Министерство образования Республики Беларусь

Учреждение образования «Гродненский государственный университет имени Янки Купалы»

Физико-технический факультет

Кафедра общей физики

**УСТРОЙСТВО ИЗМЕРЕНИЯ ЛИНЕЙНГО И УГЛОВОГО ПЕРЕМЕЩЕНИЯ НА МИКРОКОНТРОЛЛЕРЕ**

Дипломная работа студента

физико-технического факультета

Вороновича Павла Антоновича

Научный руководитель,

доктор физико-математических наук,

доцент Маскевич А. А.

ГРОДНО 2020

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ.**

**LCD** жидко-кристаллический экран.

**DD** графическое обозначение микроконтроллером.

**VCC** положительное напряжение источника питания и устройств.

**GND** общий провод источника питания и устройств.

**SDA** шина последовательных данных.

**SLC** шина тактирования.

**ДУ** дистанционное управление

**ТТЛ** транзисторно-транзисторная логика

**МК** микроконтроллер

**РЕФЕРАТ**

**УСТРОЙСТВО ИЗМЕРЕНИЯ ЛИНЕЙНГО И УГЛОВОГО ПЕРЕМЕЩЕНИЯ НА МИКРОКОНТРОЛЛЕРЕ**

Дипломная работа 58 стр., 1 табл., 6 рис., 20 источников, 8 приложений.

Объектами исследованийявлялисьмикроконтроллеры и устройства измерения линейного и углового перемещения.

Целью данной дипломной работы является разработка, проектирование и создание устройства измерения линейного и углового перемещения на микроконтроллере.

Работа состоит из 4 глав. Первая глава посвящена разработке структурной схемы устройства, а также составлению элементной базы необходимой для изготовления конечного устройства. Вторая глава описывает работу устройства по схеме электрической функциональной, так и по схеме электрической принципиальной. В третьей главе представлена программная часть разработанного устройства. В четвертой главе описывается требования охраны труда при выполнении работ.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc42620797)

[1 Устройство измерения линейного и углового перемещения на микроконтроллере 7](#_Toc42620798)

[1.1 Схема электрическая структурная 8](#_Toc42620799)

[1.2 Выбор элементной базы 11](#_Toc42620800)

[2 ПРИНЦИП РАБОТЫ УСТРОЙСТВА 16](#_Toc42620801)

[2.1 Схема электрическая функциональная 16](#_Toc42620802)

[2.2 Схема электрическая принципиальная 20](#_Toc42620803)

[3 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ 26](#_Toc42620804)

[3.1 Структура программного кода 26](#_Toc42620805)

[3.2 Описание режимов программного кода 29](#_Toc42620806)

[4 ТЕХНИКА БЕЗОПАСНОСТИ 32](#_Toc42620807)

[4.1 Техника безопасности при проведении электромонтажных работ 32](#_Toc42620808)

[4.2 Требования безопасности перед началом работы 33](#_Toc42620809)

[4.3 Требования безопасности во время работы 34](#_Toc42620810)

[4.4 Требования безопасности по окончании работы 35](#_Toc42620811)

[4.5 Требования безопасности в аварийных ситуациях 35](#_Toc42620812)

[ЗАКЛЮЧЕНИЕ 37](#_Toc42620813)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 38](#_Toc42620814)

[ПРИЛОЖЕНИЯ 39](#_Toc42620815)

ВВЕДЕНИЕ

В современном мире электроника и различная вычислительная техника играет важную роль в различных областях человеческой деятельности. Это связано с тем, что прогресс не стоит на месте, все повсеместно модернизируется, улучшается, автоматизируется. В процессе деятельности перед человеком с каждым разом стоят более серьезные вопросы и задачи, решение которых без применения различных электронных устройств либо попросту невозможно, либо займет слишком много времени. Все эти, казалось бы, маленькие факты и нюансы достаточно серьезно влияют на скорость и точность получаемых результатов, фактически любых измерений. Кроме того, в последнее время в научно-технической сфере, все чаще стали применяться различные микроконтроллеры, которые позволяют ускорить и упросить решение достаточно большого спектра задач.

Практикум и выполнение лабораторных работ является важной частью при подготовке студентов инженерной специальности. Навыки, получаемые в физической лаборатории (понимание наблюдаемых процессов, пользование измерительными приборами, обработка полученных результатов), важны для будущей трудовой деятельности студентов. Применение новых технических решений делает даже классическую лабораторную работу по физике достаточно привлекательной. В связи с этим, было принято решение о создании универсального блока управления на микроконтроллере. Данное устройство позволяет более детально изучить различные физические явления, связанные с линейными или угловыми перемещениями [9].

Целью данной работы является разработка функционального блока для управления и обработки измерений линейных и угловых перемещений физических тел с использованием микроконтроллера.

Основные задачи, необходимые для разработки устройства измерения линейного и углового перемещения на микроконтроллере:

1. Проектирование схемы электрической структурной.
2. Выбор элементной базы.
3. Проектирование схемы электрической функциональной.
4. Проектирование схемы электрической принципиальной.
5. Разработка и написание программного обеспечения.
6. Отладка программного кода.
7. Сборка и регулировка разработанного устройства.

1 Устройство измерения линейного и углового перемещения на микроконтроллере

Для достижения основной цели данной дипломной работы необходимо реализовать измерительное устройство со следующими техническими характеристиками:

- диапазон измерения линейного перемещения 0 – 100 см;

- диапазон измерения углового перемещения 0 – 180°;

- дискретность измерения линейного перемещения 1 мм;

- дискретность измерения углового перемещения 0,1°;

- реализовать дистанционное управление (ДУ) внешними устройствами (многофункциональные измерители временных интервалов, а также системами управления механизмом перемещения);

- в качестве датчика применить оптоэлектронный энкодер;

- блок управления и измерения выполнить на микроконтроллере;

- динамическая индикация;

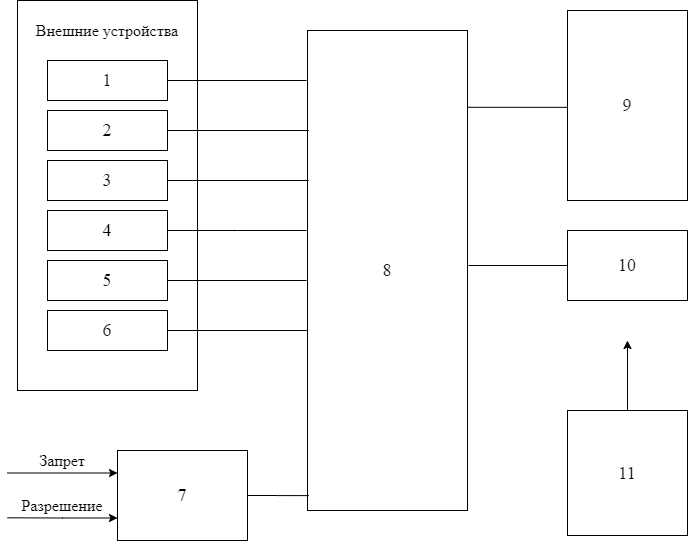
- напряжение питание 5В;

- составить структурную, функциональную и принципиальную схемы;

Схемы электрические структурная, функциональная, принципиальная должны учитывать заданные технические характеристики. И позволить изготовить устройство измерения линейного и углового перемещения на микроконтроллере.

1.1 Схема электрическая структурная

Схема электрическая структурная разработанного устройства представлена на рисунке 1.



1- датчик пуска на оптопаре, 2 - датчик подсчета периодов, 3 - датчик линейного перемещения, 4 - датчик углового перемещения влево, 5 - датчик углового перемещения вправо, 6 - блок выбора режима работы, 7 – блок дистанционного управления, 8 - блок измерения и управления, 9 - блок индикации, 10 – звуковой излучатель, 11 - блок питания 9 В

Рисунок 1 – Схема электрическая структурная

В данной схеме представлены основные функциональные компоненты разработанного устройства. Кроме того, в ней представлены взаимосвязи между основными компонентами данной схема. Стоит отметить, что каждый блок представленный в данной схеме реализует полностью законченное действие.

Поскольку блок измерения и управления 8 разрабатывается, как многофункциональное устройство, в данном блоке реализуется 3 режимом работы. Далее приведены основные режимы, реализованные на программном уровне:

1. Режим измерения линейного перемещения
2. Режим измерения углового перемещения
3. Режим подсчета периодов колебания

При включении блока управления в сеть, устройство совершает аппаратный сброс. Далее с помощью Блока выбора режима 6 представленного в схеме электрической структурной, происходит установка режима и его выбор.

Описание и рассмотрение режимов работы разработанного и изготовленного устройства описано ниже.

В состав схемы режима измерения линейного перемещения входят следующие компоненты: датчик пуска на оптопаре 1, датчик линейного перемещения 3, блок выбора режима работы 6, внешние сигналы управления 7, блок измерения и управления 8, блок индикации 9.

Первым этапом выполнения данного режима является установка расстояния при помощи блока выбора режима 6, которое должно пройти тело. Задаваемое значение отображается на блоке индикации 9. При поступлении разрешающего сигнала на блок дистанционного управления 7, после чего ожидается получение управляющего сигнала с датчика пуска 1 на блок управления и измерения 8. Данный управляющий сигнал является флагом для перехода блока 8 во второй этап выполнения управляющей программы.

Вторым этапом выполнения данного режима является получение и подсчет импульсов, получаемых от датчика линейного перемещения 3. После чего происходить обработка данных измерений на программном уровне. По окончанию линейного перемещения тела, результат измерений выводиться на блок индикации 9, после чего устройство переходит в исходное положения и может быть использовано в любом из пяти возможных режимов работы.

В состав схемы режима измерения углового перемещения входят следующие компоненты: датчик углового перемещения влево 4, углового перемещения вправо 5, блок дистанционного управления 7, блок измерения и управления 8, блок индикации 9.

При старте данного режима ожидается получение разрешающего сигнала с блок дистанционного управления. После чего ожидается получение сигналов с датчиков углового перемещения 4 и 5 на соответствующие входы блока управления и измерения 8. После чего происходит программная обработка полученных результатов. После выполнения программной части результат вычислений выводиться на блок индикации 9. После окончания данного режима, разработанное устройство переходит в исходное положения и может быть использовано в любом из возможных режимов работы.

Третий режим позволяет производить измерения периодов механических колебаний различных видом маятников (математический, маятник Обербека и др.). В состав схемы данного режима входят следующие компоненты: датчик подсчета периодов 2, блок выбора режима работы 6, блок измерения и управления 8, блок индикации 9, звуковой излучатель 10.

Начальным этапом данного режима, является установка необходимого количества колебаний для измерений с помощью блока выбора режима работы 6. После чего ожидается получение разрешающего сигнала на блок дистанционного управления 7. Данный сигнал является флагом для блока управления и измерения 8, который начинает подсчет импульсов, получаемых с датчика подсчета периодов 2. Сигнал вышеупомянутого датчика поступает на блок 8 при каждом перекрытии датчика 2 стержнем маятника. После чего происходить программная обработка получаемых данных. По завершению данного режима, подается звуковой сигнал с использованием звукового излучателя 10 и вывод информации на блок индикации 9. По окончанию данного режима, устройство переходит в исходное положения и может быть использовано в любом из пяти возможных режимов работы.

1.2 Выбор элементной базы

**1.2.1 Выбор микроконтроллера**

Микроконтроллер – это специальная микросхема, предназначенная для управления различными электронными устройствами. Данное устройство объединяет в себе процессор, память, ПЗУ и периферию внутри одного корпуса, внешне похожего на обычную микросхему [5].

Выбор микроконтроллера следует начать с описания основных семейств МК. Основные рассматриваемыми вариантами являются семейства ARM, AVR и PIC.

ARM — семейство лицензируемых 32-битных и 64-битных микропроцессорных ядер разработки компании ARM Limited. Представляет собой микроконтроллеры придерживающихся идеологии RISC для системных команд. Данное семейство микроконтроллеров является одним из лидеров на рынке микроконтроллеров за счет своей простоты и дешевизны. По приведенной статистике, на процессоры ARM приходилось порядка 90 % всех встроенных 32-разрядных процессоров. Поскольку данные процессоры имеют низкое энергопотребление, в связи с этим они находят широкое применение во встраиваемых системах и преобладают на рынке мобильных устройств, для которых данный фактор немаловажен [11].

AVR - семейство восьми битных микроконтроллеров. Данные микроконтроллеры являются представителями гарвардской архитектуры, что подразумевает под собой разделение памяти программ и памяти данных. Также стоит отметить, что данное семейство имеет систему команд, близкую к идеологии RISC. Особенностью такого подхода, является упрощение системы команд, как результат конечный пользователь получает выигрыш в производительности и быстродействии. Процессор AVR имеет 32 8-битных регистра общего назначения, объединённых в регистровый файл. Кроме всего прочего, стоит отметить, что система команд микроконтроллеров AVR весьма развита и насчитывает в различных моделях от 90 до 133 различных инструкций. Большинство команд выполняется за 1 такт. Что крайне положительно сказывается на производительности [10].

PIC - семейство восьми битных или же шестнадцати битных микроконтроллеров. Данному семейству соответствует RISC архитектура, обеспечивающая выпол­нение большинства команд процессора за один машинный цикл. Гарвардская архитектура, обес­печивающая одновременный доступ к памяти данных и программ.  КМОП технология, обеспечивающая: • полностью статический режим работы, при котором остановка тактового генератора не приводит к потере логических состояний внутренних узлов, широкий диапазон напряжений питания (2…6 В) и температур (–40…+70 °С) а также малое энергопотребление.

Следующим этапом является сравнение наиболее популярных 8 – ми разрядных МК представленных на рынке. Для этого были выбраны следующие представители. Семейство AVR представлено микроконтроллером серии Atmega8, а именно моделью ATmega328P-AU. Семейство ARM будет представлено так же 8 – ми разрядным микроконтроллером STM8L151C8T6. Представителем семейства PIC является PIC16F627A.

Из таблицы 1 можно увидеть, что все микроконтроллеры, приведенные в таблице, являются представителя RISC архитектуры. Кроме того, стоит отметить, что все 3 модели имеют одинаковый тип памяти программ, но в тоже время модель из серии STM8, имеет в два раза больший объем данной памяти, в сравнении с моделью из серии Atmega и гораздо больший объем в сравнении с моделью из серии PIC16. Кроме того в пользу выбора модели из серии STM8, мог бы говорить тот факт, что у данной модели в два раза больше энергонезависимой памяти (EEPROM). Аналогичная ситуация обстоит и с оперативной памятью (RAM). Но стоит заметить, что модель из серии STM8 проигрывает двум оставшимся моделям в тактовой частоте, которая влияет на производительность микроконтроллера. Также немаловажным параметром является количество команд для управления микроконтроллером и в данном случае, бесспорным лидером является представитель серии Atmega. Так же немаловажным фактом при выборе микроконтроллера было наличие I2C интерфейса. В модели серии PIC16 данный интерфейс отсутствует, что является серьезным недостатком данного микроконтроллера.

Сравнив 3 наиболее популярные модели, было решено выбирать модель микроконтроллера серии Atmega или же серии STM8. Поскольку модель микроконтроллера серии PIC16 пусть и дешевле, но проигрывает оставшимся двум вариантом почти по всем характеристикам и параметрам.

После более детально рассмотрения двух вышеупомянутых моделей, было принято решение отдать предпочтение модели серии Atmega. Данное решение обосновывается тем фактом, что данная модель выигрывает по быстродействию, а также имеет больший список различной технической литературы, что упрощает программную разработку. Так же немаловажным фактом является простота использования, а как результат, простата дальнейшей поддержки устройства в рабочем состоянии. Стоит отметить, что абсолютна вся документация для модели серии Atmega распространяется по свободной лицензии.

Таблица 1 - сравнительная характеристика наиболее популярных моделей семейства: AVR, ARM, PIC.

|  |  |  |  |
| --- | --- | --- | --- |
| Параметры | Модель микроконтроллера | | |
| ATmega328P-AU | STM8L151C8T6 | PIC16F627A |
| Семейство | AVR | ARM | PIC |
| Серия | ATmega | STM8 | PIC16 |
| Разрядность | 8 бит | 8 бита | 8 бита |
| Архитектура | RISC | RISC | RISC |
| Тактовая частота, МГц | 20 | 16 | 20 |
| Объем памяти программ, кбайт | 32 | 64 | 3,5 |
| Тип памяти программ | Flash | flash | flash |
| Объем EEPROM, кбайт | 1 | 2 | 0.128 |
| Объем RAM, кбайт | 2 | 4 | 0,224 |
| I2C интерфейс | 1 | 1 | - |
| Напряжение питания | 1.8…5.5 в | 1.8…3.6 в | 3…5.5 в |
| Список команд | 90 | 32 | 35 |
| Стоимость, BYN | 4.40 | 4.80 | 3.40 |

**1.2.2 Выбор операционного усилителя датчиков линейного и углового перемещения**

В качестве операционного усилителя датчиков линейного и углового перемещения, было принято использовать готовое решение на основе модуля KY-037, в состав которого входят два компаратора LM393YD.

**1.2.3 Выбор устройства для блока выбора режимов**

В качестве устройства для блока выбора режима было принято решение использовать электромеханический инкрементный энкодер KY-040 имеющий 24 импульса на один оборот. Плюсом данного устройства является его простата, поскольку в данном типе энкодера применятся скользящие контакты. Также плюсом является наличие кнопки. Минусом данного устройства является дребезг контактов, но это решается программной обработкой.

**1.2.4 Выбор устройства блока индикации**

В качестве индикации рассматривались два варианта: lcd дисплей 1602А или два семисегментных модуля tm1637. Поскольку для решения основной технической задачи касающейся индикации (динамическая индикация) подходят оба варианта, все же было решено отдать предпочтения lcd дисплею 1602А. Это вызвано тем фактом, что, выбрав данное устройство, мы можем выводить абсолютно любую текстовую информацию. Данный факт позволяет разработать приятный и отзывчивый интерфейс для пользователя.

Важным фактом является использование вышеупомянутого дисплея с подключением через I2C интерфейс. Данное решение применяется для уменьшения задействованных портов ввода/вывода микроконтроллера. Так при обычном подключении необходимо задействовать 6 цифровых портов, в тоже время при использовании модуля переходника на основе микросхемы PCF8574AT, которая предназначена для расширения количества линий ввода/вывода, задействованы лишь 2 цифровых порта МК. Микросхема подключается по I2C интерфейсу и имеет порт из 8 линий ввода/вывода.

**1.2.5 Выбор звукового излучателя**

В качестве звукового излучателя был выбран простейший пьезоизлучатель KPM-G1205В. Данный излучатель имеет высокое сопротивление 30 Ом (электродинамический излучатель имеет сопротивление 4.8 Ом), что позволяет подключать его непосредственно к выходу МК. Один из контактов подключается к цифровому порту блока управления и измерения, имеющему в наличии аппаратный широтно-импульсный модулятор. Второй контакт на общий провод.

**1.2.5 Выбор блока питания**

Данное устройство может работать от источника питания 9-24В. В качестве блока питания применяется внешний источник питания 9В/1А.

2 ПРИНЦИП РАБОТЫ УСТРОЙСТВА

2.1 Схема электрическая функциональная

На рисунке 2 приведена схема электрическая функциональная разработанного устройства. Схема содержит в себе функциональную схему блока внешних устройств, а также блока управления и измерения.

**2.1.1 Схема электрическая функциональная блока выбора режима**

Выбор режимов работы устройства происходит с помощью использования инкрементного электромеханического энкодера. Функциональная схема данного модуля построена на основе двух скользящих контактов CLK и DT, а также кнопки SW подключенной по схеме с подтягивающим резистором. При вращении энкодера с выходов CLK и DT поступают прямоугольные импульсы, на соответствующие входы блока управления и измерений, сдвинутые друг относительно друга на фазовый угол 90 °. При вращении в одну сторону задний фронт импульса на выходе CLK соответствует паузе между импульсами на выходе DT. После чего происходит программная обработка получаемых сигналов с помощью микроконтроллера.

**2.1.2 Схема электрическая функциональная блока индикации**

Поскольку для динамической индикации используется lcd дисплей 1602А подключенный по интерфейсу I2C с модулем PCF8574AT применяющемся для расширения линий ввода/вывода. Микросхема подключается по I2C интерфейсу и имеет порт из 8 линий ввода/вывода, принцип функционирования простой, при записи байта данных в микросхему, линии порта принимают уровни, соответствующие значениям битов полученного байта. Операция чтения возвращает байт данных, биты которого указывают состояние линий порта. Таким образом, микросхема позволяет расширить количество линий ввода/вывода, используя два управляющих провода.

**2.1.3 Схема электрическая функциональная датчиков линейного и углового перемещения.**

Схемы электрические функциональные и принципиальные внешних устройств построены по однотипным типовым схемам на операционном усилителе. Отличие заключается лишь в различных коэффициентах усиления, которые устанавливаются с помощью регулировочного резистора.

Принцип работы первых трех режимов, а именно режим линейного перемещения, режим углового перемещения и режим подсчета периодов рассматриваются по примеру функциональной схемы датчика линейного перемещения. Поскольку остальные режимы работы разработанного устройства реализованы на программном уровне.

Функциональная схема датчика линейного перемещения, построена на оптопаре состоящей из пары фотодиодов FD1 AL106 (излучающий) и FD2 AL107 (принимающий). Сигнал от FD1 прошедший механический энкодер, представляющий собой диск с прямоугольными отверстиями и напечатанный на 3D принтере, через FD2 поступает на однокаскадный усилитель, собранный на ВТ1 и компаратор представляющий собой операционный усилитель AD1. С операционного усилителя сигналы ТТЛ уровня поступают на советующий вход блока управления и измерения.

**2.1.4 Схема электрическая функциональная звуковой индикации**

Оповещение о завершение того-либо иного режима происходит с помощью пьезоизлучателя. При поступлении сигнала от блока управления и измерения сигнала на соответствующий цифровой порт вышеупомянутого блока, происходить широтно-импульсная модуляция благодаря наличии аппаратного широтно-импульсного модулятора на соответствующем порту. После чего за счёт ШИМ полученные импульсы поступают на пьезоизлучатель, который в свою очередь побуждает колебательные движения биморфной конструкции пьзоизлучателя, преобразовывая их в звук.

**2.1.6 Схема электрическая функциональная блока управления**

Схема содержит устройства связи системы с внешними устройствами, а также устройство индикации. Внешнее управление становится возможным и при помощи системного разъема, но по другим правилам в отличие от порта ввода/вывода, что расширяет возможности контроллера, с его помощью может осуществляется связь с шинами базовой ЭВМ. Также этот разъем может быть использован при отладке работы микропроцессорной системы в целом.

Так же, стоит отметь наличие в системе трех шин, а именно шины управления, адреса и данных. Они представляют из себя проводные линии(проводники) соединяющие между собой различные составные части всей системы. Шины имеют различные разрядности. Также стоит отметить, что разрядность шины может зависеть от производительности системы. Большое количество информации, вызывает потребность в большей шине данных.

Далее будет рассмотрено назначение каждой из шин, применяемых в данном микроконтроллере.

Шина данных – шина основное назначение которой является передача различного рода данных. Данные могут быть получены как из памяти, так и от устройства ввода/вывода и переданы процессору. Поскольку обратное также верно, можно говорить о том, что данная шина является двунаправленной. Следует отметить, что используемый микроконтроллер имеет 8-ми разрядную шину данных, что говорит о том, что за одну передачу/прием максимум может быть передано 8 бит информации.

Шина адреса — главное назначение данного элемента является указание адреса ячеек памяти или портов ввода/вывода. Разрядность же имеет прямую зависимость от необходимого размера памяти.

Шина управления – основной целью данного компонента является передача информации устройству ввода/вывода или памяти о готовности микропроцессора выполнить пересылку данных.

Как результат, процесс работы имеет следующую структуру. По управляющей шине передаются сигналы, которые позволяют устройству ввода/вывода или же памяти обращаться к процессору с запросами. Разрядность данной шины большей степени зависит от типа используемого процессора и количества его управляющих или используемых сигналов.

Из приведенной выше схемы видно, что инициатором всех действий, выполняемых системой является микропроцессор. Данный микропроцессор вырабатывает сигналы управления и выдает их на шину управления. Далее по данной шине сигналы подают на устройства памяти, а также на устройства ввода/вывода. Стоит отметить, что именно процессором задается и адрес выбранной ячейки памяти. После чего выбранный адрес подает на шину адреса и на адресные входы микросхем памяти, в свою очередь по шине данных информация идет в процессор или из него.

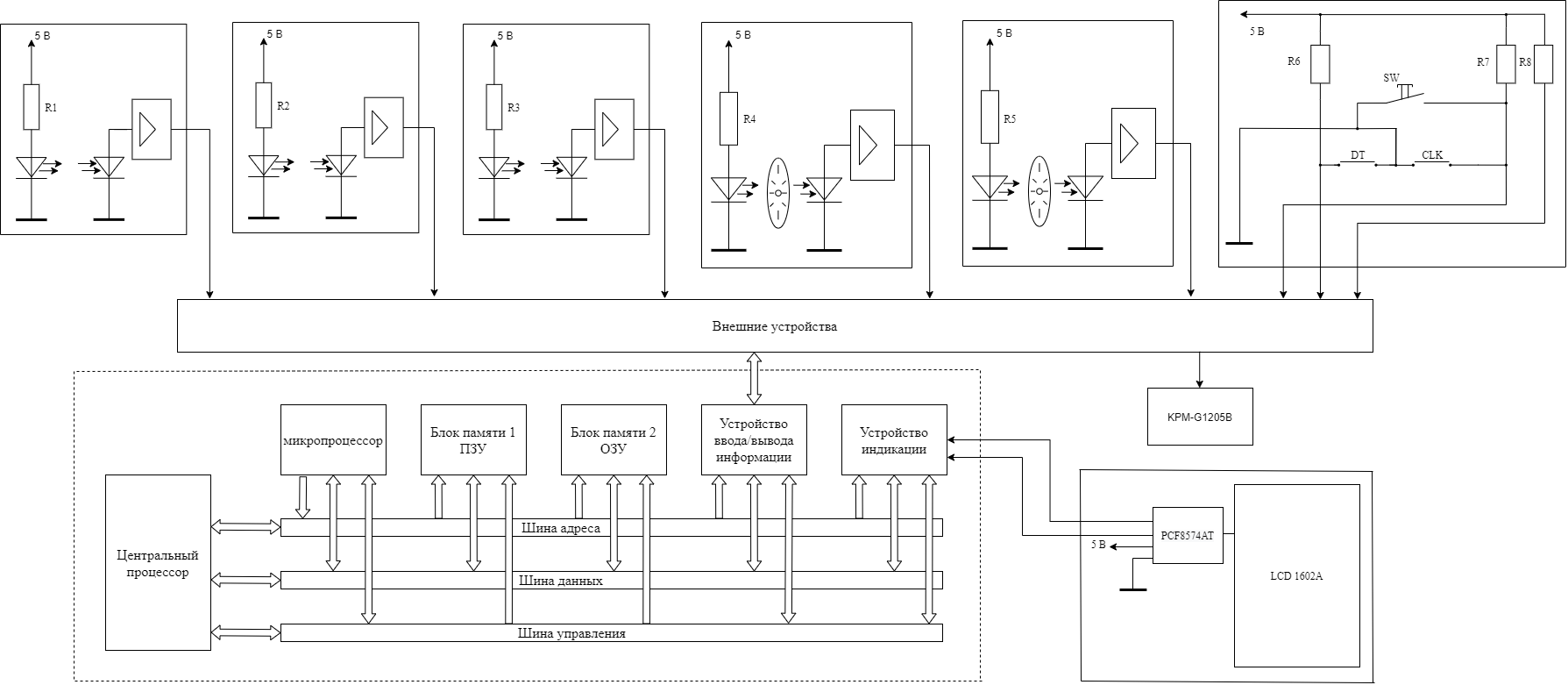


Рисунок 2 – функциональная схема разработанного устройства

2.2 Схема электрическая принципиальная

На рисунках 3а, 3б, 3в представлена схема электрическая принципиальная разработанного устройства.

**2.2.1 Схема электрическая принципиальная блока индикации**

На рисунке 3а представлена схема электрическая принципиальная блока индикации. Данный блок представляет собой lcd дисплей 1602А подключенный по интерфейсу I2C через модуль-переходник на основе микросхемы PCF8574AT.

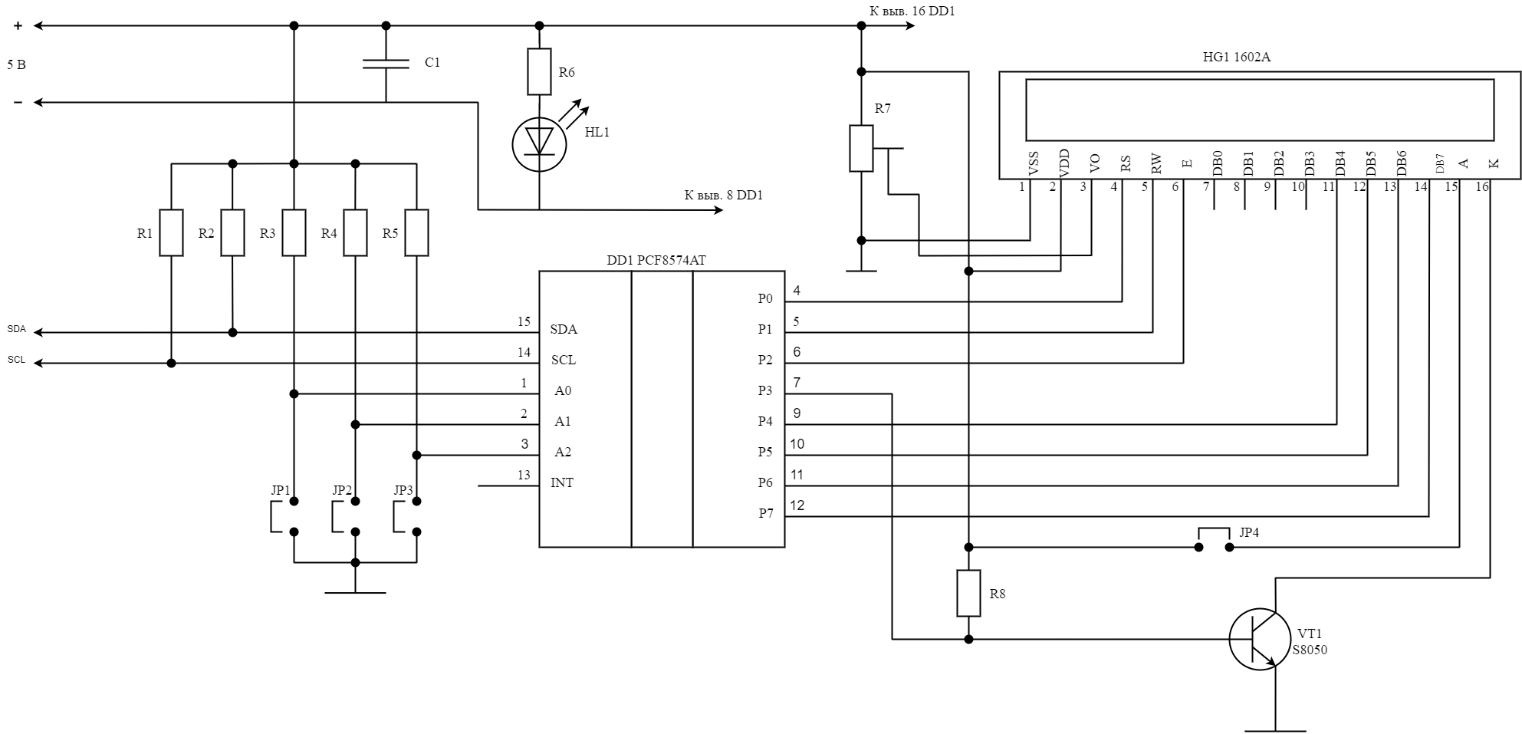


Рисунок 3а — Схема электрическая принципиальная блока индикации.

Для передачи информации на блок индикации используется две линии для передачи данных: SDA (линия данных) и SCL (линия синхронизации). Также имеется подстроечный резистор R7 для регулировки контрастности экрана.

Адрес микросхемы PCF8574AT на шине I2C можно настраивать, старшие 4 бита адреса фиксированы, и равны 0111, младшие 3 бита зависят от состояния входов микросхемы A2-A0. На модуле данные входы подтянуты к высокому уровню, соответственно адрес микросхемы принимает значение 0111111.

Как видно из схемы, к микросхеме подключена только часть линий ввода/вывода дисплея DB7-DB4, это означает, что управление дисплеем возможно только через 4-битный интерфейс. Для ввода инструкции в дисплей требуется 2 тактовых импульса на линии E, то есть последовательность уровней 1010 (“защелкивание” данных происходит по спаду уровня), в итоге необходимо записать в микросхему 4 байта для одной инструкции.

Сначала передается старший полубайт инструкции с битом E=1, затем то же самое с битом E=0, при этом в дисплей передается первая половина инструкции. Далее таким же образом передается вторая половина (младший полубайт).

Для управления подсветкой дисплея, на плате модуля установлен транзистор, подключенный к линии P3 микросхемы. Таким образом, 3-й бит в байте данных управляет подсветкой, 0 – выключена, 1 – включена.

**2.2.2 Схема электрическая принципиальная датчиков линейного и углового перемещения.**

Поскольку схемы датчиков линейного и углового перемещения по однотипным типовым схемам на операционном усилителе, то принципиальная схема электрическая будет рассмотрена на примере датчика линейного перемещения. Данная схема представлена на рисунке 3б.

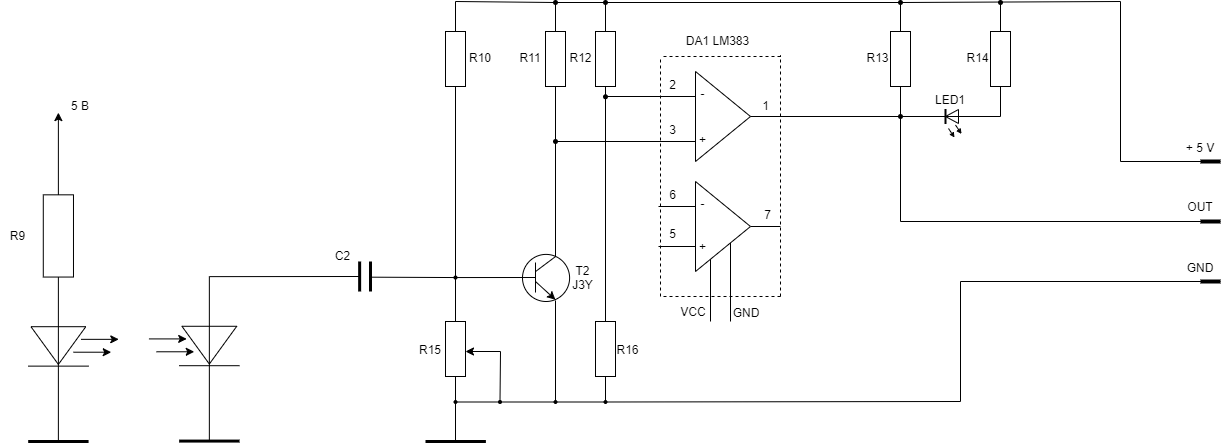


Рисунок 3б — Схема электрическая принципиальная усилителя датчиков внешних устройств.

Схема электрическая принципиальная датчика линейного перемещения, построена на оптопаре состоящей из пары фотодиодов FD1 AL106 (излучающий) и FD2 AL107 (принимающий), а также на готовом операционном усилителе KY-037. Сигнал от FD1 прошедший механический энкодер, представляющий собой диск с прямоугольными отверстиями и напечатанный на 3D принтере, через FD2 через конденсатор С2 поступает на однокаскадный усилитель. Данный усилитель представлен в виде двух компараторов LM393. После с операционного усилителя сигналы ТТЛ уровня поступают на советующий цифровой вход блока управления и измерения, представленного микроконтроллером.

**2.2.3 Схема электрическая принципиальная блока выбора режима работы**

Схема рассматриваемого блока представлена на рисунке 3в.

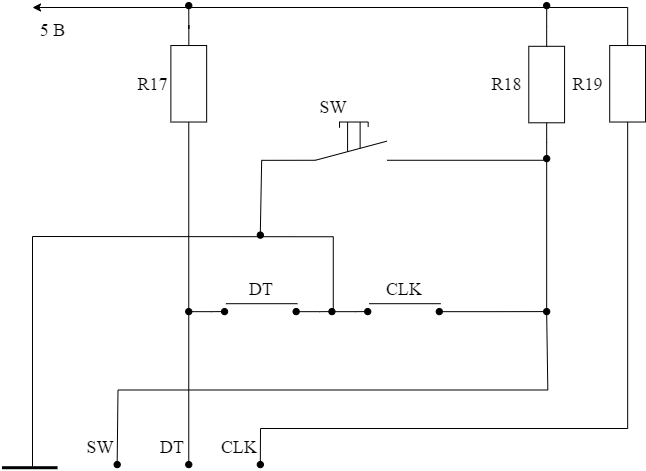


Рисунок 3в - схема электрическая принципиальная блока выбора режима работы.

Устройство представлено в виде электромеханического энкодера. В его состав входят два скользящих контакта CLK и DT, а также кнопка SW подключенной по схеме с подтягивающим резистором. При вращении энкодера в одном из направлений на контактах CLK и DT возникают прямоугольные импульсы, сдвинутые друг относительно друга на фазовый угол 90°. При нажатии на кнопку вывод замыкается на землю, тем самым создавая сигнал низкого уровня.

**2.2.4 Схема электрическая принципиальная блока измерения и управления**

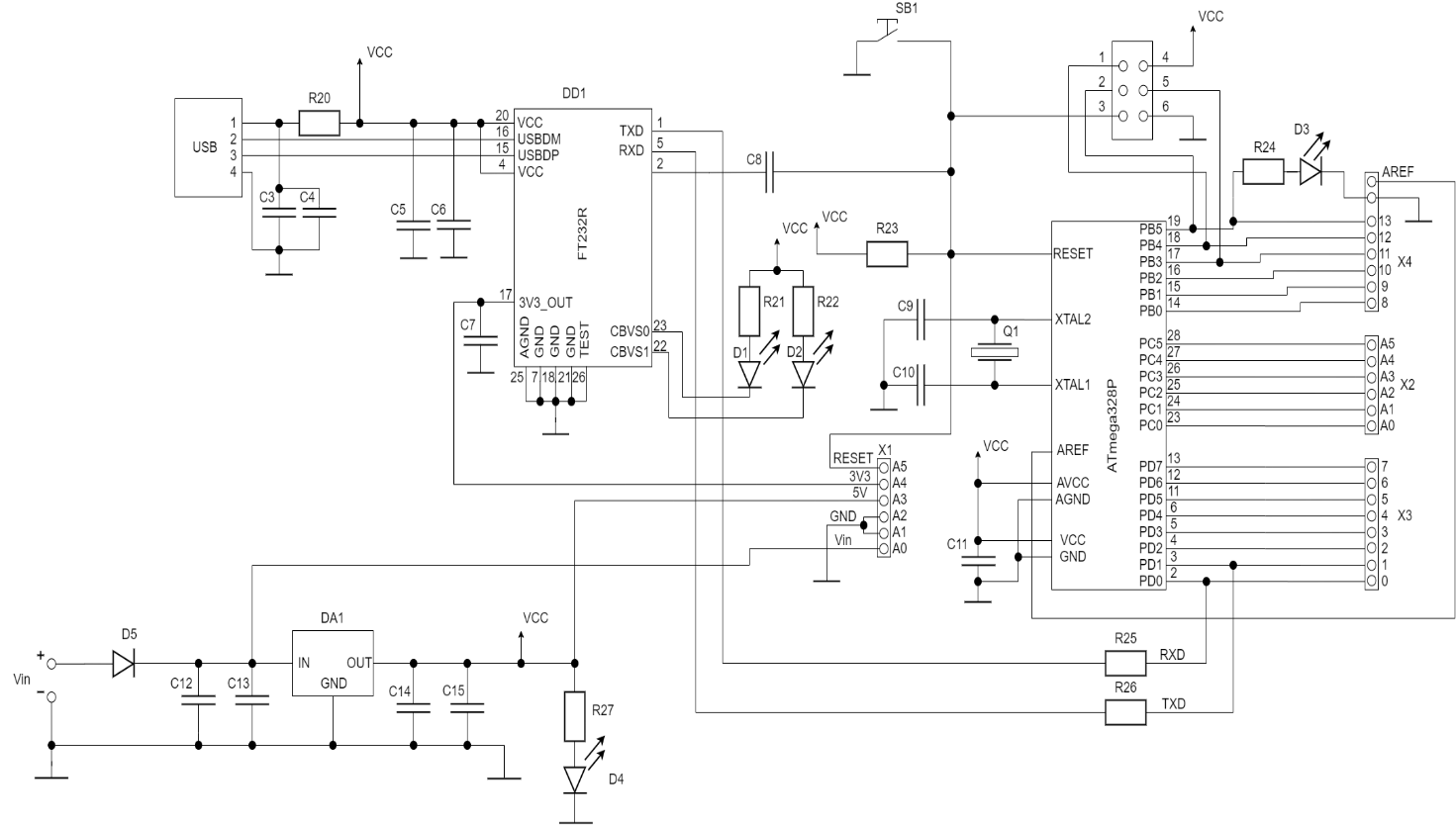
Схема блока измерения и управления представлена на рисунке 3г.

Рисунок 3г — схема электрическая принципиальная блока выбора режима работы.

Поскольку объем данной дипломной работы ограничен, а схемы электрическая структурная, функциональная данного блока рассмотрены достаточно подробно. Схема электрическая принципиальная блока измерения и управления в данной работе не рассматривается

**2.3 Конструкция устройства**

Устройство линейного и углового перемещения на микроконтроллере изготовлено в типовом корпусе G1183BUL.

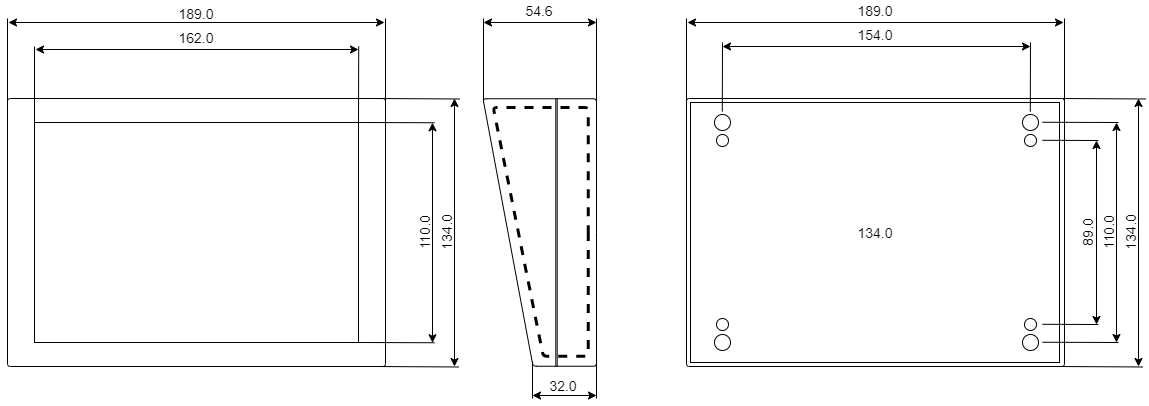


Рисунок 4 – Корпус G1183BUL.



Рисунок 5 -Разработанное и изготовленное устройства в составе модернизированной лабораторной установке «машина Атвуда»

3 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

3.1 Структура программного кода

Поскольку для реализации задачи поставленной в данной дипломной работе был выбран микроконтроллер Atmega328P, то следует сказать, что написание программного кода для данного микроконтроллера осуществляется с помощью языка С++. Полный листинг программного кода представлен в приложении З.

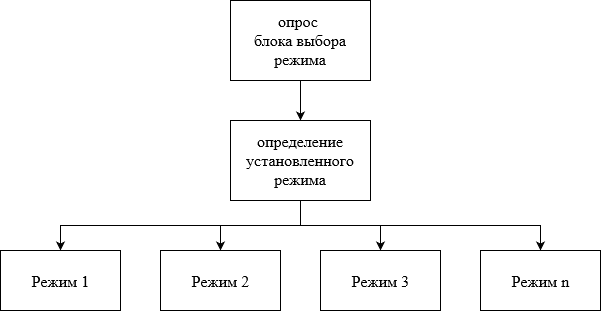


Рисунок 6 – структура программного кода

Как видно из рисунка 4, разработанный блок является многофункциональным. Это достигается за счет возможности выбора режима и написания большого количества программных модулей, реализующих нужный функционал для определенных измерений.

Логика описанного ниже программного кода представлена в виде блок-схемы. Данная блок-схема представлена в приложении Д.

Программирование микроконтроллера начинается с создания двух функций с именами *setup()* и *loop()*. Функция *setup()* выполняется лишь один раз и реализует первоначальная настройка основных компонентов используемых в дальнейшем в программном коде. По завершению, функция *setup()* передает управление микроконтроллером функции *loop()*, которая реализует основную логику выполняемую данным микроконтроллером [3].

void setup() {

/\*настройка блока индикации \*/

lcd.*init();*

lcd*.backlight();*

/\*настройка энкодера\*/

*pinMode(pin\_DT, INPUT);*

*pinMode(pin\_CLK, INPUT);*

/\*настройка кнопки энкодера\*/

*pinMode(pin\_Btn, INPUT\_PULLUP);*

/\*настройка звукового излучателя \*/

*pinMode(pin\_Sound\_Signal, OUTPUT);*

/\*настройка датчиков линейного перемещения\*/

*pinMode(pin\_Optocoupler, INPUT);*

*pinMode(pin\_Impulse\_Counter, INPUT);*

}

Поскольку в данной работе для подключения внешних устройств используется двунаправленная шина данных, в начале программного кода необходима установка режима работы используемых выводов. Для этих целей используется функция *pinMode(pinNumber, pinType)*.

В качестве параметра pinNumber используется номер вывода. Параметром pinType является константа, указывающая на режим работы. Параметр pinType может принимать следующие значения:

*INPUT* – режим в котором шина микроконтроллера поддерживается в высокоимпедансном состоянии, для работы с внешними источниками сигналов.

*OUTPUT* - режим в котором шина микроконтроллера поддерживается в низкоимпедансном состоянии, при котором на внешнее устройство выдается максимально возможный ток.

*INPUT\_PULLUP* – режим аналогичен режиму INPUT, разница заключается в том, что получаемый сигнал инвертируется.

void *loop()* {

/\*функция выбора режима работы\*/

*choiseOfMode();*

}

Следующим этапом является вызов функции *choiseOfMode()*.Данная функцияотвечает за вызов одного из 5 возможных режимов. Код данной функции представлен ниже.

void *choiseOfMode()* {

/\*Флаг для перехода в выбранный режим по нажатию кнопки энкодера\*/

isEncoderButtonPressed = false;

switch (*getEncoderState*()){

case eNone:

return;

case eButton:

isEncoderButtonPressed = true;

break;

case eLeft:

counter--;

*printOption(counter);*

break;

case eRight:

 counter++;

*printOption(counter);*

break;

}

/\*Переход в выбранный режим если кнопка энкодера нажата\*/

if (isEncoderButtonPressed){

*goToOption(counter);*

}

}

Для использования данной функции необходимы следующие функции: *getEncoderState()*, *printOption()*, *goToOption()*.

Первой рассматриваемой функцией является *getEncoderState()*.

eEncoderState GetEncoderState() {

/\*Считываем состояние энкодера\*/

eEncoderState Result = eNone;

CurrentTime = millis();

if (CurrentTime - LastTime >= 5) {

/\*Считываем не чаще 1 раза в 5 мс для уменьшения ложных срабатываний\*/

LastTime = CurrentTime;

if (digitalRead(pin\_Btn) == LOW) {

if (ButtonPrev) {

Result = eButton; // Нажата кнопка

ButtonPrev = 0;

}

}

else {

ButtonPrev = 1;

EncoderA = digitalRead(pin\_DT);

EncoderB = digitalRead(pin\_CLK);

if ((!EncoderA) && (EncoderAPrev)){

/\*Сигнал A изменился с 1 на 0\*/

if (EncoderB)

Result = eRight; // B=1 => энкодер вращается по часовой

else

Result = eLeft; // B=0 => энкодер вращается против часовой

}

EncoderAPrev = EncoderA; // запомним текущее состояние

}

}

return Result;

}

Поскольку данная функция позволяет определить положение энкодера. Данная функция опрашивает энкодер каждые 5 миллисекунд возвращая одно из четырех значений:

eNone – энкодер не изменил своего положения.

eLeft – энкодер совершил поворот влево.

eRight – энкодер совершил поворот вправо.

eButton – кнопка энкодера нажата.

Второй рассматриваемой функцией является *printOption()*. Принимающая в качестве параметра целое значение указывающее на выбранный режим. Данная функция выводит название режима, в соответствии выбранному режиму, в виде текстовой информации на блок индикации.

Оставшейся функцией является *goToOption()*. Данная функция, как и уже описанная, принимает значение выбранного режима, после чего контроль переходит к выбранному режиму.

3.2 Описание режимов программного кода

Поскольку программная часть основных режимов устройства идентичны, в данной главе будет рассмотрен программный код режима измерения линейного перемещения.

Логика описанного ниже программного кода представлена в виде блок-схем. Блок-схема режима измерения линейного перемещения представлена на рисунке 7. Кроме того, в приложении Ж представлена блок-схема работы режима подсчета периодов колебания.

После выбора режима измерения линейного перемещения, работы функция *goToOption()* предает управлениемикроконтроллером первому модулю.

void *firstMove()* {

/\*Предварительная настройка необходимых переменных\*/

const byte stepForLinearMovement = 5;

long timeForStartCounting = 0;

long distanceTravelTime = 0;

int needDistance = 0;

int last = counter;

/\* Номер аналогового портов для подачи разрешающего сигнала с ДУ \*/

int pinDistanceControlStart = 2 ;

do {

/\*Вывод задаваемого расстояния\*/

if (last != counter) {

/\*запрет на задание отрицательно расстояния\*/

counter = counter <= 0 ? 0 : counter;

last = counter;

needDistance = last \* stepForLinearMovement;

*printLinearMovementDistance(needDistance);*

}

/\*Функция измерения. Данная функция начинает работать при получении разрешающего сигнала с ДУ\*/

if (*isAnalogButtonPressed(pinDistanceControlStart)* && needDistance > 0) {

while (true) {

/\*Ожидание старта отсчёта\*/

if (isPinHigh(pin\_Optocoupler)) {

distanceTravelTime = *linearDistanceCalculation(needDistance);*

*printResultDistanceTravelTime(distanceTravelTime, needDistance);*

delay(2000);

break;

}

}

}

*encoder();*

} while (!isEncoderButtonPressed);

}

long startLinearDistanceCalculation(int needLinerDistance){

/\*Предварительная настройка необходимых переменных\*/

long timeForStartCounting = 0;

long distanceTravelTime = 0;

/\*коэффициент для подсчета пройденного расстояния\*/

double coefficientForSensor = 0.097;

while (needLinerDistance > 0) {

timeForStartCounting = micros();

if (isPinHigh(pin\_Impulse\_Counter)) {

needLinerDistance = needLinerDistance - coefficientForSensor;

distanceTravelTime = micros() - timeForStartCounting;

printResultDistanceTravelTime(distanceTravelTime, needLinerDistance);

}

}

return distanceTravelTime;

}

Для функционирования рассмотренного выше листинга кода, необходимы следующие функции: *printLinearMovementDistance(), encoder(), micros(), printResultDistanceTravelTime(), startLinearDistanceCalculation().*

Функция *printLinearMovementDistance()* принимает в качестве параметра значение задаваемого расстояния, после чего выводит данную информацию на блок индикации.

Функция *printResultDistanceTravelTime ()* принимает в качестве параметров время, за которое происходило перемещение, а также пройденное расстояние. Данная функция аналогично предыдущей, а именно является функцией, обеспечивающей вывод результатов измерения на блок индикации.

Функция *encoder()* во многом похожана функцию *choiseOfMode()* поскольку также взаимодействует с блоком выбора режима.

Функция *micros()* используется для подсчета времени требуемого для перемещения физического тела, посколькувозвращает количество микросекунд с момента начала выполнения текущей программы.

Последней из рассмотренных функций будет *linearDistanceCalculation()*. Данная функция принимает значение заданного расстояния. После чего ожидает получения импульсов с датчика линейного перемещения. По окончанию линейного перемещения, функция возвращает время, за которое физическое тело переместилось на заданное расстояние.

После того, как данный режим завершиться, устройство можно использовать повторно или же выбрать другой режим работы.

4 ТЕХНИКА БЕЗОПАСНОСТИ

4.1 Техника безопасности при проведении электромонтажных работ

К самостоятельному выполнению работ допускаются лица, достигшие 18 - летнего возраста, прошедшие медицинский осмотр, инструктаж по технике безопасности (не реже одного раза через каждые 3 месяца) и имеющие 1 квалификационную группу по электробезопасности.

При пайке, лужении оловянно-свинцовыми припоями производить работу только под вытяжной вентиляцией, установленной на рабочем месте.

При пайке, лужении существует опасность:

1. Попадания припоя на кожу рук, в глаза;
2. Попадания вредных веществ, содержащих свинец, в дыхательные пути;
3. Поражения электротоком.

Выполнять работы разрешается только на исправном оборудовании, исправным инструментом, приспособлением, приборами и т.д. и применять их только по назначению.

Выполнять работы нужно те, которые поручены работающему и правила безопасности, выполнения которых усвоены работающим.

Содержать рабочее место в чистоте и порядке, не допускать наличие на рабочем месте посторонних предметов.

Постоянно следить за работой вытяжной вентиляции.

На участке должна находиться медицинская аптечка, содержащая необходимые средства оказания первой помощи.

Для размещения на рабочих местах и транспортировки по цеху изделий, деталей, а также отходов производства, должна применяться соответствующая тара, изготовленная по чертежам и имеющая четкую надпись.

Хранить инструмент необходимо в ящике монтажного стола, соблюдая определенный порядок.

Нагревательные приборы: паяльники, обжигалки убирать в стол холодными.

Заметив нарушения правил техники безопасности другим сотрудником, предупредите его о недопустимости подобных случаев.

При переходе с одной работы на другую, работающий обязан получить инструктаж по новой работе и пройти стажировку под руководством опытного работника.

При несчастном случае или общем недомогании необходимо сообщить руководителю участка, обратиться за помощью в медпункт.

При получении больничного листа в связи с несчастным случаем на производстве, следует сообщить самому или через товарищей администрации подразделения.

Необходимо сообщать руководителю участка обо всех неисправностях оборудования, которые могут быть причиной травматизма и не приступать к работе до их устранения.

При возникновении пожара следует немедленно сообщить в пожарную охрану, приступить к тушению огня имеющимися средствами пожаротушения, сообщить руководителю подразделения.

За несоблюдение данной инструкции виновные несут ответственность в соответствии с действующим законодательством.

4.2 Требования безопасности перед началом работы

Надеть технологическую одежду, предназначенную для работы (халат, шапочку, тапочки), чтобы она не стесняла движений работающего, не имела свисающих концов и соответствовала требованиям электровакуумной гигиены.

Необходимо привести в порядок рабочее место (удаление ненужных предметов, рациональное и безопасное размещение необходимого для работы инструмента, приспособления, деталей и т. д.).

Проверить наличие и действие вытяжной вентиляции.

Необходимо проверить освещение рабочего места, а также рациональное размещение светильников.

Необходимо проверить правильность заземления измерительных и нагревательных приборов.

Внешним осмотром проверить:

1. исправность инструмента, приспособлений, приборов;
2. наличие и исправность защитного заземления;
3. маркировку инструмента или соответствие их ГОСТу.

Расположить с правой стороны флюс, припой на достаточном расстоянии во избежание попадания в глаза расплавленного припоя и канифоли.

В случае обнаружения каких - либо неисправностей сообщить мастеру или руководителю участка и приступить к работе только после их устранения.

4.3 Требования безопасности во время работы

Содержать рабочее место в чистоте и порядке в течение всего рабочего времени.

Пайку изделий производить только под вытяжной вентиляцией.

Во время работы быть внимательным, не отвлекаться посторонними разговорами и делами.

Выполнять только ту работу, которая поручена администрацией и согласно технического процесса.

Не допускается самовольное исправление каких - либо неисправностей электрооборудования, паяльников, обжигалок, а также самовольная замена перегоревших ламп и предохранителей.

Применять в работе только исправные инструменты, не допускать применения случайных приспособлений, не указанных в маршрутных картах.

Не курить и не принимать пищу на рабочих местах.

При перерывах в работе нельзя бегать, спешить, нужно проявлять осторожность при спуске с лестницы.

Напряжение паяльника не должно превышать 36В, обжигалки - 6 В.

Канифоль при пайке необходимо отставлять как можно дальше от лица - на расстоянии, удобном для монтажных работ.

Во время пайки проводов, радиоэлементов, необходимо пользоваться пинцетом, не допускать разбрызгивания капель припоя.

В случае заболевания или получения травмы работу прекратить, сообщить об этом руководителю участка и обратиться за медицинской помощью.

При получении ожога рук, лица во время работы обработать пораженные места средствами первой помощи, имеющимися в аптечке, находящейся на участке и обратиться в медпункт.

Необходимо соблюдать правила личной гигиены, применять профилактические средства, предупреждающие заболевания, отравления (пасты, дезинфицирующие средства, полоскания рта, чистка зубов и пр.).

При возникновении пожара немедленно выключить все электрооборудование и приборы, сообщить руководителю участка, цеха и принять меры к тушению очага загорания первичными средствами пожаротушения.

4.4 Требования безопасности по окончании работы

* Убрать рабочее место, произвести влажную уборку теплой водой.
* Остудить, очистить, протереть, проверить исправность инструмента, приспособления и убрать, выключить освещение.
* Отключить оборудование от электросети.
* Вымыть руки теплой водой с 1% раствором уксусной кислоты.
* О всех обнаруженных замечаниях сообщить руководству участка.
* Убрать спецодежду в отведенное для этого место.
* Неиспользованные комплектующие ЭРИ, материалы и изделия сдать мастеру.

4.5 Требования безопасности в аварийных ситуациях

В случае возникновения аварийной ситуации на рабочем месте или на другом объекте немедленно поставить в известность руководителя.

В случае угрозы жизни немедленно покинуть опасную зону: при необходимости отключить электроэнергию; прекратить все работы.

Оказывающий помощь должен уметь:

* Определять степень тяжести пострадавшего и в какой первой помощи он нуждается;
* Обеспечить свободный поход верхних дыхательных путей;
* Выполнять искусственное дыхание “изо рта в рот”, “изо рта в нос” и закрытый (непрямой) массаж сердца и оценить их эффективность;
* Временное остановка крови путем наложения жгута;
* Накладывание бинтовой повязки при ранении, ушибе.
* Первая помощь пострадавшим от электрического тока или других несчастных случаях:
* Соблюдать меры личной безопасности, освободить пострадавшего от действия электрического тока;
* Устранить воздействие на организм повреждающих факторов, угрожающих здоровью и жизни пострадавшего, оценить состояние пострадавшего;
* Незамедлительно вызвать врача.

При неудовлетворительном состоянии незамедлительно произвести искусственное дыхание и непрямой массаж сердца. Искусственное дыхание и массаж сердца делать до прибытия врача.

При порезах обязательно обработать рану йодом или зеленкой.

При отравлении газами, вывести пострадавшего из загазованной зоны, расстегнуть одежду, создать приток свежего воздуха, вызвать врача.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной дипломной работы, были проанализированы основные варианты микроконтроллеров, которые могли бы быть использованы в разработке конечного устройства. Также следует отметить, что был проделан большой объем работ как в процессе проектирования и разработки схем электрических: структурной, функциональной, и принципиальной, которые полностью отвечают задачам, поставленным в данной дипломной работе.

Кроме того, было разработано программное обеспечение, достоинством которого является использование так называемых модулей. Это позволяет с легкость добавлять новый функционал.

Как результат, разработанное устройство является устройством линейного и углового перемещения на микроконтроллере имеющее хорошие технические характеристики для работы в составе лабораторных установок лаборатории «механика» кафедра общей физики.

Поскольку разработанное устройство является универсальным, благодаря нему можно в значительной степени упростить, ускорить, а кроме того увеличить точность большого количества исследовательских, а также различных лабораторных работ при выполнении которых, требуется измерение линейного или же углового перемещения. Со следующими характеристиками: дискретность измерения углового перемещения 0.1°, дискретность измерения линейного перемещения 1 мм.

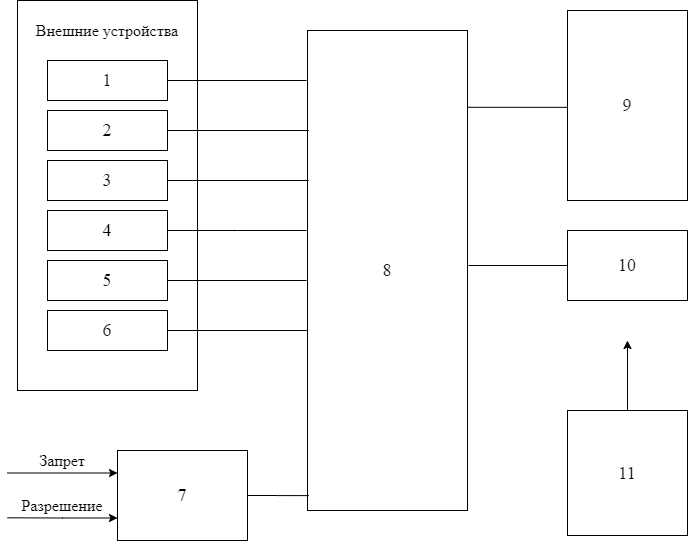
Кроме того, стоит отметь, что данное устройство может быть использовано в комплекте с секундомером/таймером, что позволит еще больше расширить диапазон выполняемых лабораторных работ. Примером совместного использования разработанного устройства и таймера с дистанционным управлением является модернизированная лабораторная установка «машина Атвуда». В данной лабораторной установке, устройство используется в режиме измерения линейного перемещения.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

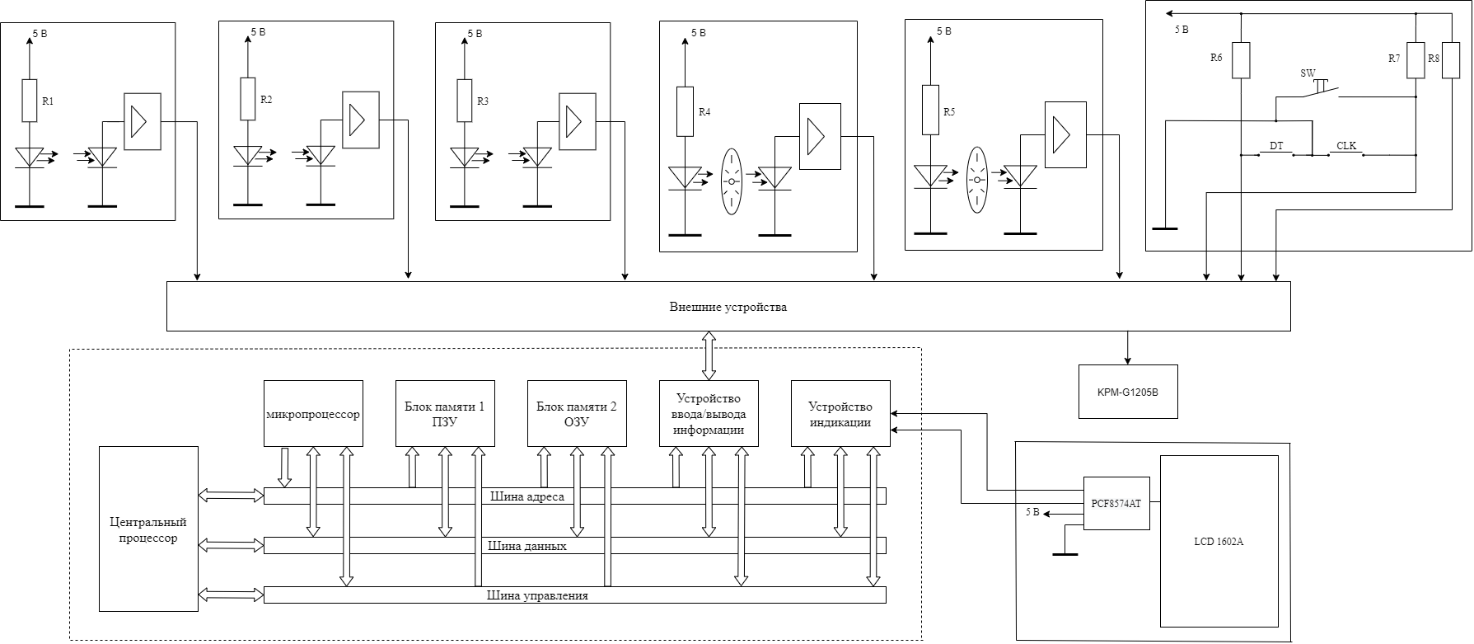
1. Гребнев В.В. Микроконтроллеры семейства AVR фирмы Atmel: ИП РадиоСофт, 2002. - 170c.
2. Голубцов М.С. Микроконтроллеры ARM. От простого к сложному: СОЛОН-Пресс, 2003. – 288c.
3. Литвиненко Н. А. Технология программирования на С++: Изд-во БХВ-Петербург, 2010 – 281с.
4. Коротаев В.В. Оптико-электронные преобразователи линейных и угловых перемещений: Национальный исследовательский институт Санкт-Петербург, 2012 – 117с.
5. Оптика и атомная физика: Лабораторный практикум по физике/ Под рел.Р.И.Солоухина. Новосибирск: Наука. Сиб. ото-пне, 1983. 384 с.
6. Физический практикум/ Под ред.В.И.Ивероновой1 В 2 ч. М.: Физматгиз, 1967 - 1968. Ч.1-2.
7. ЭВМ в курсе общей физики.Изд-во Моск.ун-та, 1982. 230 с.
8. Методы физических измерений: Лабораторный практикум по физике/Под ред.Р.И.Солоухяна.
9. https://www.alldatasheet.com/datasheetpdf/pdf/506518/STMICROELECTRONICS/STM8L151C8T6.html Дата доступа 11.04.2020.
10. <https://web.mit.edu/6.115/www/document/16f628.pdf> Дата доступа 29.05.2020.
11. <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf> Дата доступа 13.03.2020.
12. https://ru.wikipedia.org/wiki/AVR Дата доступа 06.04.2020.
13. https://ru.wikipedia.org/wiki/ARM Дата доступа 06.04.2020.
14. Трамперт, В. Измерение, управление и регулирование с помощью AVR- микроконтроллеров.: Перевод с нем. / В. Трамперт. - К.: “МК-Пресс”, 2006. - 208с.
15. https://t-31.ru/podklyuchenie-lcd-1602-i2c Дата доступа 03.05.2020.
16. http://easyelectronics.ru/avr-uchebnyj-kurs-arxitektura.html#more-17 Дата доступа 18.05.2020.
17. https://www.instructables.com/id/LCD-display-I2C-with-PCF8574A Дата доступа 12.05.2020.
18. http://geekmatic.in.ua/arduino\_lesson\_121 Дата доступа 02.05.2020.
19. https://electropeak.com/learn/how-to-use-ky-037 Дата доступа 05.06.2020.
20. https://www.chipdip.by/product/g1183b Дата доступа 10.02.2020.

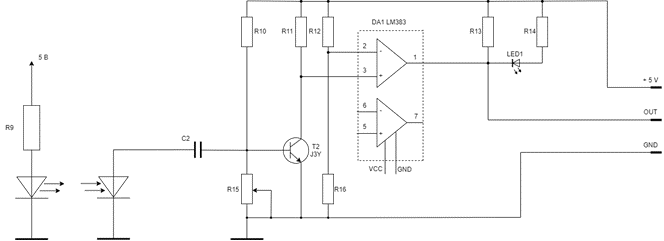
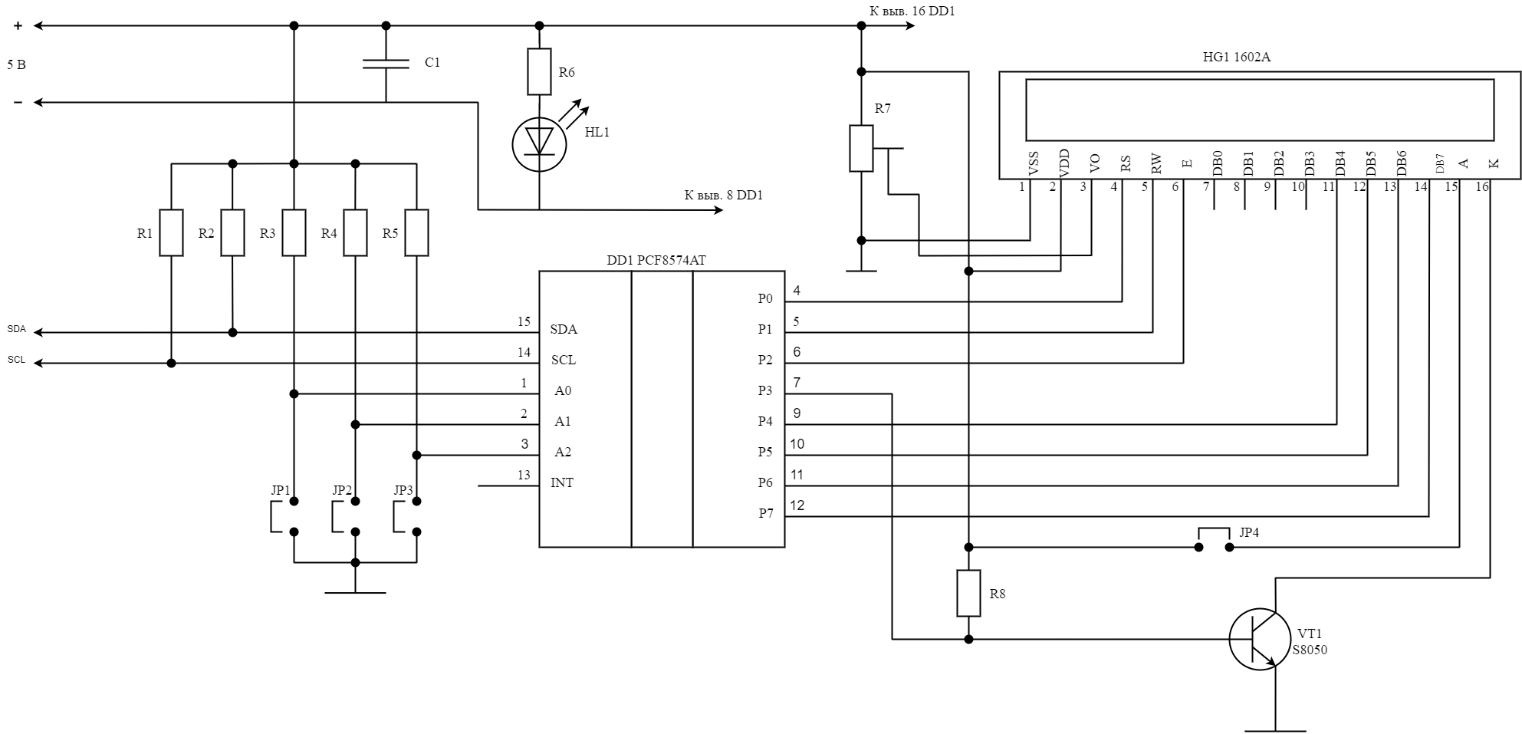
ПРИЛОЖЕНИЯ

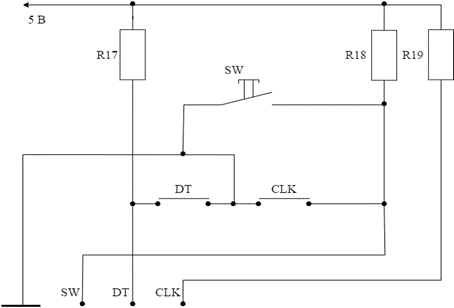
**Приложение А – Схема электрическая структурная**



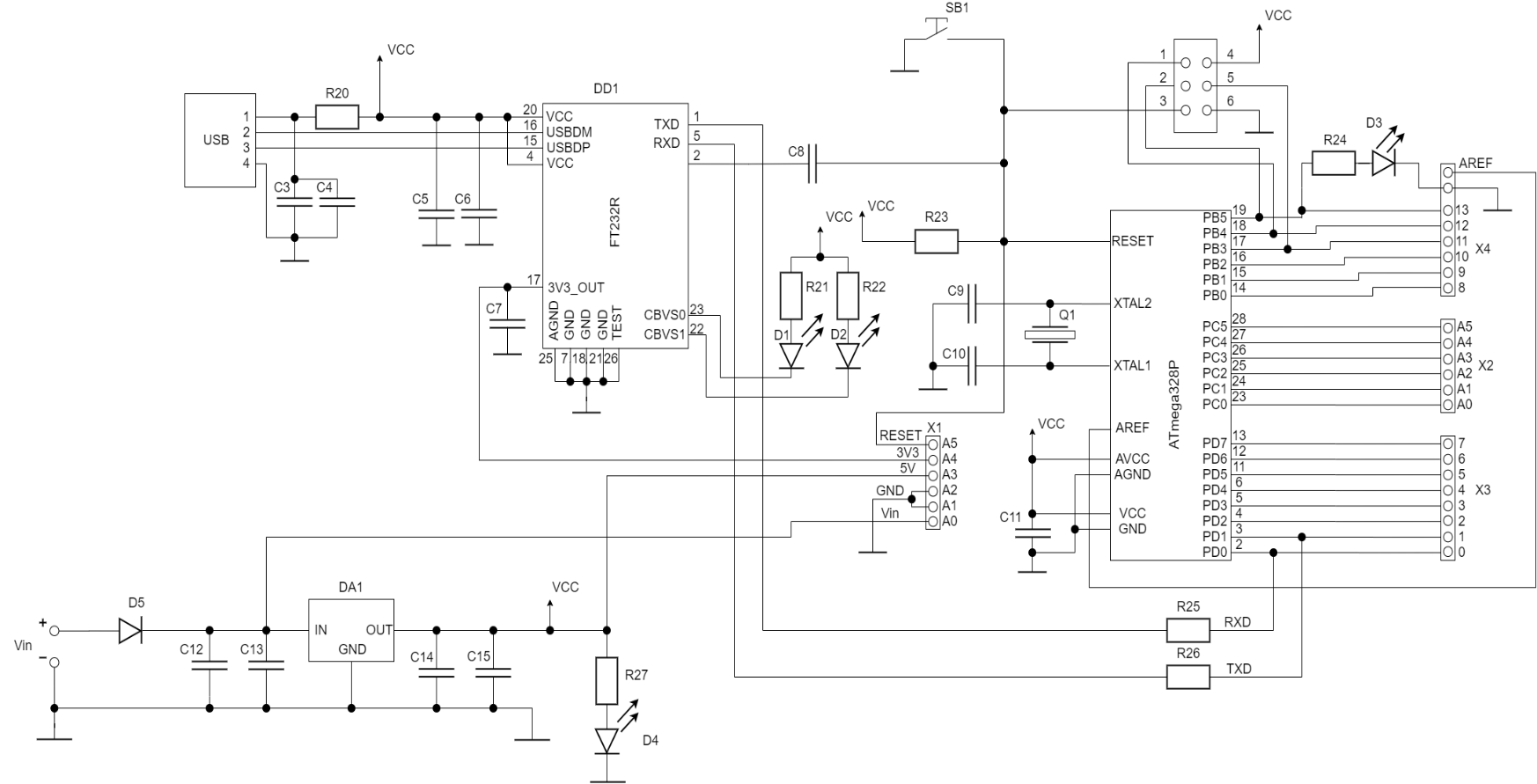
**Приложение Б - Схема электрическая функцианальная**

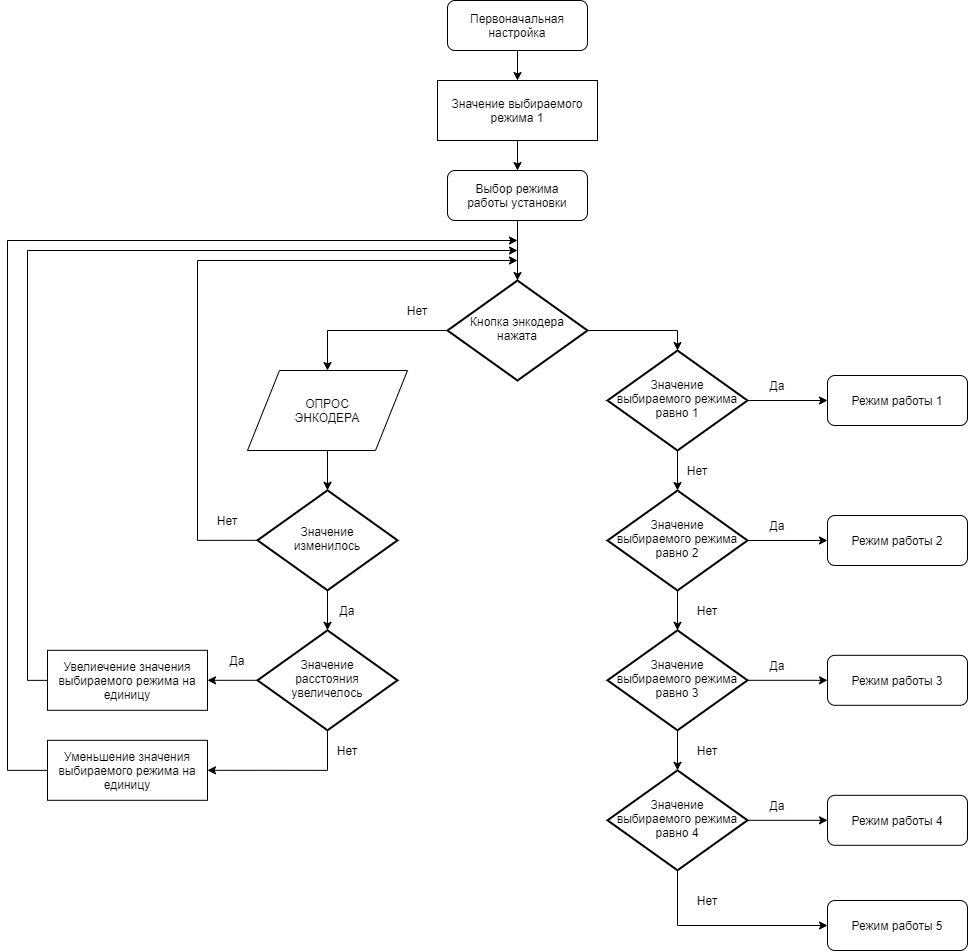


**Приложение В - Схема электрическая принципиальная**

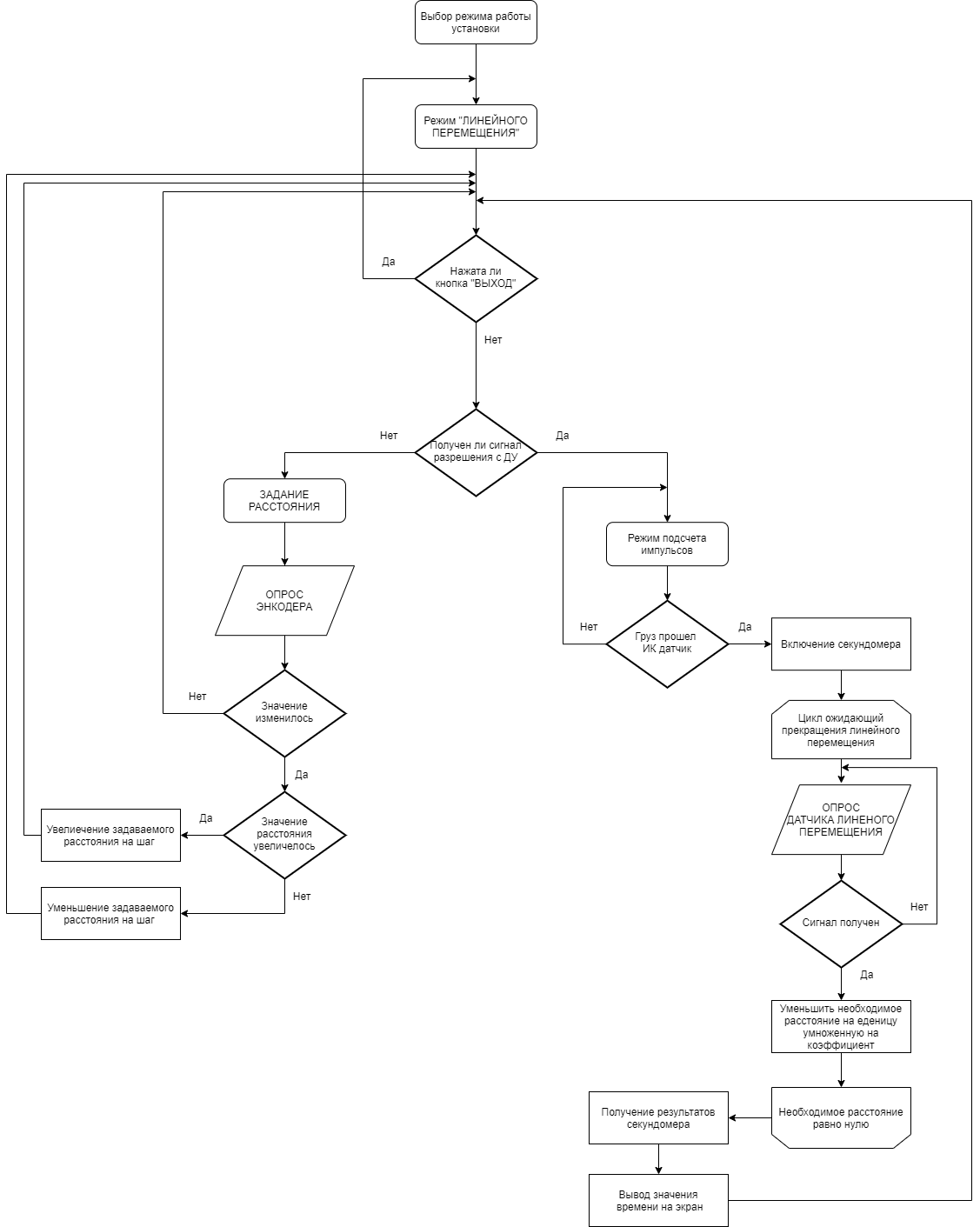
**Приложение В - Схема электрическая принципиальная**

**Приложение В – Схема электрическая принципиальная**

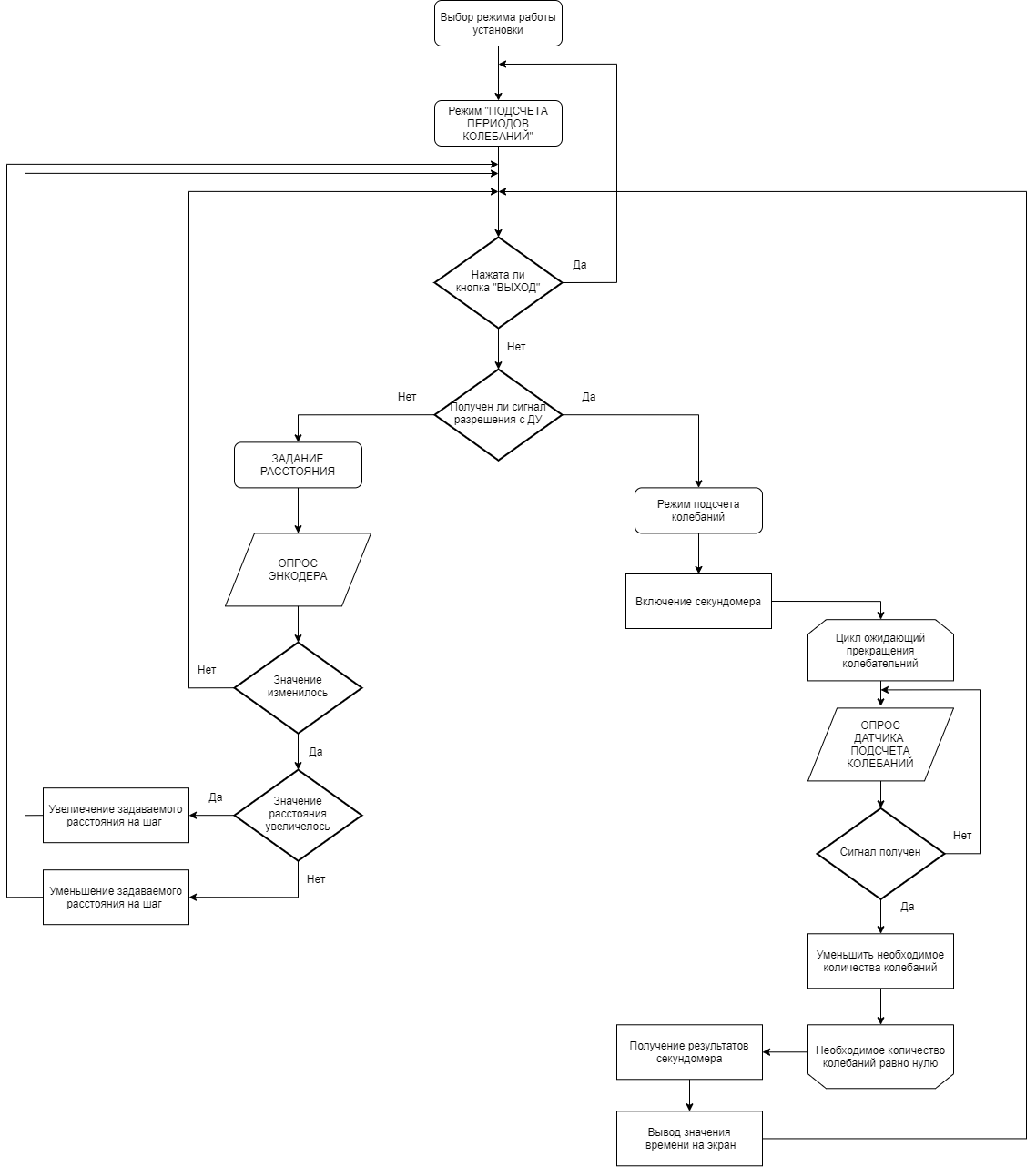


**Приложение Д - Блок-схема выбора режима работы устройства.**

**Приложение Е - Блок-схема режима линейного перемещения**



**Приложение Ж - Блок-схема режима подсчета периодов колебаний***.*

****

**Приложение З – Программный код**

#include <LCD\_1602\_RUS.h>

LCD\_1602\_RUS lcd(0x27, 16, 2);

#define pin\_CLK 4

#define pin\_DT 2

#define pin\_Btn 9

const int pin\_Optocoupler = 5;

const int pin\_Impulse\_Counter = 6;

const int pin\_Sound\_Signal = 3;

bool isNeedPreView = true;

bool isEncoderButtonPressed = false;

unsigned long CurrentTime, LastTime;

enum eEncoderState {

eNone,

eLeft,

eRight,

eButton

};

uint8\_t EncoderA, EncoderB, EncoderAPrev;

int8\_t counter = 0;

bool ButtonPrev;

int leftBallValues[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

int rightBallValues[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

long leftBallTimeValues[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

long rightBallTimeValues[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

eEncoderState GetEncoderState() {

eEncoderState Result = eNone;

CurrentTime = millis();

if (CurrentTime - LastTime >= 5) {

LastTime = CurrentTime;

if (digitalRead(pin\_Btn) == LOW) {

if (ButtonPrev) {

Result = eButton; // Нажата кнопка

ButtonPrev = 0;

}

}

else {

ButtonPrev = 1;

EncoderA = digitalRead(pin\_DT);

EncoderB = digitalRead(pin\_CLK);

if ((!EncoderA) && (EncoderAPrev)) {

if (EncoderB)

Result = eRight;

else

Result = eLeft;

}

EncoderAPrev = EncoderA;

}

}

return Result;

}

void setup() {

Wire.begin();

lcd.init();

lcd.backlight();

pinMode(pin\_DT, INPUT);

pinMode(pin\_CLK, INPUT);

pinMode(pin\_Btn, INPUT\_PULLUP);

pinMode(pin\_Sound\_Signal, OUTPUT);

Serial.begin(9600);

counter = 0;

pinMode(pin\_Optocoupler, INPUT);

pinMode(pin\_Impulse\_Counter, INPUT);

}

void loop(){

if (isNeedPreView) {

prtintTitle();

printOption();

isNeedPreView = false;

}

if (isEncoderButtonPressed) {

goToOption();

isNeedPreView = true;

}

switch (GetEncoderState()){

case eNone:

return;

case eButton:

isEncoderButtonPressed = true;

break;

case eLeft:

counter--;

prtintTitle();

printOption();

isEncoderButtonPressed = false;

break;

case eRight:

counter++;

prtintTitle();

printOption();

isEncoderButtonPressed = false;

break;

}

delay(100);

}

void encoder(){

switch (GetEncoderState()) {

case eNone:

isEncoderButtonPressed = false;

return;

case eButton:

isEncoderButtonPressed = true;

break;

case eLeft:

counter++;

isEncoderButtonPressed = false;

break;

case eRight:

counter--;

isEncoderButtonPressed = false;

break;

}

}

void prtintTitle(){

if (counter > 4){

counter = 0;

}

if (counter < 0){

counter = 4;

}

Serial.println(counter);

lcd.setCursor(0, 0);

lcd.clear();

lcd.print(L"PEЖИМ");

}

void printOption(){

lcd.setCursor(0, 1);

if (counter > 4){

counter = 0;

}

if (counter < 0){

counter = 4;

}

switch (counter){

case 0:

lcd.print(L"1. ЛИНЕЙНОЕ");

break;

case 1:

lcd.print(L"2. ПЕРИОД");

break;

case 2:

lcd.print(L"3. УГЛОВОЕ");

break;

}

}

void goToOption(){

switch (counter){

case 1:

linearMovement();

counter = 0;

exitFromOption();

break;

case 2:

angleMovement();

counter = 4;

exitFromOption();

break;

case 3:

oscillation();

counter = 3;

exitFromOption();

break;

}

}

void exitFromOption(){

encoder();

delay(500);

}

void linearMovement(){

const byte stepForLinearMovement = 5; //cm

long timeForStartCounting = 0;

long distanceTravelTime = 0;

int needDistance = 0;

int pinDistanceControlStart = 2 ;

int pinDistanceControlStop = 3 ;

isNeedPreView = true;

counter = 0;

int last = counter;

delay(50);

do {

if (last != counter || isNeedPreView) {

counter = counter <= 0 ? 0 : counter;

last = counter;

needDistance = last \* stepForLinearMovement;

isNeedPreView = false;

printLinearMovementDistance(needDistance);

}

if (isAnalogButtonPressed(pinDistanceControlStart) && needDistance > 0) {

while (true) {

if (isPinHigh(pin\_Optocoupler)) {

distanceTravelTime = startLinearDistanceCalculation(needDistance);

printResultDistanceTravelTime(distanceTravelTime, needDistance);

delay(2000);

break;

}

}

}

encoder();

} while (!isEncoderButtonPressed);

}

void printResultDistanceTravelTime(long timeToPrint, int distanceToPrint){

lcd.setCursor(0, 0);

lcd.clear();

lcd.print(L"ДИСТАНЦИЯ ");

lcd.print(String(distanceToPrint));

lcd.print(L" см");

lcd.setCursor(0, 1);

lcd.print(" ");

long second = timeToPrint / 1000000;

long millisecond = timeToPrint - second \* 1000000;

millisecond = millisecond / 100;

lcd.print(String(second));

lcd.print(".");

lcd.print(String(millisecond));

lcd.print(L" сек");

}

void printLinearMovementDistance(int distanceToPrint){

if (distanceToPrint < 0) {

distanceToPrint = 0;

}

lcd.setCursor(0, 0);

lcd.clear();

lcd.print(L"РАССТОЯНИЕ");

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.print(String(distanceToPrint));

lcd.print(L" см");

}

long startLinearDistanceCalculation(int needLinerDistance) {

long timeForStartCounting = 0;

long distanceTravelTime = 0;

while (needLinerDistance > 0) {

timeForStartCounting = micros();

if (isPinHigh(pin\_Impulse\_Counter)) {

needLinerDistance-- ;

distanceTravelTime = micros() - timeForStartCounting;

printResultDistanceTravelTime(distanceTravelTime, needDistance);

}

}

return distanceTravelTime;

}

void angleMovement() {

isNeedPreView = true;

isEncoderButtonPressed = false;

int lastLeftBallValues = leftBallValues[0];

int lastRightBallValues = rightBallValues[0];

do {

if (isNeedPreView) {

printAngleMovement();

isNeedPreView = false;

}

encoderLeftAngle();

encoderRightAngle();

encoder();

if (lastLeftBallValues != leftBallValues[0] || lastRightBallValues != rightBallValues[0]) {

printInfo(leftBallValues[0], rightBallValues[0]);

lastLeftBallValues = leftBallValues[0];

lastRightBallValues = rightBallValues[0];

}

delay(100);

} while (!isEncoderButtonPressed);

isNeedPreView = true;

}

void encoderLeftAngle(){

int leftPlusPin = 3;

int leftMinusPin = 2;

int step = 1;

if(isAnalogButtonPressed(leftPlusPin)) {

changeLeftBallArray(1 \* step);

}

if(isAnalogButtonPressed(leftMinusPin)) {

changeLeftBallArray(-1 \* step);

}

}

void changeLeftBallArray(int newLeftValue){

leftBallValues[0] = leftBallValues[0] + newLeftValue;

if(leftBallValues[0] < 0) {

leftBallValues[0] = 0;

}

leftBallValues[1] = leftBallValues[0];

leftBallValues[2] = leftBallValues[1];

leftBallValues[3] = leftBallValues[2];

leftBallValues[4] = leftBallValues[3];

leftBallValues[5] = leftBallValues[4];

leftBallValues[6] = leftBallValues[5];

leftBallValues[7] = leftBallValues[6];

leftBallValues[8] = leftBallValues[7];

leftBallValues[9] = leftBallValues[8];

}

void encoderRightAngle(){

int rightPlusPin = 1;

int rightMinusPin = 0;

int step = 1;

if(isAnalogButtonPressed(rightPlusPin)) {

changeRightBallArray(1 \* step);

}

if(isAnalogButtonPressed(rightMinusPin)) {

changeRightBallArray(-1 \* step);

}

}

void changeRightBallArray(int newRightValue){

rightBallValues[0] = rightBallValues[0] + newRightValue;

if(rightBallValues[0] < 0){

rightBallValues[0] = 0;

}

rightBallValues[1] = rightBallValues[0];

rightBallValues[2] = rightBallValues[1];

rightBallValues[3] = rightBallValues[2];

rightBallValues[4] = rightBallValues[3];

rightBallValues[5] = rightBallValues[4];

rightBallValues[6] = rightBallValues[5];

rightBallValues[7] = rightBallValues[6];

rightBallValues[8] = rightBallValues[7];

rightBallValues[9] = rightBallValues[8];

}

void printAngleMovement(){

lcd.setCursor(0, 0);

lcd.clear();

lcd.print(L"ОТКЛОНИТЕ ШАРЫ");

}

void printInfo(long left, long right){

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(L"ЛЕВЫЙ УГОЛ ");

lcd.print(String(left));

lcd.setCursor(0, 1);

lcd.print(L"ПРАВЫЙ УГОЛ ");

lcd.print(String(right));

}

void oscillation() {

long timeForStartCounting = 0;

long countOfOscillations = 0;

int startPin = 3;

int pinImpulse = 2;

isNeedPreView = true;

counter = 0;

int last = counter;

delay(50);

do {

if (last != counter || isNeedPreView) {

counter = counter <= 0 ? 0 : counter;

last = counter;

isNeedPreView = false;

printOscillation(0, counter);

}

if (isAnalogButtonPressed(startPin) && counter > 0) {

timeForStartCounting = micros();

counter \*= 2;

while (counter != 0) {

if (isAnalogButtonPressed(pinImpulse)) {

counter--;

}

if(counter % 2 == 0) {

printOscillation(micros() - timeForStartCounting, counter/2);

} else{

printOscillation(micros() - timeForStartCounting, (counter + 1)/2);

}

delay(100);

}

isNeedPreView = true;

delay(2500);

}

encoder();

} while (!isEncoderButtonPressed);

}

void printOscillation(long time, int numberOfOscillation){

lcd.setCursor(0, 0);

lcd.clear();

lcd.print(L"КОЛЕБАНИЯ ");

lcd.print(" ");

lcd.print(String(numberOfOscillation));

lcd.setCursor(0, 1);

long second = time / 1000000;

long millisecond = time - second \* 1000000;

millisecond = millisecond / 100;

lcd.print(String(second));

lcd.print(".");

lcd.print(String(millisecond));

lcd.print(L" сек.");

}

bool isAnalogButtonPressed(byte analogPinNumber) {

return analogRead(analogPinNumber) < 300;

}

bool isPinHigh(byte pinNumber) {

return digitalRead(pinNumber) == HIGH;

}

void soundSignal(int numbersOfSound ,int timeBetweenSonds) {

const int valueOfSignal = 100;

for (int j = 0; j < numbersOfSound; j++){

analogWrite(pin\_Sound\_Signal, 100);

delay(timeBetweenSonds);

analogWrite(pin\_Sound\_Signal, 0);

delay(timeBetweenSonds);

}

}