya-sozdaniya-robototehniki-robototehnika-istoriya-i-sovremennost.html (дата обращения: 20.02.2022)

7. Бекзодбек Салимович Саидов Роль робототехники в жизни человека // Science and Education. 2021. №8. URL: https://cyberleninka.ru/article/n/rol-robototehniki-v-zhizni-cheloveka (дата обращения: 20.02.2022)

8. Овчинников Валентин Васильевич, Батанов Александр Федорович, Мингалеев Салават Галимджанович РОБОТЫ В ЧЕРНОБЫЛЕ // Технологии гражданской безопасности. 2019. №4 (62). URL: https://cyberleninka.ru/article/n/roboty-v-chernobyle (дата обращения: 20.02.2022)

9. Константин Кузнецов Умная фабрика: как автомобили собираются без людей // Популярная механика. — 2017. — № 5. — С. 86-87. —

URL: http://www.popmech.ru/business-news/334232-umnaya-fabrika-kak-avtomobili-sobirayutsya-bez-lyudey/ (дата обращения: 20.02.2022)
10. Акименко Татьяна Алексеевна, Кузнецова Татьяна Рудольфовна ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ ПРОМЫШЛЕННЫХ РОБОТОВ // Известия ТулГУ. Технические науки. 2019. №10. URL: https://cyberleninka.ru/article/n/osobennosti-proektirovaniya-promyshlennyh-robotov (дата обращения: 20.02.2022)
11. 15 различных типов роботов [Электронный ресурс] URL: https://new-science.ru/15-razlichnyh-tipov-robotov/ (дата обращения: 20.02.2022)
12. Балансировка – Библиотека [Электронный ресурс] URL: https://prmeh.ru/tech/library/balancing_about/ (дата обращения: 20.02.2022)
13. Madgwick S. An efficient orientation filter for inertial and inertial/magnetic sensor arrays / S. Madgwick //Report x-io and University of Bristol (UK). – 2010. – Т. 25. – pp. 32 (дата обращения: 20.02.2022)
14. Андреев Дмитрий Васильевич, Лукачевский Ньургун Иннокентьевич Охрана труда на производстве // Московский экономический журнал. 2018. №5 (1). URL: https://cyberleninka.ru/article/n/ohrana-truda-na-proizvodstve (дата обращения: 01.03.2022)
15. ГОСТ 12.1.0.003-74 ССБТ «Опасные и вредные производственные факторы. Классификация»
16. ГОСТ Р 55710-2013 «Освещение рабочих мест внутри зданий. Нормы и методы измерений»
17. Ходаков Никита Дмитриевич, Шамсутдинов Амир Борисович, Буданов Борис Владимирович ИСПОЛЬЗОВАНИЕ СРЕДСТВ ИНДИВИДУАЛЬНОЙ ЗАЩИТЫ // E-Scio. 2021. №4 (55). URL: https://cyberleninka.ru/article/n/ispolzovanie-sredstv-individualnoy-zaschity (дата обращения: 24.03.2022)
18. ГОСТ 12.2.032-78 «Система стандартов безопасности труда. Рабочее место при выполнении работ сидя»

19. Осолодкова Е.В. Проблемы утилизации компьютерной техники // Инновационная наука. 2018. №11. URL: https://cyberleninka.ru/article/n/problemy-utilizatsii-kompyuternoy-tehniki (дата обращения: 25.03.2022)
20. О порядке ведения государственного кадастра отходов и проведения паспортизации опасных отходов [Электронный ресурс] URL: http://pravo.gov.ru/proxy/ips/?doc_itself=&backlink=1&nd=102068056&page=1&rdk=1#I0 (дата обращения: 25.03.2022)

ПРИЛОЖЕНИЕ А

Техническое задание

СОГЛАСОВАНО
Преподаватель
_____/ Кочнев П.В.
Личная подпись /Расшифровка
«___»_____ 2022 г.

УТВЕРЖДАЮ
Председатель ПЦК
_____/ Апталаев М.Н.
Личная подпись /Расшифровка
«___»_____ 2022г.

Техническое задание

на разработку «проекта самобалансирующего робота с микроконтроллерным управлением»

На 4 листах

СОГЛАСОВАНО
Руководитель разработки
Кочнев П.В.
_____/_____/
«___»_____2022г.
Исполнитель
Шуман А.С.
_____/_____/
«___»_____2022 г.

1. Общие сведения

1.1. Наименование устройства

1.1.1. Полное наименование устройства

Полное наименование устройства: «Самобалансирующий робот»

1.1.2. Краткое наименование устройства

Краткое наименование: Комплекс

1.2. Шифр темы

Шифр темы: СРМ

2. Назначение и цели создания устройства

2.1. Назначение устройства

Самобалансирующий робот предназначен для использования в качестве прототипа на предприятиях, разрабатывающих такие устройства, как гироскутеры, и для наглядного представления систем управления и балансировки при езде на ровной поверхности.

2.2. Цели создания системы

Самобалансирующий робот создаётся с целью получения практических знаний о балансировке.

В результате создания комплекса информация, используемая при разработке, может использоваться для дальнейших работ, связанных с самобалансировкой.

3. Требования к устройству

3.1. Требования к устройству в целом

3.1.1. Требования к структуре и функционированию устройства

Комплекс должен быть централизован и связан с мобильным приложением с помощью Bluetooth. При срабатывании определённой команды с мобильного устройства сигнал подаётся на микроконтроллер самобалансирующего робота. Он соединён со всеми датчиками и исполнительными устройствами, от которых ему также приходят сигналы. В системе имеется блок питания, основанный на 4 литий-ионных аккумуляторах типа 18650. Комплекс имеет 2 режима работы:

1) основной, в котором устройство поддерживает режим работы и выполняет все свои функции;

2) выключенный, в котором устройство отключено.

3.1.2. Показатели назначения

В мобильном приложении должно быть видно подключение к системе, а также, если она включена, на микроконтроллере должен гореть светодиод. На смартфоне в приложении должна отображаться и регулироваться скорость робота в виде перемещаемого ползунка, а кнопки, отвечающие за выполнение определённых команд, при нажатии должны становиться активными.

3.1.3. Требования к надёжности

Уровень надёжности должен достигаться согласованным применением программно-аппаратных средств и качественным выполнением поставленной задачи – построения самобалансирующего робота. Надёжность обеспечивается за счёт соблюдения правил эксплуатации и технического обслуживания программно-аппаратных средств.

3.1.4. Требования безопасности

К самобалансирующему роботу предъявляется ряд требований для обеспечения безопасности:

- должно быть обеспечено соблюдение общих требований безопасности в соответствии с ГОСТ 12.2.003-91. «Общие требования безопасности» при обслуживании системы в процессе эксплуатации;
- включенное устройство должно находиться в защищённом от влаги месте;
- выполнять разборку и проверку соединения контактов на плате только при отключённом питании.

3.1.5. Требования к эргономике и технической эстетике

Должно быть обеспечено компактное размещение всех разъёмов и остальных элементов на плате устройства. В мобильном приложении должен быть обеспечен удобный для конечного пользователя интерфейс.

3.1.6. Требования к эксплуатации и хранению

Для эксплуатации и хранения самобалансирующего робота в помещении должна быть обеспечена температура в установленных пределах – от 0 до +35 °C. При этом он не должен располагаться близко к системам отопления. Также для обеспечения работоспособного состояния устройства относительная влажность места хранения должна быть больше 35%, но ниже 80%.

3.1.7. Требования к защите информации от несанкционированного доступа

Не предъявляются.

3.2. Требования к функциям

Самобалансирующий робот должен включаться с помощью кнопочного переключателя. В нём будут представлены следующие функциональные блоки:

- блок измерения угла наклона и балансировки;
- блок определения расстояния до преграды;
- блок питания;
- блок связи с устройством.

Данная система должна выполнять следующие функции:

- самобалансировать на месте и при езде с помощью гиродатчика и колёс;
- измерять расстояние до препятствия с использованием ультразвукового датчика, чтобы развернуться;
- подключаться по Bluetooth к приложению на смартфоне для управления.

Для защиты от падений в устройстве должна быть дуга, на которую оно будет опираться. Самобалансирующий робот с помощью своей подвижной руки должен подниматься с поверхности.

3.3. Требования к видам обеспечения

3.3.1. Требования к математическому обеспечению

Не предъявляются.

3.3.2. Требования к информационному обеспечению

Предъявляются требования:

1) к информационному обмену между компонентами системы;

2) к контролю, хранению данных.

3.3.3. Требования к лингвистическому обеспечению

При реализации системы должна применяться программа, написанная на Си-подобном языке.

3.3.4. Требования к программному обеспечению

Перечень программных средств, используемых при разработке системы:

- EasyEDA;
- ArduinoIDE;
- FreeCAD.

3.3.5. Требования к составу технических средств

В состав технических средств должны входить:

- микроконтроллер;
- два шаговых двигателя;
- два драйвера шаговых двигателей;
- датчик ультразвукового излучения;
- гироскоп-акселерометр;
- сервомотор;
- два понижающих DC-DC преобразователя.

3.3.6. Требования к метрологическому обеспечению

Не предъявляются.

3.3.7. Требования к методическому обеспечению

Не предъявляются.

4. Требования к программной документации

В ходе разработки программы должны быть подготовлены:

- код программы;
- технико-экономическое обоснование.

ПРИЛОЖЕНИЕ Б – Листинг управляющей программы

Код основного файла:

```
#include <BluetoothSerial.h>
#if     !defined(CONFIG_BT_ENABLED)  ||
!defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;  //создаём Bluetooth-порт

#include <Wire.h>  //Связь по I2C
#include <Adafruit_MPU6050.h>

// Подключаем внутренние программные файлы.
#include "ZnachPerem.h"  //Переменные и константы
#include "MadgwickAHRS.h" //Фильтр Маджвика
#include "OpisMotor.h"  //Описание моторов
#include "NastroikaSonara.h" // Настройка сонара
#include "ServoRuka.h" //Обмен командами
#include "Prerivaniya.h"  //Работа с прерываниями
#include "RaschetShag.h" //Пересчет шагов
#include "ObmenBL.h" //Обмен командами со смартфоном
//IBusBM IBus; // IBus object
//HardwareSerial FSkySerial(2); //rx in 16gpio

Adafruit_MPU6050 mpu;
sensors_event_t a, g, temper;

#include "Hranenie.h"  //Хранение настроек
```

```
//Задаем GPIO для Wire
const int _SDA = 21;
const int _SCL = 22;

void setup()
{
  servo_hand_setup();  //функция инициализации сервомашинки руки
  sonar_setup();  //функция УЗ датчика
  setup_motor_system();     //функция инициализации моторов (выключаем моторы перемещения)

  Wire.begin(_SDA, _SCL);
  Serial.begin(115200);

  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

mpu.setGyroRange(MPU6050_RANGE_500_DEG);//MPU6050_RANGE_1000_DEG
mpu.setFilterBandwidth(MPU6050_BAND_260_HZ);
delay(1000);
```

```
Calc_CompensatorZ(5000); //остановка работы программы на 5000 миллисекунд для расчёта смещения нуля гироскопа (для каждой оси)
  arming = false;   //включение моторов
  timer_setup();
  Serial.print("start2");
  Serial.print("start3=");
  Serial.println(portTICK_PERIOD_MS);//configMAX_PRIORITIES);

  mpu.getEvent(&a, &g, &temper);
  commandRIGHT_WEEL_SPEED = 0;  //скорость правого колеса
  commandLEFT_WEEL_SPEED = 0;   //скорость левого колеса
  OldTime =  micros();  //присваиваем количество микросекунд с начала выполнения программы
  Pereschet = millis() + 3000;  //даём 3 секунды на предстартовую подготовку
  SerialBT.begin("ESP32BALROBOT"); //Bluetooth имя робота
}

uint32_t Dtime = 5000;  //время в микросекундах между опросами
uint32_t start_period = 0;
double dta = 0;

void loop()
{
 BT_input();  //функция приёма с BT
  micros_1 = micros();
  if ((micros_1 - OldTime) < 5000) return;   //пересчёт балансировки каждые 5 миллисекунд

  Dtime = micros_1 - OldTime;
```

```
  Dtime = constrain(Dtime, 4000, 10000);  //установим пределы для значения (если больше или меньше, то примет эти значения)
  Dt = double(Dtime) * 0.000001;  //переводим время между опросами из микросекунд в секунды (0,005 секунд - время от предыдущего опроса)
  OldTime = micros_1;

  mpu.getEvent(&a, &g, &temper);
  //Фильтр Маджвика
  MadgwickAHRSupdateIMU(Dt, g.gyro.x - CompensatorX , g.gyro.y - CompensatorY, g.gyro.z - CompensatorZ, a.acceleration.x, a.acceleration.y, a.acceleration.z);
  quat[0] = q0; quat[1] = q1; quat[2] = q2; quat[3] = q3;
  if ((millis() < Pereschet) && (!avtostart)) return;  //если прошло меньше 3 секунд при включении робота и функция avtostart (поднятие рукой с помощью BT) активна, то возвращаемся
  else avtostart = true;  //иначе рука срабатывает и робот поднимается

  quat2Euler(&quat[0], &imu[0]);  // преобразуем кватернион в углы Эйлера
  if (fabs(imu[1]) > 0.75) arming = false;  //если угол наклона робота больше указанного в радианах (или 42,9 градуса), то моторы отключены
  else if ((fabs(imu[1]) < 0.55) && (!arming)){ arming = true; }  //если угол наклона робота меньше указанного в радианах (или 31,5 градуса), то моторы можно включить

  opros_sys_EN_motors(); //Отключаем моторы в случае необходимости

  if (STEPER_EN_level == MOTORS_ON)  //Расчёт системы балансировки робота
  {
    Error_ang = imu[1] - base_ang; //Ошибка угла наклона в радианах
```

```
  dError_ang = (Error_ang - OldError_ang) / Dt; //Скорость изменения ошибки угла наклона (угловая скорость изменения ошибки)
  Error_speed = Sbo - CommandSpeed; //Ошибка скорости в шагах
  dError_speed = (Error_speed - OldError_speed) / Dt; //Скорость изменения ошибки скорости
  iError_speed += Error_speed * Dt * Ki_s; //Накопленная - интегральная ошибка скорости (ошибка пройдённого расстояния)

  iError_speed = constrain(iError_speed, -2 * MAX_WEEL_SPEED, 2 * MAX_WEEL_SPEED);

  New_Speed = Error_ang * Kp_a + dError_ang * Kd_a + Error_speed * Kp_s + dError_speed * Kd_s + iError_speed;  //новая скорость
  OldError_ang = Error_ang;
  OldError_speed = Error_speed;
  commandRIGHT_WEEL_SPEED = constrain(New_Speed - Turn, commandRIGHT_WEEL_SPEED - Dt * ACCELERATION_WEEL, commandRIGHT_WEEL_SPEED + Dt * ACCELERATION_WEEL); //пересчитываем реальную скорость колёс, ограничивая скорость поворота и новую возможную скорость
  commandLEFT_WEEL_SPEED = constrain(New_Speed + Turn, commandLEFT_WEEL_SPEED - Dt * ACCELERATION_WEEL, commandLEFT_WEEL_SPEED + Dt * ACCELERATION_WEEL);
  OldSpeed = (commandLEFT_WEEL_SPEED + commandRIGHT_WEEL_SPEED) * 0.5 ;  //средняя скорость, чтобы не выходить за пределы максимальной скорости
 }
 else
 {
  New_Speed = OldSpeed = 0;
```

```
    commandRIGHT_WEEL_SPEED = 0;
    commandLEFT_WEEL_SPEED = 0;
  }
  CalcSpeedAndPosition(Dt);  //расчёт длины шага
}
```

Код внутреннего файла Hranenie.h:

```
#include <Preferences.h>   //отвечает за запись/чтение информации из/в встроенную в контроллер ESP32 FLASH память
Preferences preferences;
//=================================================
//=================================================
bool write_to_flash()
{
  //Сохранение в хранилище
  preferences.begin("robo_steps", false);
  preferences.clear();
  preferences.putFloat("CompensatorZ", CompensatorZ);
  preferences.putFloat("CompensatorX", CompensatorX);
  preferences.putFloat("CompensatorY", CompensatorY);
  preferences.end();
  return true;
}
//=================================================
//=================================================
//=================================================
void Calc_CompensatorZ(float mill_sec)  //останавливает работу программы на заданное количество миллисекунд, в течение которых робот рассчитывает смещение нуля гироскопа (для каждой оси)
{
  uint32_t ms = mill_sec;
```

```
  double i = 0;  //количество опросов
  CompensatorZ = 0;
  CompensatorX = 0;
  CompensatorY = 0;
  uint32_t  timer = millis();
  uint32_t endtime = millis() + ms;
  while (endtime > timer) {
    timer = millis();
    mpu.getEvent(&a, &g, &temper);

    CompensatorZ += g.gyro.z;
    CompensatorX += g.gyro.x;
    CompensatorY += g.gyro.y;
    delay(2);
    i++;
  }
  CompensatorZ /= i;
  CompensatorX /= i;
  CompensatorY /= i;
//  write_to_flash();
}
//=================================================
//=================================================
//=================================================
bool read_from_flash()
{
  //Открываем хранилище на чтение
  preferences.begin("robo_steps", false);
  CompensatorZ= preferences.getDouble("CompensatorZ",0);
  CompensatorX= preferences.getDouble("CompensatorX",0);
```

```
  CompensatorY= preferences.getDouble("CompensatorY",0);
  preferences.end();
  return true;
}
```

Код внутреннего файла MotPin.h:

```
#define GPIO_LEFT_WEEL_STEP 16// //Пин левого шага шагового мотора
#define GPIO_LEFT_WEEL_DIR 19// //Пин направления левого шага

#define GPIO_RIGHT_WEEL_STEP 15// //Пин правого шага шагового мотора
#define GPIO_RIGHT_WEEL_DIR 13// ////Пин направления правого шага

#define GPIO_MOTORS_EN 5 //Пин включения моторов

// Константы направления вращения приводов

bool DIR_LEFT_FORWARD = true;  //левый - вперёд
bool DIR_LEFT_BACKWARD = !DIR_LEFT_FORWARD;

bool DIR_RIGHT_FORWARD = false;  //правый - вперёд
bool DIR_RIGHT_BACKWARD = !DIR_RIGHT_FORWARD;

const bool MOTORS_ON = false;  //включение двигателей
const bool MOTORS_OFF = !MOTORS_ON;
```

Код внутреннего файла OpisMotor.h:

```
#include "MotPin.h"
bool STEPER_EN_level = false;   //флаг, обозначающий, что моторы уже включились

void setup_motor_system()   //Функция инициализации управления моторами (выключены)
```

```
{
  STEPER_EN_level = MOTORS_OFF;
  pinMode(GPIO_LEFT_WEEL_STEP, OUTPUT);
  pinMode(GPIO_LEFT_WEEL_DIR, OUTPUT);
  pinMode(GPIO_RIGHT_WEEL_STEP, OUTPUT);
  pinMode(GPIO_RIGHT_WEEL_DIR, OUTPUT);
  pinMode(GPIO_MOTORS_EN, OUTPUT);

  digitalWrite(GPIO_LEFT_WEEL_STEP, false);
  digitalWrite(GPIO_LEFT_WEEL_DIR, DIR_LEFT_FORWARD);

  digitalWrite(GPIO_RIGHT_WEEL_STEP, false);
  digitalWrite(GPIO_RIGHT_WEEL_DIR, DIR_RIGHT_FORWARD);

  digitalWrite(GPIO_MOTORS_EN, STEPER_EN_level);
}
```

Код внутреннего файла NastroikaSonara.h:

```
//Создаем указатель на таймер
//генерации шагов для моторов робота
hw_timer_t * sonar_timer_t = NULL;

//этапы анализа УЗ датчиком расстояния
#define FLAG_SONAR_START1 1  //переводим лимит из сантиметров в микросекунды и генерируем импульс
#define FLAG_SONAR_START2 2  //ждём отражения импульса
#define FLAG_SONAR_REGread 3  //пересчитываем время в расстояние
#define FLAG_SONAR_STOP 4  //получаем значение в миллиметрах и цикл завершается

//Номер GPIO сонара.
```

```
#define SONAR_GPIO 17
#define SONAR_TRIG  27

bool sonarAVTO=false;  //УЗ датчик выключен
const uint32_t long_mm_GPIO = 1 << SONAR_GPIO;



volatile uint32_t dist_mm;  //рассчитываем дистанцию в миллиметрах


//= Измерение расстояния ============================
uint32_t registr_in;

uint32_t Limit = 55;  //лимит измерения в сантиметрах



void sonar_setup()  //инициализация УЗ датчика
{
  //=Режим пинов/портов ================================
  pinMode(SONAR_TRIG, OUTPUT);
  pinMode(SONAR_GPIO, INPUT);
  // Создаем таймер

  // Прикрепляем функциию обработчики

  //Прерывание повторяется каждые 10 микросекунд

  //Старт таймера, теперь прерывания будут генерироваться
```

```
sonarAVTO=true; //Включаем
}
```

Код внутреннего файла ObmenBL.h:

```
//переменные для отслеживания связи со смартфоном для остановки робота
uint32_t MOVE_PERIOD = 300;
uint32_t   time_start_move = 0;

//константы для задания максимальной скорости робота
const int32_t  MOVSPEED0 = (MAX_WEEL_SPEED / 100) * 3;
const int32_t  MOVSPEED1 = (MAX_WEEL_SPEED / 100) * 5;
const int32_t  MOVSPEED2 = (MAX_WEEL_SPEED / 100) * 10;
const int32_t  MOVSPEED3 = (MAX_WEEL_SPEED / 100) * 15;
const int32_t  MOVSPEED4 = (MAX_WEEL_SPEED / 100) * 20;
const int32_t  MOVSPEED5 = (MAX_WEEL_SPEED / 100) * 25;
const int32_t  MOVSPEED6 = (MAX_WEEL_SPEED / 100) * 30;
const int32_t  MOVSPEED7 = (MAX_WEEL_SPEED / 100) * 35;
const int32_t  MOVSPEED8 = (MAX_WEEL_SPEED / 100) * 40;
const int32_t  MOVSPEED9 = (MAX_WEEL_SPEED / 100) * 45;
const int32_t  MOVSPEED10 = (MAX_WEEL_SPEED / 100) * 50;
int32_t ASPEED = MOVSPEED2;     //текущая максимальная скорость управляемого движения
int32_t DeltaSpeed_100sec = 10000;   //приращение скорости (каждые 50 миллисекунд, когда приходят команды) для плавного изменения скорости


bool BT_input()  //функция приёма с BT
{
  static bool local_shot_dist = false;  //флаг того, что дистанция до препятствия мала
  static char bt_input;  //символы для BT
```

```
 if (SerialBT.available())  //проверяем наличие команд от смартфона
 {
  bt_input = (char)SerialBT.read();  //читаем из буфера символ
 }
 else  //если связи нет, то обнуляем скорость
 {
  if (time_start_move < millis())
  {
   // CommandSpeed = 0; //Обнуляем скорость управляемого движения.
   if (CommandSpeed > 0) CommandSpeed = constrain(CommandSpeed - DeltaSpeed_100sec, 0, CommandSpeed); //Обнуляем скорость управляемого движения.
   if (CommandSpeed < 0) CommandSpeed = constrain(CommandSpeed + DeltaSpeed_100sec, CommandSpeed, 0); //Обнуляем скорость управляемого движения.
    Turn = 0;  //поворот
  }
  return false;
 }
 time_start_move = millis() + MOVE_PERIOD;

 if ((dist_mm < 450) && ((bt_input == 'F') || (bt_input == 'G') || (bt_input == 'I'))) //если дистанция до препятствия меньше 450 мм и движемся вперёд или поворачиваем влево или вправо при движении вперёд
 {
  bt_input = 'L';  //поворачиваем налево на месте
  CommandSpeed = -ASPEED;
  Turn = -ASPEED;    //-, потому что по гироскопу поворот влево отрицательный
```

```
  bt_input = 'e';

 }
 switch (bt_input)  //оператор выбора команд
 {
  case 'S': // остановка
    //      CommandSpeed = 0; //Обнуляем скорость управляемого движения.
    if (CommandSpeed > 0) CommandSpeed = constrain(CommandSpeed - DeltaSpeed_100sec, 0, CommandSpeed); //Обнуляем скорость управляемого движения.
    if (CommandSpeed < 0) CommandSpeed = constrain(CommandSpeed + DeltaSpeed_100sec, CommandSpeed, 0); //Обнуляем скорость управляемого движения.
    if (local_shot_dist)
    {
     Turn = -ASPEED / 2;
    }
    else
    {
     Turn = 0;
    }
    break;
  case 'F': // вперед

    if (!local_shot_dist)
    {
     Turn = 0;
     CommandSpeed = constrain(CommandSpeed + DeltaSpeed_100sec, CommandSpeed, ASPEED);      //ограничиваем скорость приращением (поступает много команд)
```

```
    }
    else
    {
     Turn = -ASPEED * 2;
     //Обнуляем скорость управляемого движения.
     if (CommandSpeed > 0) CommandSpeed = constrain(CommandSpeed - DeltaSpeed_100sec, 0, CommandSpeed);
     if (CommandSpeed < 0) CommandSpeed = constrain(CommandSpeed + DeltaSpeed_100sec, CommandSpeed, 0);
    }
    break;
  case 'B': //назад
    CommandSpeed = constrain(CommandSpeed - DeltaSpeed_100sec, -ASPEED, CommandSpeed);
    Turn = 0;
    break;
  case 'L':  //Влево на месте
    Turn = -ASPEED / 2;  //-, потому что по гироскопу поворот влево отрицательный
    //Обнуляем скорость управляемого движения.
    if (CommandSpeed > 0) CommandSpeed = constrain(CommandSpeed - DeltaSpeed_100sec, 0, CommandSpeed);
    if (CommandSpeed < 0) CommandSpeed = constrain(CommandSpeed + DeltaSpeed_100sec, CommandSpeed, 0);

    break;
  case 'R':  //Вправо на месте
    Turn = ASPEED / 2;  //поворот вправо по гироскопу положительный (поэтому просто ASPEED)
    //Обнуляем скорость управляемого движения.
```

```
    if (CommandSpeed > 0) CommandSpeed = constrain(CommandSpeed -
DeltaSpeed_100sec, 0, CommandSpeed);
    if (CommandSpeed < 0) CommandSpeed = constrain(CommandSpeed +
DeltaSpeed_100sec, CommandSpeed, 0);
     break;
   case 'G':  //Влево вперед
     CommandSpeed = constrain(CommandSpeed + DeltaSpeed_100sec,
CommandSpeed, ASPEED);
     Turn = -ASPEED / 2;
     break;
   case 'I':  //Вправо вперед
     //CommandSpeed = ASPEED; //
     CommandSpeed = constrain(CommandSpeed + DeltaSpeed_100sec,
CommandSpeed, ASPEED);
     Turn = ASPEED / 2;
     break;
   case 'H':  //Вправо
     CommandSpeed = constrain(CommandSpeed - DeltaSpeed_100sec, -ASPEED,
CommandSpeed);
     Turn = ASPEED / 2;
     break;
   case 'J':  //Влево
     CommandSpeed = constrain(CommandSpeed - DeltaSpeed_100sec, -ASPEED,
CommandSpeed);
     Turn = -ASPEED / 2;
     break;
   case '0':
     ASPEED = MOVSPEED0;
     break;
   // Скорость 10%
```

```
case '1':
  ASPEED = MOVSPEED1;
  break;
// Скорость 20%
case '2':
  ASPEED = MOVSPEED2;
  break;
// Скорость 30%
case '3':
  ASPEED = MOVSPEED3;
  break;
// Скорость 40%
case '4':
  ASPEED = MOVSPEED4;
  break;
// Скорость 50%
case '5':
  ASPEED = MOVSPEED5;
  break;
// Скорость 60%
case '6':
  ASPEED = MOVSPEED6;
  break;
// Скорость 70%
case '7':
  ASPEED = MOVSPEED7;
  break;
case '8':
  // Скорость 80%
  ASPEED = MOVSPEED8;
```

```
    break;
  // Скорость 90%
  case '9':
    ASPEED = MOVSPEED9;
    break;
  // Скорость 100%
  case 'q':
    ASPEED = MOVSPEED10;
    break;
  case 'V':  //поднятие робота с помощью руки
    if (avtostart) {
      handACTIV = true;
      servo_hand_bottom(imu[1]);     //проверяет, в какую сторону наклонен робот, и поворачивает руку в попытке его поднять
    }
    break;
  case 'v':  //рука в исходное положение
    if (avtostart) {
      handACTIV = true;
      servo_hand_up();   //поднятие руки в исходное положение, если робот поднят
    }
    break;
  case 'X':
    break;
  case 'x':
    break;
  default:
    break;
 }
```

```
  return true;
}
```

Код внутреннего файла Prerivaniya.h:

```
//Создаем указатель на таймер
//генерации шагов для моторов робота
hw_timer_t * Timer = NULL;

// Объявляем переменные, хранящие текущее
// состояние уровня сигнала на GPIO шагов для моторов
// Флаг получения новой скорости
volatile bool newSpeedflag_ = false;

//Маска выключения высокого состояния шагов всех моторов
//Маски высокого состояния шагов для каждого мотора
//Маски подъема сигналов на GPIO шагов моторов
const uint32_t STEP_mask_on_WEEL_LEFT_ = uint32_t(1) << GPIO_LEFT_WEEL_STEP;
const uint32_t STEP_mask_on_WEEL_RIGHT_ = uint32_t(1) << GPIO_RIGHT_WEEL_STEP;
const uint32_t off_step = 0xFFFFFFFF ^ ( STEP_mask_on_WEEL_LEFT_ | STEP_mask_on_WEEL_RIGHT_);

uint32_t sost;  //составная маска для записи в регистр состояния GPIO шагов
//Направление шагов

bool DIR_tec_WEEL_LEFT_ = DIR_LEFT_FORWARD;
bool DIR_tec_WEEL_RIGHT_ = DIR_RIGHT_FORWARD;
// Новое время - длительность шага в 10микСек
uint32_t newcounter_do_step_WEEL_LEFT_ = 100;
uint32_t newcounter_do_step_WEEL_RIGHT_ = 100;
```

```
//Установленное время - длительность шага в 10микСек
uint32_t  counter_do_step_WEEL_LEFT_ = 100;
uint32_t  counter_do_step_WEEL_RIGHT_ = 100;

//Пройденное время в 10х мк.сек

//Прошедшее время текущего шага в 10микСек
int32_t counter_WEEL_LEFT_ = 0;
//int32_t counter_ROT_LEFT_ = 0;
int32_t counter_WEEL_RIGHT_ = 0;
//int32_t counter_ROT_RIGHT_ = 0;
bool old_sost = false;

const uint32_t half_step_ = 2; //пусть будет на 2 такте шага (опускается сигнал шага)

uint32_t LONG_STEP_WEEL_LEFT_;
uint32_t LONG_STEP_WEEL_LEFT;
bool dir_STEP_WEEL_LEFT_;
bool dir_STEP_WEEL_LEFT;
int32_t _STOP_POINT_WEEL_LEFT;
bool True_speed_WEEL_LEFT_;
bool True_speed_WEEL_LEFT;

uint32_t LONG_STEP_WEEL_RIGHT_;
uint32_t LONG_STEP_WEEL_RIGHT;
bool dir_STEP_WEEL_RIGHT_;
bool dir_STEP_WEEL_RIGHT;
int32_t _STOP_POINT_WEEL_RIGHT;
```

```
bool True_speed_WEEL_RIGHT_;
bool True_speed_WEEL_RIGHT;

bool True_speed_WEEL_RIGHT_Old=false;
bool True_speed_WEEL_LEFT_Old=false;

//обработчик прерываний (каждые 10 микросекунд)
void IRAM_ATTR core_timer() {
  if (newSpeedflag_)
  {
    LONG_STEP_WEEL_RIGHT_ = LONG_STEP_WEEL_RIGHT;
    dir_STEP_WEEL_RIGHT_ = dir_STEP_WEEL_RIGHT;
    if (LONG_STEP_WEEL_RIGHT_ == 0) { True_speed_WEEL_RIGHT_ =
false; True_speed_WEEL_RIGHT_Old=false;}
    else True_speed_WEEL_RIGHT_ = true;
    // = Если пришла новая скорость, шагов нет, и новая скорость не нулевая
    if ((True_speed_WEEL_RIGHT_)&&(!True_speed_WEEL_RIGHT_Old))
    {
      True_speed_WEEL_RIGHT_Old=true;
      counter_WEEL_RIGHT_ = half_step_;
      counter_do_step_WEEL_RIGHT_ = half_step_;
    }
    LONG_STEP_WEEL_LEFT_ = LONG_STEP_WEEL_LEFT;
    dir_STEP_WEEL_LEFT_ = dir_STEP_WEEL_LEFT;
    if (LONG_STEP_WEEL_LEFT_ == 0) { True_speed_WEEL_LEFT_ = false;
True_speed_WEEL_LEFT_Old=false;}
    else True_speed_WEEL_LEFT_ = true;
    // = Если пришла новая скорость, шагов нет, и новая скорость не нулевая
    if ((True_speed_WEEL_LEFT_)&&(!True_speed_WEEL_LEFT_Old))
    {
```

```
    True_speed_WEEL_LEFT_Old=true;
    counter_WEEL_LEFT_ = half_step_;
    counter_do_step_WEEL_LEFT_ = half_step_;
   }
   newSpeedflag_ = false;
  }
 //Шаг сделан рассчитываем параметры следующего шага
  if (counter_WEEL_RIGHT_ == half_step_) // если мы сделали сигнал шага и планируем следующий или стартуем со стоянки (проверка того, что текущий шаг находится в середине)
  {
   //Анализируем знак скорости
   //при необходимости изменяем направление
   //вращения мотора
   //Если изменились параметры скорости
   if ((LONG_STEP_WEEL_RIGHT_ != newcounter_do_step_WEEL_RIGHT_) || (DIR_tec_WEEL_RIGHT_ != dir_STEP_WEEL_RIGHT_))
   {
    if (DIR_tec_WEEL_RIGHT_ != dir_STEP_WEEL_RIGHT_)
    {
     DIR_tec_WEEL_RIGHT_ = dir_STEP_WEEL_RIGHT_;
     digitalWrite(GPIO_RIGHT_WEEL_DIR, DIR_tec_WEEL_RIGHT_);//DIR_RIGHT_WEEL_counterclockwise);
    }
    newcounter_do_step_WEEL_RIGHT_ = LONG_STEP_WEEL_RIGHT_;
   }
  }
  //Шаг сделан рассчитываем параметры следующего шага
```

```
  if (counter_WEEL_LEFT_ == half_step_) // если мы сделали сигнал шага и планируем следующий или стртуем со стоянки (проверка того, что текущий шаг находится в середине)
  {
   //Анализируем знак скорости
   //при необходимости изменяем направление
   //вращения мотора
   //Если изменились параметры скорости
   if ((LONG_STEP_WEEL_LEFT_ != newcounter_do_step_WEEL_LEFT_) || (DIR_tec_WEEL_LEFT_ != dir_STEP_WEEL_LEFT_))
   {
    if (DIR_tec_WEEL_LEFT_ != dir_STEP_WEEL_LEFT_)
    {
     DIR_tec_WEEL_LEFT_ = dir_STEP_WEEL_LEFT_;
     digitalWrite(GPIO_LEFT_WEEL_DIR, DIR_tec_WEEL_LEFT_);//DIR_RIGHT_WEEL_counterclockwise);
    }
    newcounter_do_step_WEEL_LEFT_ = LONG_STEP_WEEL_LEFT_;
   }
  }

  //обработка шагов

  //Опустить шаг для всех моторов
  // Заменяем множество опускающих команд на
  sost = 0;

  if (True_speed_WEEL_RIGHT_) //Если есть шаги
  {
```

```
  if (counter_WEEL_RIGHT_ >= counter_do_step_WEEL_RIGHT_)//Если шаг закончен
   {
    sost |= STEP_mask_on_WEEL_RIGHT_; //Поднять шаг -  пока только флаг
    //Счетчик длительности шага - обнуляем для начала нового шага
    counter_WEEL_RIGHT_ = 0;
   //  STEPS_counter_WEEL_RIGHT_--;  //Счетчик количества шагов – уменьшаем
    // Принимаем следующую длительность шага
    counter_do_step_WEEL_RIGHT_ = newcounter_do_step_WEEL_RIGHT_;
   }
  counter_WEEL_RIGHT_++; //Счетчик длительности шага
 }

 if (True_speed_WEEL_LEFT_) //Если есть шаги
 {
  if  (counter_WEEL_LEFT_  >=  counter_do_step_WEEL_LEFT_)//Если  шаг закончен
   {
    sost |= STEP_mask_on_WEEL_LEFT_; //Поднять шаг -  пока только флаг
    //Счетчик длительности шага - обнуляем для начала нового шага
    counter_WEEL_LEFT_ = 0;
   //  STEPS_counter_WEEL_LEFT_--;  //Счетчик количества шагов – уменьшаем
    // Принимаем следующую длительность шага
    counter_do_step_WEEL_LEFT_ = newcounter_do_step_WEEL_LEFT_;
   }
  counter_WEEL_LEFT_++; //Счетчик длительности шага
 }
```

```
  //Если шаг нужно делать
  //Опускаем сигналы шагов (пока только в переменной)
  //Поднимаем шаги для моторов, которым это нужно
  //Записываем значение с измененными битами шагов в регистр GPIO (0-31)
  if (sost != 0)
  {
    old_sost = true;
    REG_WRITE(GPIO_OUT_REG, (REG_READ(GPIO_OUT_REG) | sost));
  }
  // Если шаги делать не нужно, но нужно опустить
  // поднятые при прошлом вызове импульсы шагов
  else if (old_sost) {
    old_sost = false;
    // Опускаем все значения на шагов на 0
    REG_WRITE(GPIO_OUT_REG, (REG_READ(GPIO_OUT_REG) & off_step));
  }

  if (sonarAVTO)  //работа с УЗ датчиком
  {
    static uint32_t start_time = 0;
    static uint32_t stop_time = 0;
    static bool startFlag = false;
    static bool stopFlag = false;
    static uint32_t sonar_delay_time = 0;
    static uint32_t duration = 0;   //максимальное время ожидания отраженного сигнала в десятках микросекунд
    static uint32_t time_ = 0;  //текущее время
    static uint32_t flag_sonar = FLAG_SONAR_START1;
    if (flag_sonar == FLAG_SONAR_START1)
```

```
  {
   startFlag = false;
   stopFlag = false;
   // Переводим лимит из сантиметров в относительные величины (десятки
микросек.)
   duration = (Limit * 588) / 100;
   //Генерируем импульс
   digitalWrite(SONAR_TRIG, HIGH);
   flag_sonar = FLAG_SONAR_START2;

  }
  else if (flag_sonar == FLAG_SONAR_START2)
  {
   digitalWrite(SONAR_TRIG, LOW);
   //Ждем отражения импульса.
   // Пересчитываем время в расстояние (по скорости звуку).
   time_ = 0;
   stop_time = duration;
   flag_sonar = FLAG_SONAR_REGread;
  }
  else if (flag_sonar == FLAG_SONAR_REGread)
  {
   if ((time_ < duration) && (stopFlag == false))
   {

    registr_in = REG_READ(GPIO_IN_REG);

    if (startFlag == false)
    {
     if ((registr_in & long_mm_GPIO) != 0) {
```

```
          start_time = time_;
          startFlag = true;
        }
       }
       else
       {
        if ((registr_in & long_mm_GPIO) == 0)
        {
          stop_time = time_ - start_time;
          stopFlag = true;
        }
       }
       time_ ++;
     }
     else
     {
       flag_sonar = FLAG_SONAR_STOP;
       dist_mm    = (stop_time * 1000) / 588;   //в миллиметрах возвращаем
значение
       sonar_delay_time = 0;
     }
   }
   else if (flag_sonar == FLAG_SONAR_STOP)
   {
    if (sonar_delay_time < 8000) //80 миллисекунд пауза
      sonar_delay_time++;
    else
      flag_sonar = FLAG_SONAR_START1;
   }
 }
```

```
  static bool startSig = false;  //работа с рукой
  if (handACTIV)
  {
   if (LongSig < 2000) //частота 50Гц
   {
    LongSig++;
    if ((LongSig > handLS) && startSig)
    {
     startSig = false;
     REG_WRITE(GPIO_OUT_REG, (REG_READ(GPIO_OUT_REG) &
hand_maskSigOff));
     //  digitalWrite(2,LOW);
    }
   }
   else
   {
    REG_WRITE(GPIO_OUT_REG, (REG_READ(GPIO_OUT_REG) |
hand_maskSigOn));
    //  digitalWrite(2,HIGH);
    startSig = true;
    LongSig = 0;
   }
  }
  else if (startSig) {
   digitalWrite(2, LOW);
   startSig = false;
  }
}
```

```
//Готовим состояния масок
// Создание таймеров, прикрепление к ним
// функций обработчиков прерываний

void timer_setup()
{
  digitalWrite(GPIO_LEFT_WEEL_STEP, false);
  digitalWrite(GPIO_LEFT_WEEL_DIR, DIR_LEFT_FORWARD);
  DIR_tec_WEEL_LEFT_ = DIR_LEFT_FORWARD;

  digitalWrite(GPIO_RIGHT_WEEL_STEP, false);
  digitalWrite(GPIO_RIGHT_WEEL_DIR, DIR_RIGHT_FORWARD);
  DIR_tec_WEEL_RIGHT_ = DIR_RIGHT_FORWARD;



  // Создаем таймер
  Timer = timerBegin(2, 80, true);
  // Прикрепляем функциию обработчики
  timerAttachInterrupt(Timer, &core_timer, true);
  //Прерывание повторяется каждые 10 микросекунд
  timerAlarmWrite(Timer, 10, true);
  //Старт таймера, теперь прерывания будут генерироваться
  timerAlarmEnable(Timer);
}
```

Код внутреннего файла RaschetShag.h:

```
//функции обработки состояний моторов для разных режимов
void opros_sys_EN_motors()  //опрос моторов
{
  static bool wait_MOTORS_ON = false;
  static uint32_t safe_start_timer = 0;  //таймер включения моторов
```

```
 static bool realSTEPER_EN_level = MOTORS_OFF;  //флаг, обозначающий начало включения моторов
 if ((arming == true) && (STEPER_EN_level != MOTORS_ON))  //проверка готовности моторов (включение)
 {
  if (!wait_MOTORS_ON)  //если моторы готовы
  {
   realSTEPER_EN_level=MOTORS_ON;
   digitalWrite(GPIO_MOTORS_EN, MOTORS_ON);  //включение моторов
   safe_start_timer = 1000 + millis(); // даем секунду на безопасное включение
   wait_MOTORS_ON = true;  //поднимаем флаг, что моторы включены и ими можно управлять
  }
  else
  {
   if (safe_start_timer < millis())  //разрешение на управление моторами
   {
    STEPER_EN_level = MOTORS_ON;
    wait_MOTORS_ON = false;
   }
  }
 }
 else
 { if ((arming == false) && (realSTEPER_EN_level == MOTORS_ON)) //отключение моторов
  {
   realSTEPER_EN_level =MOTORS_OFF;
   STEPER_EN_level = MOTORS_OFF;
   digitalWrite(GPIO_MOTORS_EN, STEPER_EN_level);
   wait_MOTORS_ON = false;
```

```
   }
  }
}
void CalcSpeedAndPosition(double Dt)   //расчёт длины шага (скорости моторов)
{
  //Получаем от приемника два значения
  // 1. Командный угол поворота newROT_Angel_RAD - в радианах
  // 2. Командную скорость newWheelSpeed - в шагах на секунду

  if (STEPER_EN_level == MOTORS_ON)  //моторы включены, ограничиваем минимальную и максимальную скорости
  {
   RIGHT_WEEL_SPEED = constrain(commandRIGHT_WEEL_SPEED, -MAX_WEEL_SPEED, MAX_WEEL_SPEED);
   LEFT_WEEL_SPEED = constrain(commandLEFT_WEEL_SPEED, -MAX_WEEL_SPEED, MAX_WEEL_SPEED);
  }
  else
  {
   RIGHT_WEEL_SPEED = 0;
   LEFT_WEEL_SPEED = 0;
  }
  reaLsPeed = 0;  //реальная скорость
  temp = fabs(LEFT_WEEL_SPEED);  //временная переменная
  if (temp >= MIN_WEEL_SPEED) {
   LONG_STEP_WEEL_LEFT = uint32_t(100000.0 / temp);   //длительность шага левого колеса (десятков микросекунд или одна секунда)
```

```
  LONG_STEP_WEEL_LEFT = constrain(LONG_STEP_WEEL_LEFT, 3, MaxLONGStep); //ограничиваем длительность шага (самый короткий шаг 3x10 микросекунд)
  reaLsPeed = 100000.0 / double(LONG_STEP_WEEL_LEFT); //реальная скорость в шагах на секунду
  if (LEFT_WEEL_SPEED < 0) reaLsPeed = -reaLsPeed; //отрицательная скорость (торможение)
 } else  LONG_STEP_WEEL_LEFT = 0;  //длительность шага левого колеса = 0, так как скорость меньше 400 шагов в секунду
 if (LEFT_WEEL_SPEED < 0.0) dir_STEP_WEEL_LEFT = DIR_LEFT_BACKWARD;  //если скорость у левого колеса меньше 0, то мотор вращается назад
 else dir_STEP_WEEL_LEFT = DIR_LEFT_FORWARD;  //иначе мотор вращается вперёд

 temp = fabs(RIGHT_WEEL_SPEED);
 if (temp >= MIN_WEEL_SPEED) {
  LONG_STEP_WEEL_RIGHT = uint32_t(100000.0 / temp);  //длительность шага правого колеса
  LONG_STEP_WEEL_RIGHT = constrain(LONG_STEP_WEEL_RIGHT, 3, MaxLONGStep);

  if (RIGHT_WEEL_SPEED < 0) reaLsPeed -= 100000.0 / double(LONG_STEP_WEEL_RIGHT);  //добавляем скорость правого колеса к левому
  else reaLsPeed += 100000.0 / double(LONG_STEP_WEEL_RIGHT);

 } else LONG_STEP_WEEL_RIGHT = 0;  //длительность шага правого колеса = 0, так как скорость меньше 400 шагов в секунду
```

```
 if (RIGHT_WEEL_SPEED < 0.0) dir_STEP_WEEL_RIGHT = DIR_RIGHT_BACKWARD;  //если скорость у правого колеса меньше 0, то мотор вращается назад
 else dir_STEP_WEEL_RIGHT = DIR_RIGHT_FORWARD;  //иначе мотор вращается вперёд
 newSpeedflag_ = true;  //флаг получения новой скорости
 Sbo = (RIGHT_WEEL_SPEED + LEFT_WEEL_SPEED) / 2.0;  //текущая скорость
 reaLsPeed *= 0.5;  //средняя реальная скорость
}
```

Код внутреннего файла ServoRuka.h:

```
int minUs_div10 = 80; //800  перевод из градусов в длительность ШИМ сигнала для управления сервомашинкой руки
int maxUs_div10 = 250; //2500
int one_handGPIO = 23;
bool one_handattach = false;
#define hand_GPIO 2
const uint32_t hand_maskSigOn = 1 << hand_GPIO;
const uint32_t hand_maskSigOff = 0xFFFFFFFF ^ hand_maskSigOn;
bool handACTIV = false;  //работает ли в данный момент рука
uint32_t handANGLE = 0;  //угол руки
uint32_t handLS = 0;  //длина шим сигнала

uint32_t LongSig = 0;
void write_newHandrot(uint ang)  //функция пересчитывания угла в длину импульса
{
 ang = constrain(ang, 0, 180);
 handLS = map(ang, 0, 180, 80, 250);
 handANGLE = ang;
```

```
}
void servo_hand_setup()  //инициализация сервопривода
{
 pinMode(hand_GPIO, OUTPUT);

 handACTIV = false;
 write_newHandrot(90);
}


void servo_hand_bottom(double test_angle)   //функция поворота руки в направление, необходимое для поднятия
{
 if (test_angle < - 0.7) //Если вошли в критичный режим (0.7 - угол наклона робота в радианах (40 град.))
  {
    write_newHandrot(200);   //задаём угол поворота руки в градусах (в какую сторону упал робот, туда и поворачиваем руку для подъёма)
  }
 else if (test_angle >  0.7) //Если вошли в критичный режим
  {
    write_newHandrot(0);   //задаём угол поворота руки в градусах (в какую сторону упал робот, туда и поворачиваем руку для подъёма)
  }
}

void servo_hand_up()   //функция поднятия руки в исходное положение, если робот поднят
{
 write_newHandrot(90);  //задаём угол поворота руки в градусах
```

}

Код внутреннего файла ZnachPerem.h:

```
const int32_t dSmeshenie = 20;

uint32_t safe_start_timer = 0; // Таймер включения моторов

double  CompensatorZ = 0;  //дрейф нуля по оси Z
double  CompensatorX = 0;  //дрейф нуля по оси X
double  CompensatorY = 0;  //дрейф нуля по оси Y


int32_t newSpeedLEFTwheel; //скорость левого колеса
int32_t newSpeedRIGHTwheel; //скорость правого колеса

double Turn=0;  //поворот

const double DTIME = 0.005; //Типовое время в секундах между опросами
double _1_d_DTIME = 1.0 / DTIME; //Типовое перевернутое время в секундах между опросами
const uint32_t MaxLONGStep = 100000 / 400;

double temp = 0;

const double MAX_WEEL_SPEED = 15000;  //Шагов в сек
const double MIN_WEEL_SPEED = 400;  //Шагов в сек
double ACCELERATION_WEEL = 37000;  //Шагов в сек
double commandRIGHT_WEEL_SPEED = 0;   //задаваемая скорость правого колеса
double commandLEFT_WEEL_SPEED = 0;   //задаваемая скорость левого колеса
```

```
double WEEL_SPEED = 0;
double LEFT_WEEL_SPEED = 0;  //реальная скорость левого колеса
double RIGHT_WEEL_SPEED = 0;  //реальная скорость правого колеса

uint32_t Pereschet;  //время для предстартовой подготовки
bool avtostart=false;  //поднятие рукой с помощью BT (спустя некоторое время после включения, чтобы пересчитались углы наклона и пришли в норму)
bool arming = false;  //включение моторов

int i = 0, j = 0; //Счетчики
float imu[3];
float quat[4];
float e[3];
double Dt = 0;

double reaLsPeed=0;  //реальная скорость
double Sbo=0;  //текущая скорость
double New_Speed = 0;//новая скорость
double OldSpeed = 0; //Скорость от прошлой итерации
double  Error_ang = 0; //Ошибка угла
double  OldError_ang = 0; //Ошибка угла
double  base_ang = 0; //Базовый угол
double  dError_ang = 0; //Скорость изменения ошибки

double  Error_speed = 0; //Ошибка скорости в шагах
double  dError_speed; //Скорость изменения ошибки скорости
double  OldError_speed; //Ошибка скорости в шагах
double   iError_speed;//Накопленная - интегральная ошибка скорости (ошибка пройденного расстояния)
```

```
double  Kp_a = 200000;  //Коэффициент при ошибке угла
double  Kd_a = 20000;  //Коэффициент при скорости изменения ошибки угла
double  Kp_s = 4;  //Коэффициент при ошибке скорости движения
double  Kd_s = 0.001;  //Коэффициент при скорости изменения ошибки скорости движения
double  Ki_s = 4;  //Коэффициент при интегральной ошибке скорости

uint32_t micros_1 = 0;
uint32_t OldTime = 0;

double CommandSpeed = 0;  //скорость управляемого движения
```

ПРИЛОЖЕНИЕ В – Чертежи устройства

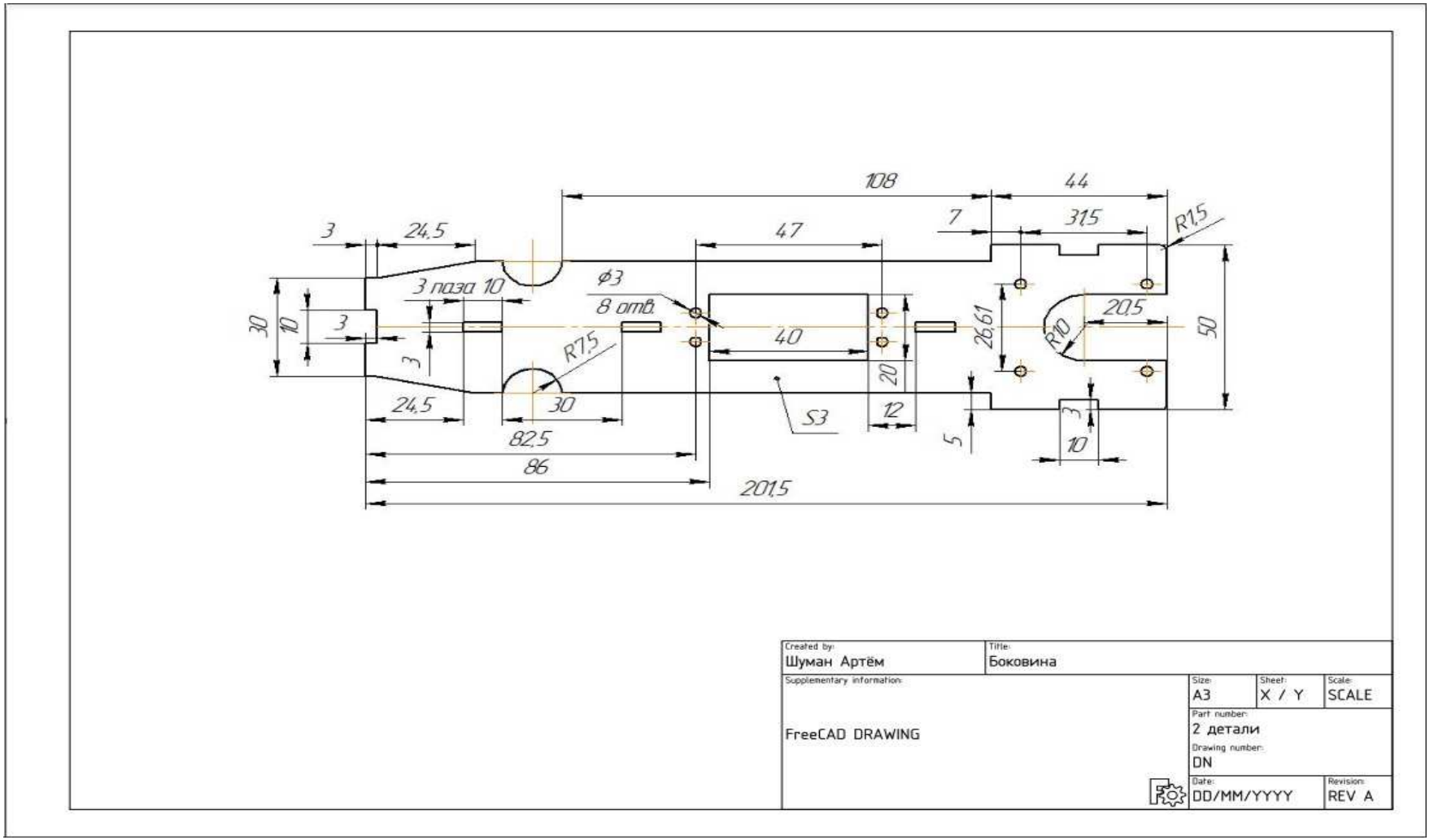


Рисунок В.1 – Чертёж детали «Боковина»

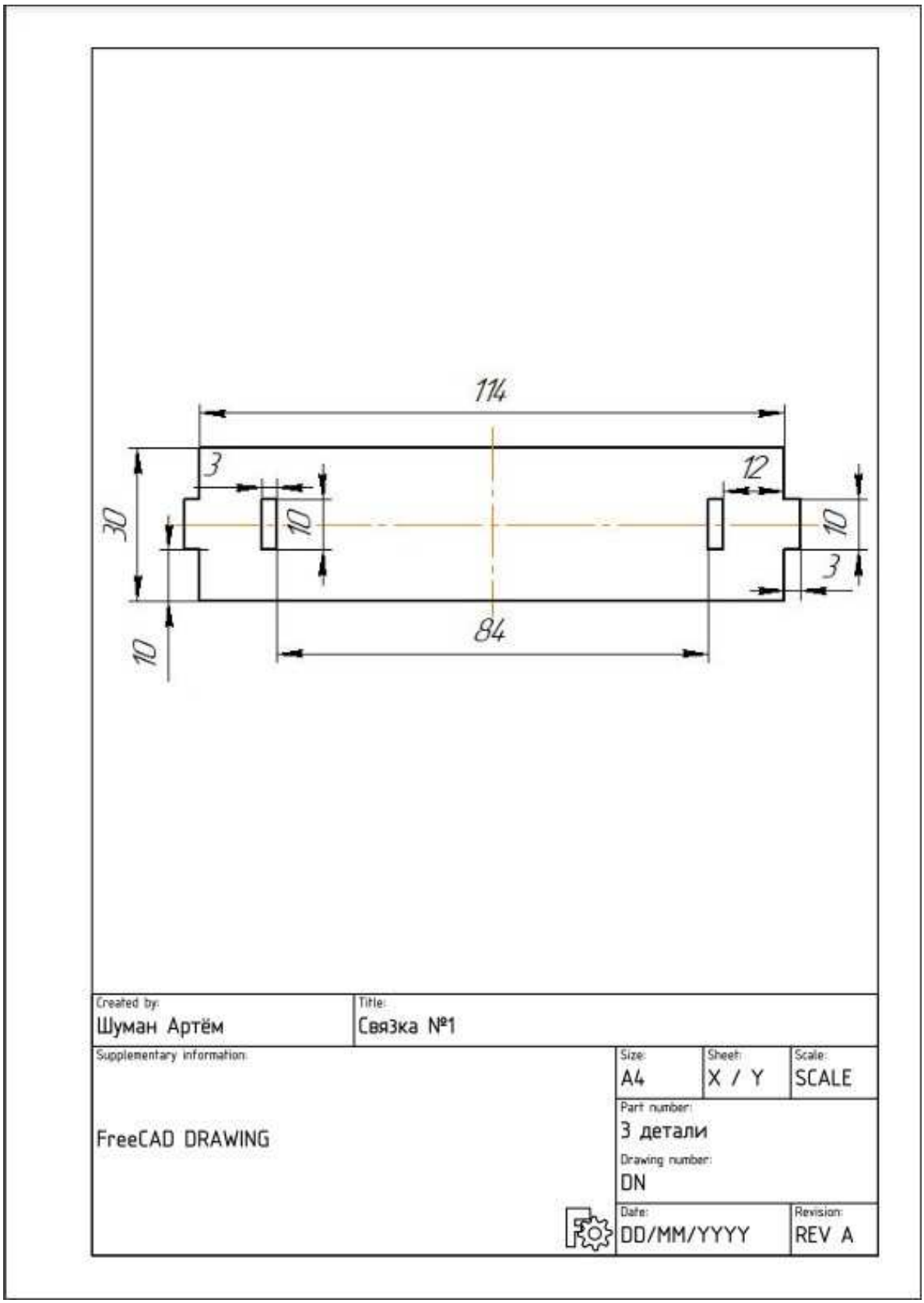


Рисунок В.2 – Чертёж детали «Связка №1»

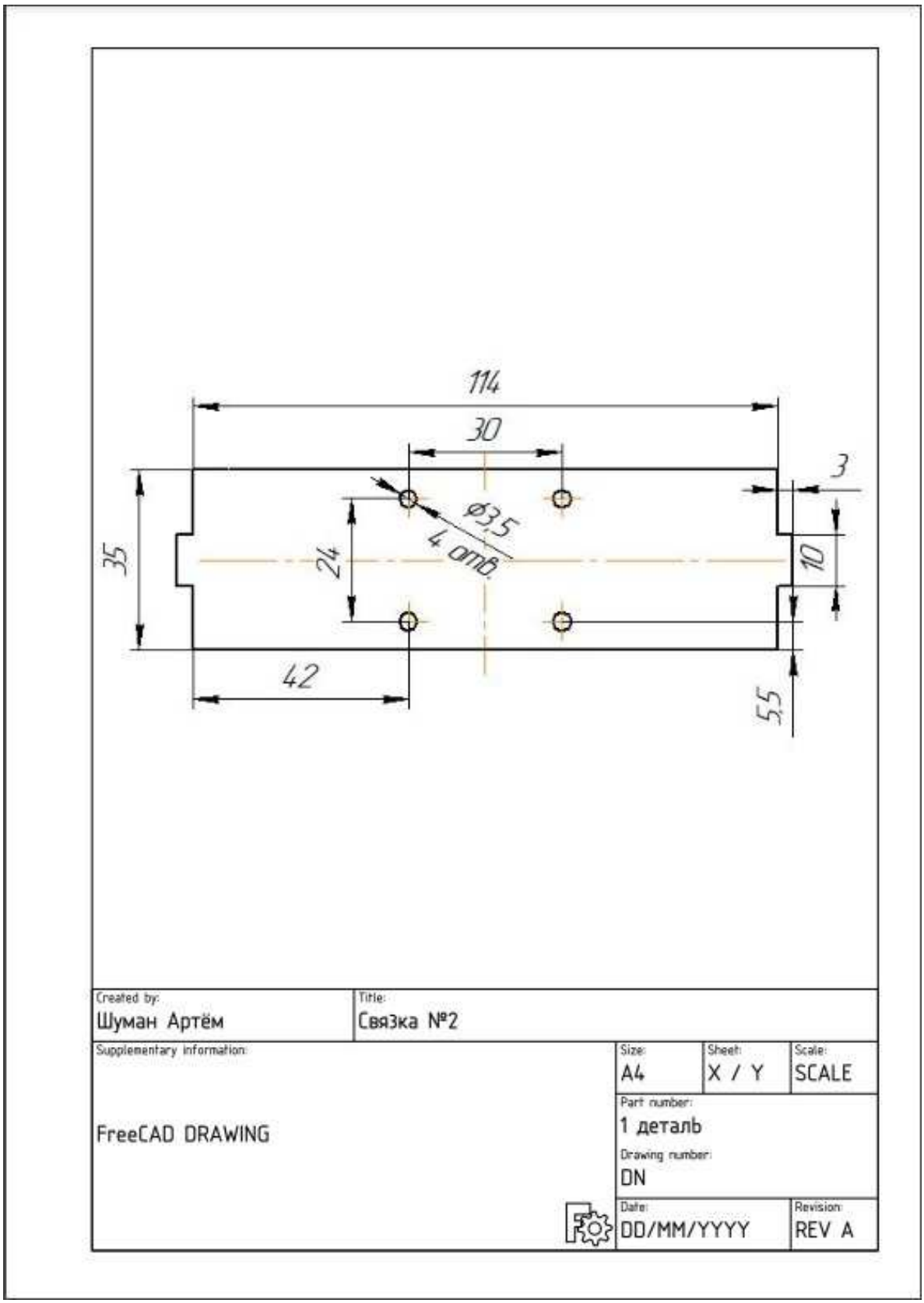


Рисунок В.3 – Чертёж детали «Связка №2»

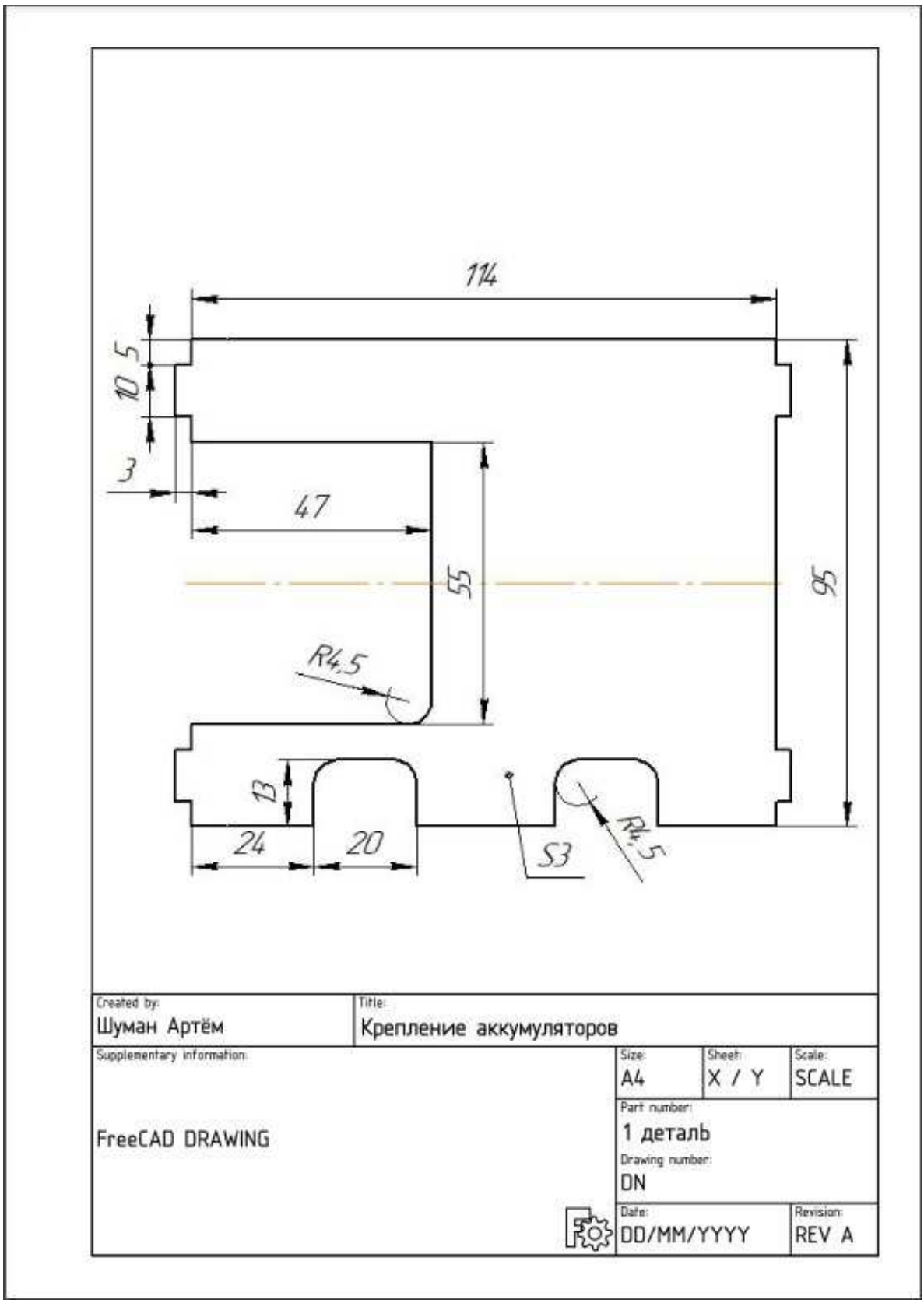


Рисунок В.4 – Чертёж детали «Крепление аккумуляторов»

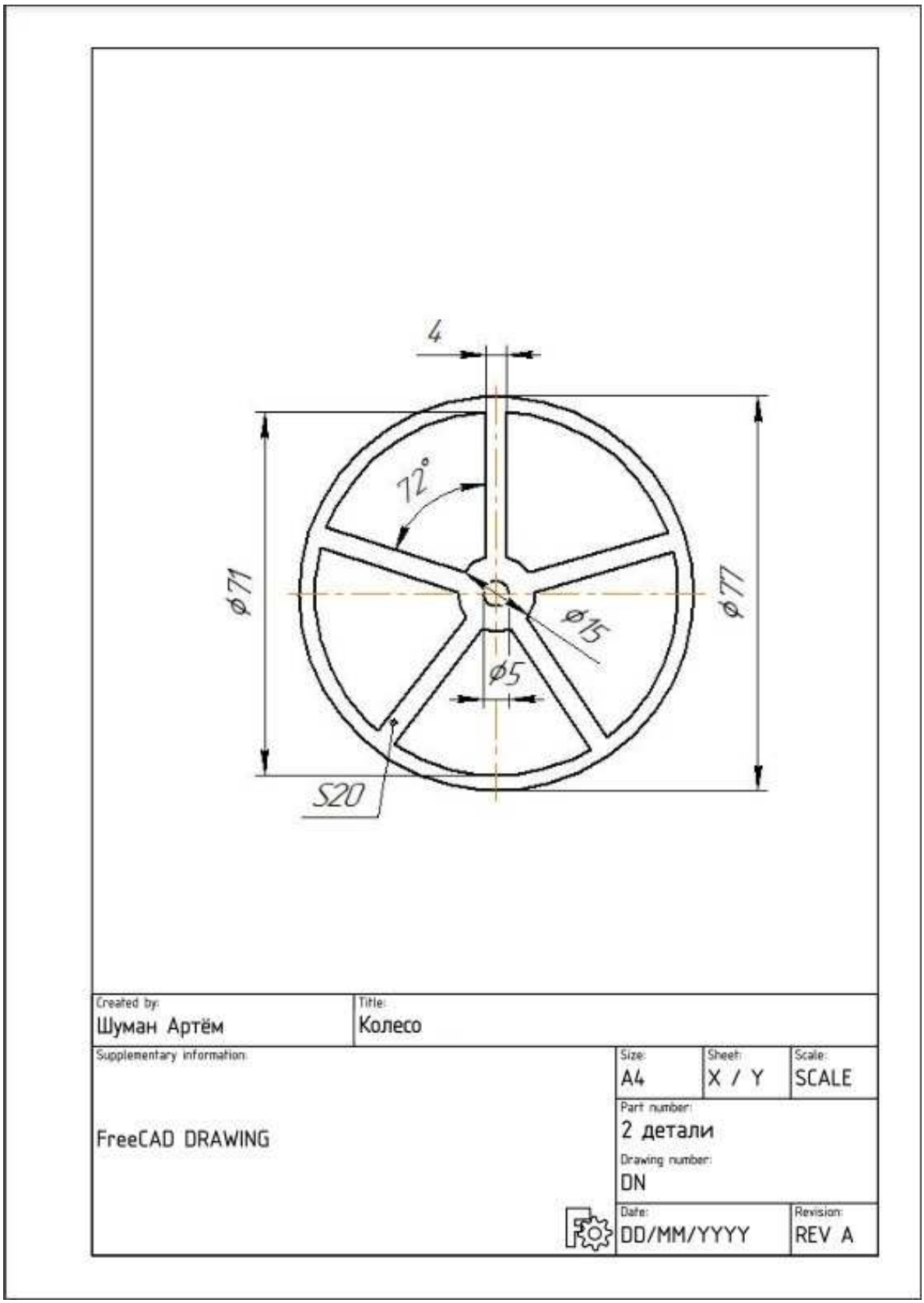


Рисунок В.5 – Чертёж колеса

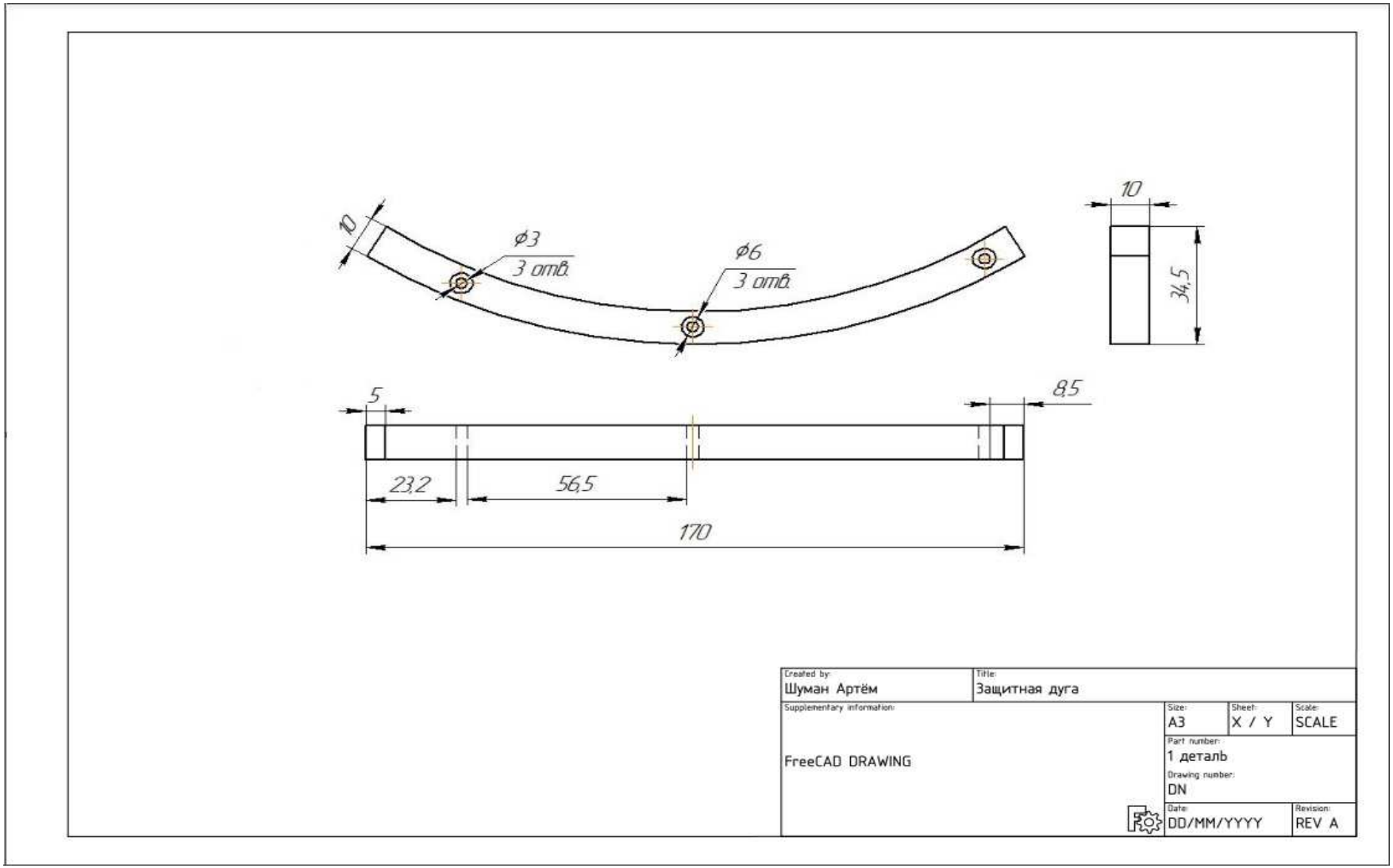


Рисунок В.6 – Чертёж защитной дуги

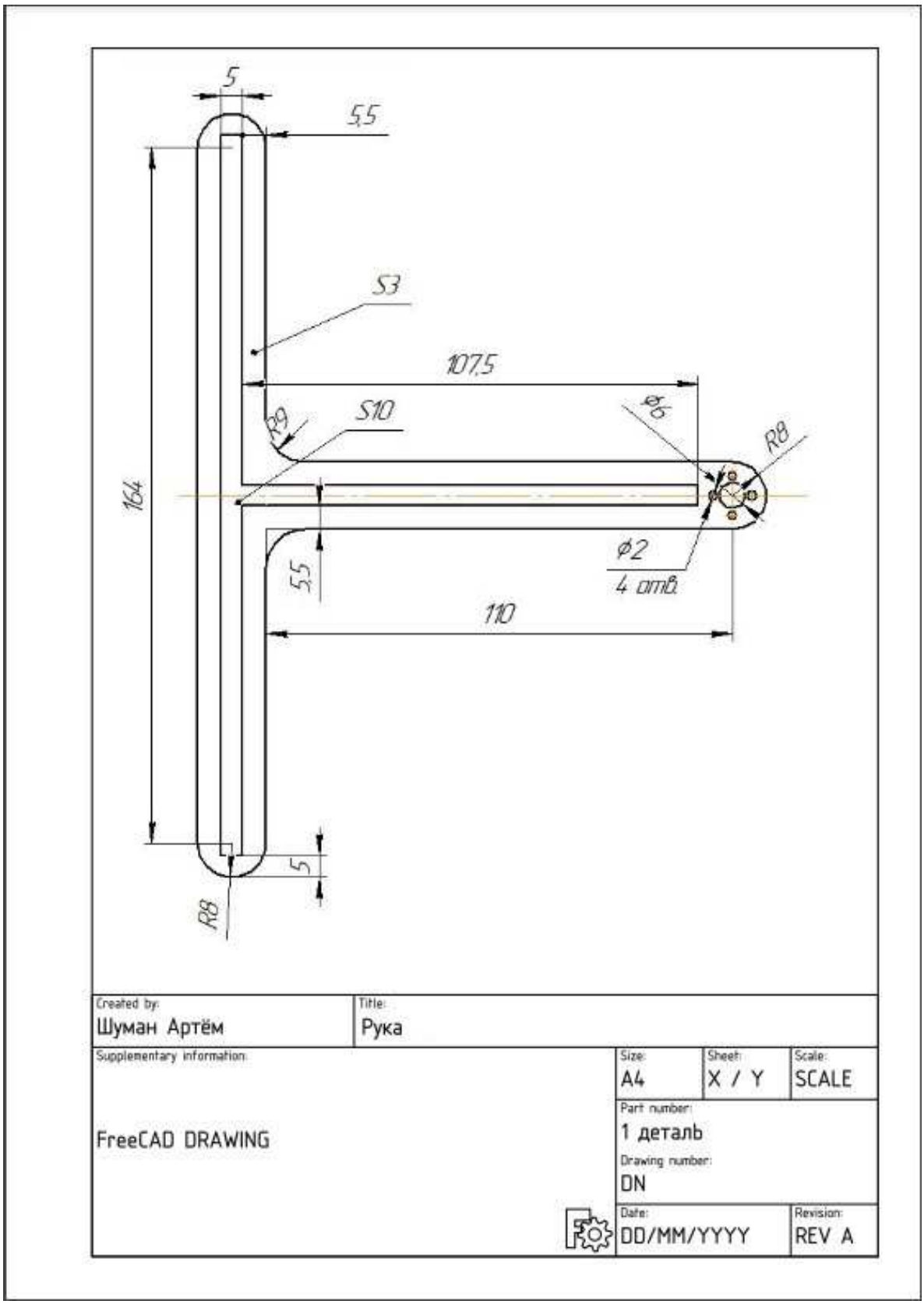


Рисунок В.7 – Чертёж руки