МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №41

ОЦЕНКА РЕФЕРАТА

РУКОВОДИТЕЛЬ

доц., канд. тех. наук		О. А. Кононов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

Разработка контроллера для управления лифтом

по дисциплине: ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

РЕФЕРАТ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4711		Б. Р. Хасанов
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

УТВЕРЖДАЮ:
Заведующий кафедрой №41

Зав. каф., д-р техн. наук, проф		Г.А. Коржавин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

по дисциплине Основы микропроцессорной техники

выполняемую в осеннем семестре 2020/2021 учебного года
студенту Хасанову Булату Рифкатовичу
Ф.И.О. студента полностью в родительном падеже

группы 4711 института № 4 Вычислительных систем и программирования

1.Тема работы: Контроллер для управления лифтом

2. Дата выдачи задания «03» сентября 2020 г.

3. Сроки сдачи студентом оконченной работы «24» декабря 2020 г.

4. Исходная технико-экономическая информация по работе: Управление лифтом на базе микроконтроллера arduino

6 Состав и объем работы

6.1 Чертежи, схемы, диаграммы: структурная схема устройства, схема компьютерного моделирования

6.2 Программа расчетов на ЭВМ: Proteus версии не ниже 7.7, Arduino IDE, Keil uVision

6.3 Расчетно-пояснительная записка не менее 30 стр.

Задание принято к исполнению:

Студент		Хасанов Б. Р.
	дата, подпись	ФИО
Руководитель		О. А. Кононов
	дата, подпись	ФИО

РЕФЕРАТ

Отчёт 41 с., 1ч., 21 рис., 1 таблицы, 12 источников, 4 приложения.

ПЛАТФОРМА ARDUINO, ARDUINO UNO, ЛИФТ, PROTEUS, ARDUINO IDE, ATMEGA.

Объектом исследования являются микроконтроллеры платформы Arduino и создание на их базе контроллера для управления лифтом.

Цели работы: исследование особенностей работы с шаговым двигателем, приобретение практических навыков в среде Arduino IDE во время работы с микропроцессором ATmega328P, создание макета контроллера лифта.

Результат: Изучен микроконтроллер ATmega328P, плата Arduino uno, шаговый двигатель, произведено моделирование, а затем сборка макета контроллера лифта.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....5
ОСНОВНАЯ ЧАСТЬ.....7
1 Постановка задачи.....7
2 Выбор и обоснование элементной базы.....8
3 Написание программ и моделирование работы выбранных устройств.....13
3.1 Моделирование работы шагового двигателя.....15
3.2 Моделирование работы семисегментного дисплея.....18
4 Разработка устройства.....22
4.1 Написание программы и моделирование.....23
4.2 Макетирование.....26
ЗАКЛЮЧЕНИЕ.....28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....29
ПРИЛОЖЕНИЕ А – Тестовый скетч, показывающий управление шаговым двигателем с помощью библиотеки CustomStepper.....31
ПРИЛОЖЕНИЕ Б – Тестовый скетч, показывающий управление семисегментным дисплеем с помощью библиотеки SevSeg.....33
ПРИЛОЖЕНИЕ В – Алгоритм программы.....34
ПРИЛОЖЕНИЕ Г – Листинг программы.....39

ВВЕДЕНИЕ

Лифтом называется транспортное устройство прерывного действия, предназначенное для подъёма и спуска людей (грузов) с одного уровня на другой, кабина (платформа) которого перемещается по жёстким вертикальным направляющим, установленным в шахте, снабжённой на посадочных (загрузочных) площадках запираемыми дверями. Электрическим считается лифт, лебёдка которого приводится в действие электродвигателем.

Ни один современный жилой дом не обходится без лифтового оборудования, а в общественных заведениях этот механизм выглядит настолько естественно, что скорее вызывает удивление отсутствиме лифта, чем его присутствие. Последнее время появилась потребность в пассажирских лифтах для индивидуальных домов, потому что этот механизм не просто облегчает жизнь и делает ее более комфортной, но иногда бывает просто насущной необходимостью - это видно на примере людей с ограниченными способностями самостоятельно передвигаться. Помимо обыкновенных лифтов и эскалаторов на заводах изготавливают специальные подъёмные механизмы для инвалидов, которые устанавливаются в общественных зданиях, жилых домах, на авто-, аэро- и железнодорожных вокзалах, в автобусах. Исполнение этих лифтов и отделочные материалы, которые используются для облицовки пассажирских кабин в состоянии удовлетворить самый изысканный вкус. Большой популярность пользуются панорамные лифты, когда прозрачной является не только кабина лифта, но и шахта внутри которой происходит перемещение кабины.

Основное преимущество вертикального транспорта - небольшая площадь, занимаемая его оборудованием в здании.

Прообразы современных лифтов были известны в Древнем Риме в I в. до н. э. Упоминание о лифтах более позднего периода относится к VI в. (лифт Синайского монастыря в Египте), к первой четверти XIII в. (во Франции) и

XVII в. (лифт Виндзорского замка в Англии, “летающий стул” Велайера в одном из парижских дворцов).

В середине XIX в. в США появились лифты Э. Отиса с ловителями, удерживающими кабину от падения в случае обрыва канатов.

С 60-х годов XIX в. в практику лифтостроения вошли лифты с паровым приводом, затем с гидравлическим и только к началу XX в. широкое и преимущественное развитие получили электрические лифты.

С ростом количества выпускаемых лифтов совершенствуется и их конструкция. Особенность лифтов заключается в том, что они представляют собой изолированную автоматизированную систему, действующую циклически по командам пассажиров. При этом все операции по доставке пассажиров на требуемый этаж и по обеспечению безопасности перевозок выполняются автоматически. Обслуживающий персонал - электромеханики по лифтам, лифтёры-обходчики, диспетчеры и дежурные электромеханики ЛАС.

В данной курсовой работе будет разработан контроллер лифта.

ОСНОВНАЯ ЧАСТЬ

1 Постановка задачи

Перед началом проектирования нужно определиться с тем какие конкретно функции будет выполнять разрабатываемое устройство.

Контроллер лифта должен управлять перемещением лифтовой кабины по этажам. В обычных лифтах это происходит с помощью электродвигателя. Оборудование такого класса слишком дорогое, для разработки прототипа подойдёт маломощный шаговый двигатель, однако, для обеспечения работоспособности контроллера с двигателями требующими более мощного напряжения должен быть использован усилитель.

Чтобы управлять процессом перемещения нужен микроконтроллер, он же будет взаимодействовать со всей периферией, принимать решения о направлении движения и т. д.

Контроллеру лифта нужно как-то понимать на каком этаже находится пользователь и куда он хочет поехать, а значит, должен быть какой-то интерфейс взаимодействия – это будут кнопки вызова. Они будут располагаться как на этажах, рядом с местом, куда лифт должен приезжать, так и внутри самого лифта.

Для того, чтобы сообщать пользователю о перемещениях лифта нужно выводить номер текущего этажа. Это будет делаться с помощью семисегментного дисплея.

Очевидно, что в рамках данного проекта учесть все нюансы работы и эксплуатации лифта невозможно, хотя бы потому, что лифт – составное устройство, включающее в себя не только двигатель, кнопки вызова, кнопки перемещения и дисплей, но и например, кнопку вызова диспетчера при неполадках или двери, открывающиеся и закрывающиеся в начале поездки, им (сторонним устройствам) нужно сообщать о статусе лифта (движется или нет) для этого будет использован отдельный вывод. В рамках проекта вместо сторонних устройств можно использовать светодиод, чтобы было видно подачу

сигнала о том, что лифт едет.

В результате описания функций лифта можно составить структурную схему. Она показана на рисунке 1.

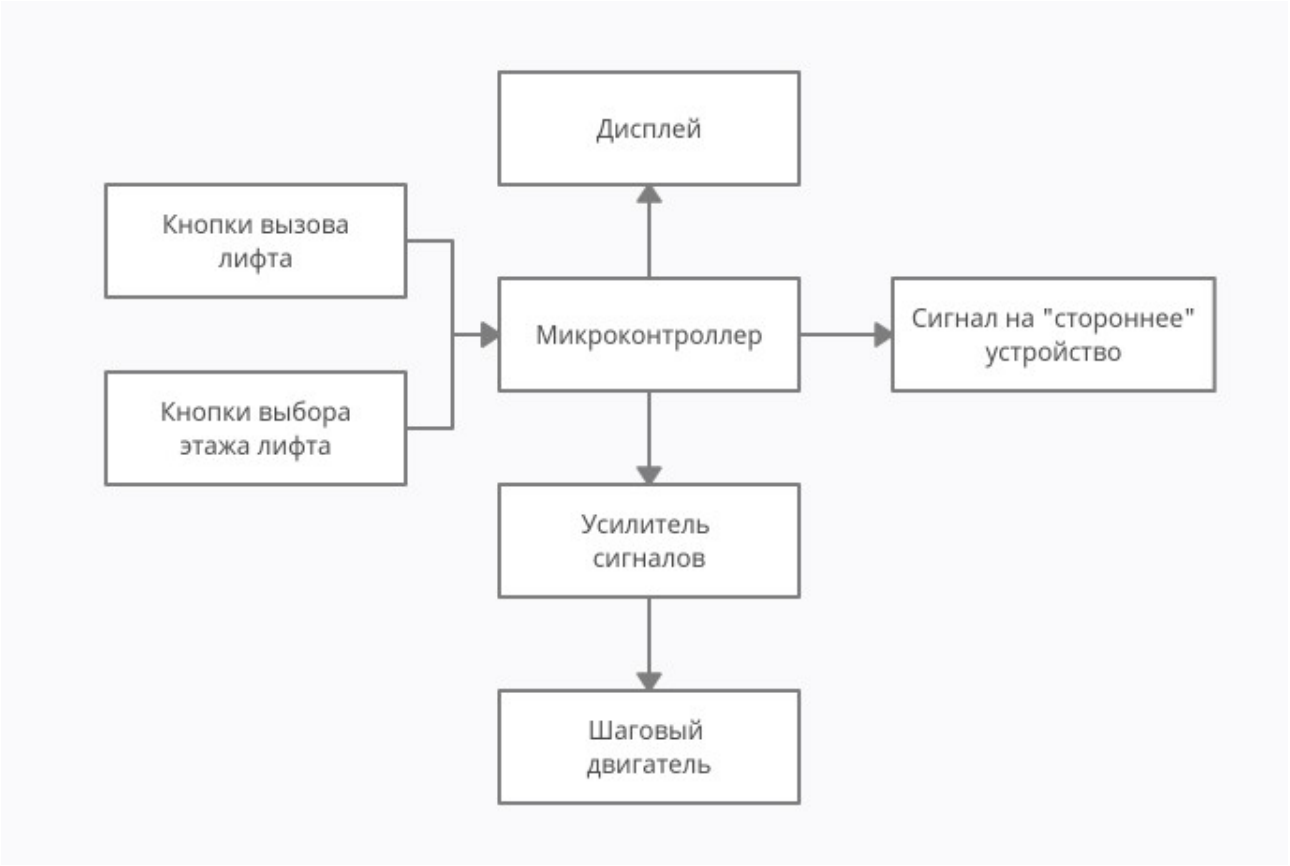


Рисунок 1 – Структурная схема устройства

2 Выбор и обоснование элементной базы

Далее будет обоснован выбор каждого элемента устройства, ссылки на позиции, стоимость и количество закупленных элементов будут приведены в подразделе 4.2.

Макетная плата, изображённая на рисунке 2, была выбрана потому, что позволяет быстро заменять и переподключать элементы проекта. Наиболее доступной и подходящей под цели проекта является макетная плата SYB-120, за счёт того, что имеет две шины питания, большую длину, позволяющую располагать элементы менее кучно, так, чтобы они друг другу не мешали, удобную маркировку, на которую легко ориентироваться при корректировке схемы на этапе сборки проекта с реальными элементами.

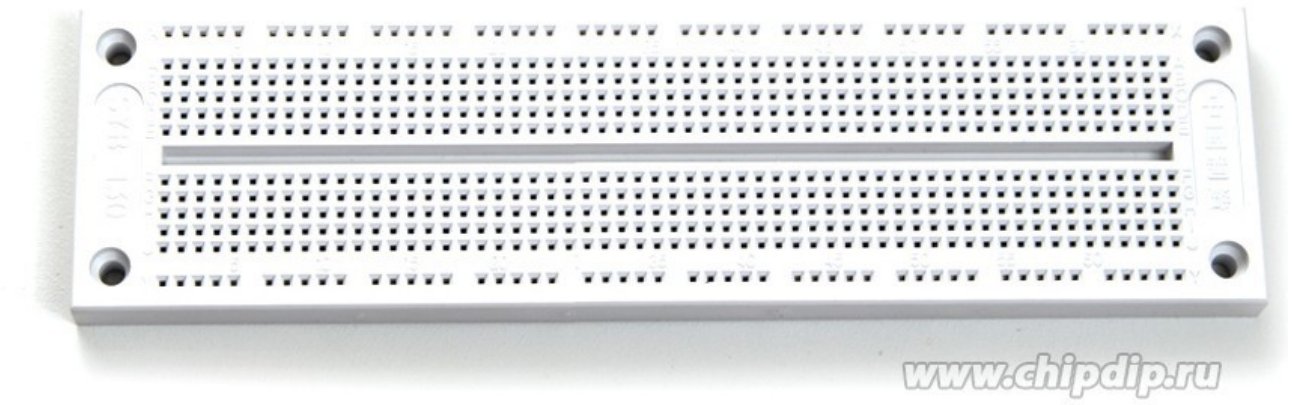


Рисунок 2 – Макетная плата [1]

В качестве кнопок для выбора этажа, на который поднимется лифт или кнопки вызова лифта, была выбрана KLS7-TS1204-7.3-180, потому что имеет возможность замены шляпки кнопки, что будет важно при обслуживании лифтов. Помимо этого она имеет достаточные размеры, чтобы на неё было удобно нажимать, что тоже не маловажно. Кнопка показана на рисунке 3.

Рисунок 3 – Тактовая кнопка [2]

Электрические характеристики кнопки:

- Рабочий ток: 50 мА;
- Рабочее напряжение: 12 В[2].

В качестве дисплея, на котором будет отображаться номер этажа, на котором находится лифт был выбран BL-S56A-11UB за счёт своей низкой цены и простоты обращения. Последнее выражается в том, что на данный вид дисплеев написана библиотека для arduino, позволяющая упростить взаимодействие с дисплеем и ускорить скорость разработки. Он показан на рисунке 4.

Электрические характеристики дисплея:

- Максимальное значения яркости достигается при 30 мА;
- Пиковое значение по току 150 мА;
- Общий катод;
- Рабочее напряжение: 5 В [3].

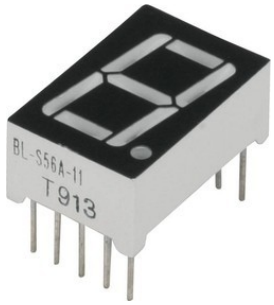


Рисунок 4 – 7 сегментный дисплей [4]

В качестве шагового двигателя был выбран 28BYj-48 за маленькую цену и простоту в обращении. Последнее, опять же, выражается в наличии соответствующей библиотеки для arduino, которая упростит взаимодействие с двигателем и ускорит разработку. Показан на рисунке 5.

Рисунок 5 – Шаговый двигатель [5]

Электрические характеристики шагового двигателя:

- Напряжение питания: 5 В в униполярном и 10 В в биполярном;
- Номинальный ток фазы: 160 мА;
- Номинальное сопротивление обмоток 50 Ом [6].

В качестве усилителя сигнала для подачи на шаговый двигатель был выбран uln2003a за такие характеристики как: исполнение корпуса, который несложно вставить в макетную плату и низкую цену. Может усилить сигнал вплоть до 50В. Изображён на рисунке 6.

Рисунок 6 – Усилитель ulq2003a [7]

Электрические характеристики усилителя:

- Максимальное напряжение насыщения коллектор-эмиттер: 1.6 В;
- Максимальное напряжение коллектор-эмиттер: 50 В;
- Максимальный непрерывный ток коллектора: 500мА;

- Минимальный коэффициент усиления по постоянному току: 1000 [7].

Теперь стоит выбрать плату на базе arduino, на которой и будет выполнен проект. По количеству портов ввода-вывода (коих нам потребуется 18) выбирать приходится между arduino uno и arduino nano. Не смотря на то, что второй стоит дешевле, первый был куплен ранее, именно это и становиться решающим фактором в выборе отладочной платы. Она показана на рисунке 7.

Рисунок 7 – Arduino UNO [8]

Электрические характеристики отладочной платы:

- Входное напряжение питания через DC-рзъём или Vin: 7.5–12 В;
- Входное напряжение питания через USB: 5 В;
- Максимальный выходной ток вывод 3V3: 150 мА;
- Максимальный выходной ток вывода 5V: 1 А [9].

3 Написание программ и моделирование работы выбранных устройств

Теперь, когда мы определились с элементной базой, можно перейти к написанию простейших программ с её использованием. Это необходимый этап для того, чтобы объединить элементы в одном устройстве.

Для написания программ будет использована Arduino IDE, как наиболее простая в использовании для данной платформы. Arduino IDE – программное обеспечение для пользователей, позволяющее писать свои программы (скетчи) для платформы Arduino [10].

Для моделирования работы микроконтроллера и устройств будет использован proteus.

Proteus – Мощнейшая система автоматизированного проектирования, позволяющая виртуально смоделировать работу огромного количества аналоговых и цифровых устройств.

Программный пакет Proteus VSM позволяет собрать схему любого электронного устройства и симулировать его работу, выявляя ошибки, допущенные на стадии проектирования и трассировки. Программа состоит из двух модулей. ISIS – редактор электронных схем с последующей имитацией их работы. ARES – редактор печатных плат, оснащенный автотрассировщиком Electra, встроенным редактором библиотек и автоматической системой размещения компонентов на плате. Кроме этого ARES может создать трехмерную модель печатной платы.

Proteus VSM включает в себя более 6000 электронных компонентов со всеми справочными данными, а также демонстрационные ознакомительные проекты. Программа имеет инструменты USBCONN и COMPIM, которые позволяют подключить виртуальное устройство к портам USB и COM компьютера. При подсоединении к этим портам любого внешнего прибора виртуальная схема будет работать с ним, как если бы она существовала в реальности. Proteus VSM поддерживает следующие компиляторы: CodeVisionAVR и WinAVR (AVR), ICC (AVR, ARM7, Motorola), HiTECH 28

(8051, PIC Microchip) и Keil (8051, ARM). Существует возможность экспорта моделей электронных компонентов из программы Pspice [11].

3.1 Моделирование работы шагового двигателя

Шаговый электродвигатель это синхронный бесщёточный электродвигатель с несколькими обмотками. Ток, подаваемый в одну из обмоток статора, вызывает фиксацию ротора. Последовательная активация обмоток двигателя вызывает дискретные угловые перемещения ротора, они же шаги. Именно поэтому двигатель называется шаговым. Для управления шаговым двигателем используется специальный контроллер, который называют драйвером шагового двигателя [12]. В качестве последнего у нас выступает, как мы уже решили uln2003a.

Четырехфазный шаговый двигатель (28BYJ-48) — это бесколлекторный двигатель, вращение вала осуществляется шагами (дискретное перемещение). На роторе (валу), расположен магнит, а вокруг него расположены катушки, если поочередно подавать ток на эти катушки, создается магнитное поле, которое отталкивает или притягивает магнитный вал, тем самым заставляя двигатель вращаться. Такая конструкция позволяет с большой точностью управлять валом, относительно катушек. Принципиальная схема четырехфазного шагового двигателя 28BYJ-48 приведена на рисунке 8 [13].

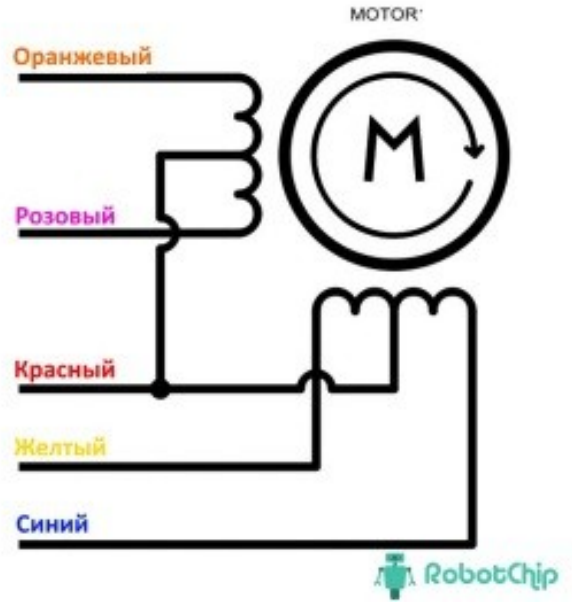


Рисунок 8 – Принципиальная схема двигателя [13]

Цифровой вывод микроконтроллера может выдать ток ~40 мА, а одна обмотка 28BYJ-48 в пике потребляем ~320 мА, следовательно если подключить двигатель напрямую, микроконтроллер сгорит. Для защиты будет использоваться микросхема ULN2003A (по сути, состоящая из 7 ключей), позволяющая управлять нагрузкой до 500 мА (один ключ). Схема подключения показана на рисунке 9.

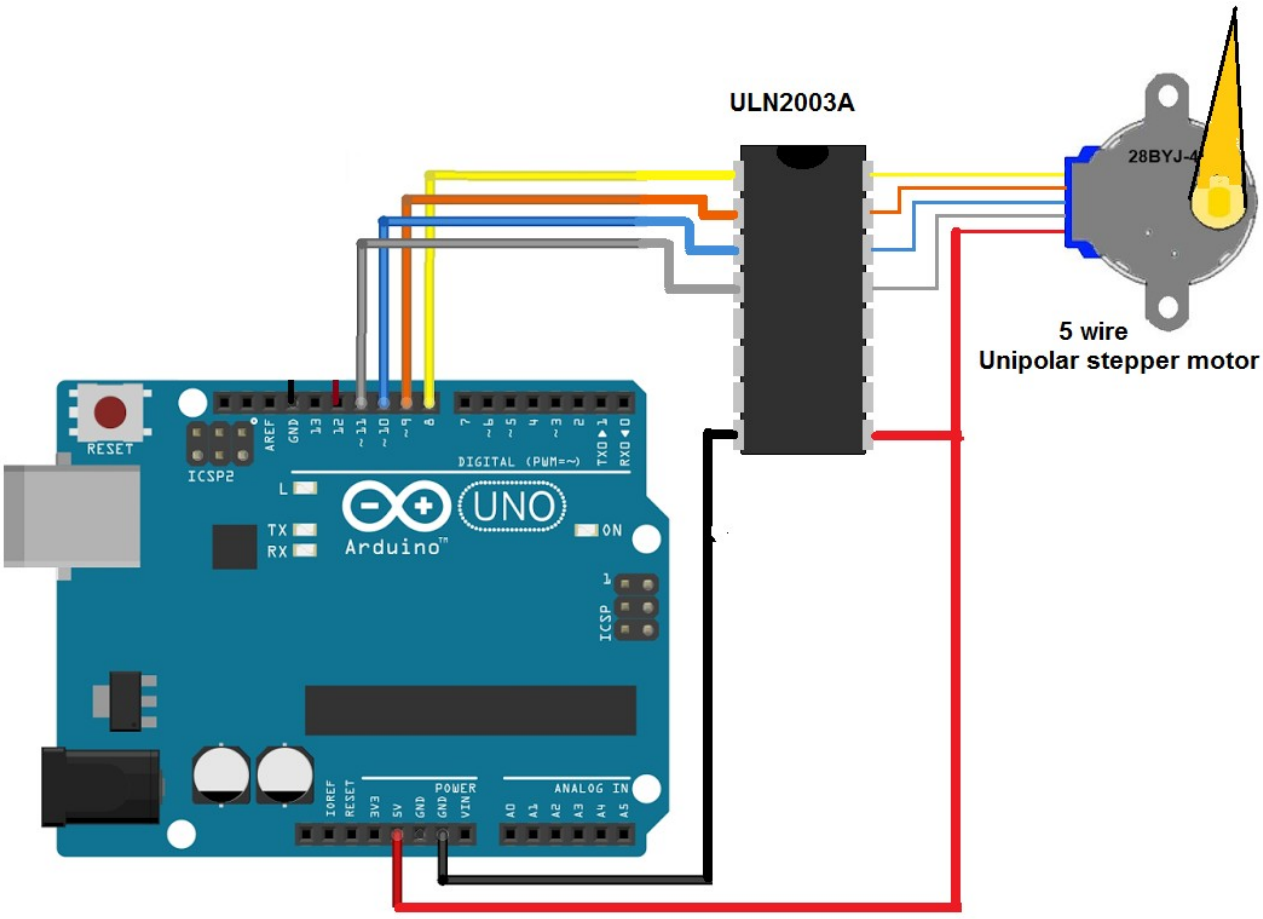


Рисунок 9 – Схема подключения шагового двигателя через uln2003a [14]

Для работы с двигателем будет использоваться библиотека CustomStepper. Она позволяет в пару команд произвести настройку двигателя. В частности будут использоваться следующие команды:

- stepper.setRPM() – Устанавливает количество оборотов в минуту;
- stepper.setSPR() – Устанавливает количество шагов на полный оборот; Максимальное значение 4075.7728395
- stepper.isDone() – Проверяет выполнена ли предыдущая команда;
- stepper.setDirection() – Задаёт направление вращения;
- stepper.rotate() – Устанавливает количество вращений;
- stepper.run() – Выполняет заданные команды.

Итак, разобравшись с нюансами перейдём к моделированию. Схема в

протеусе показана на рисунке 10.

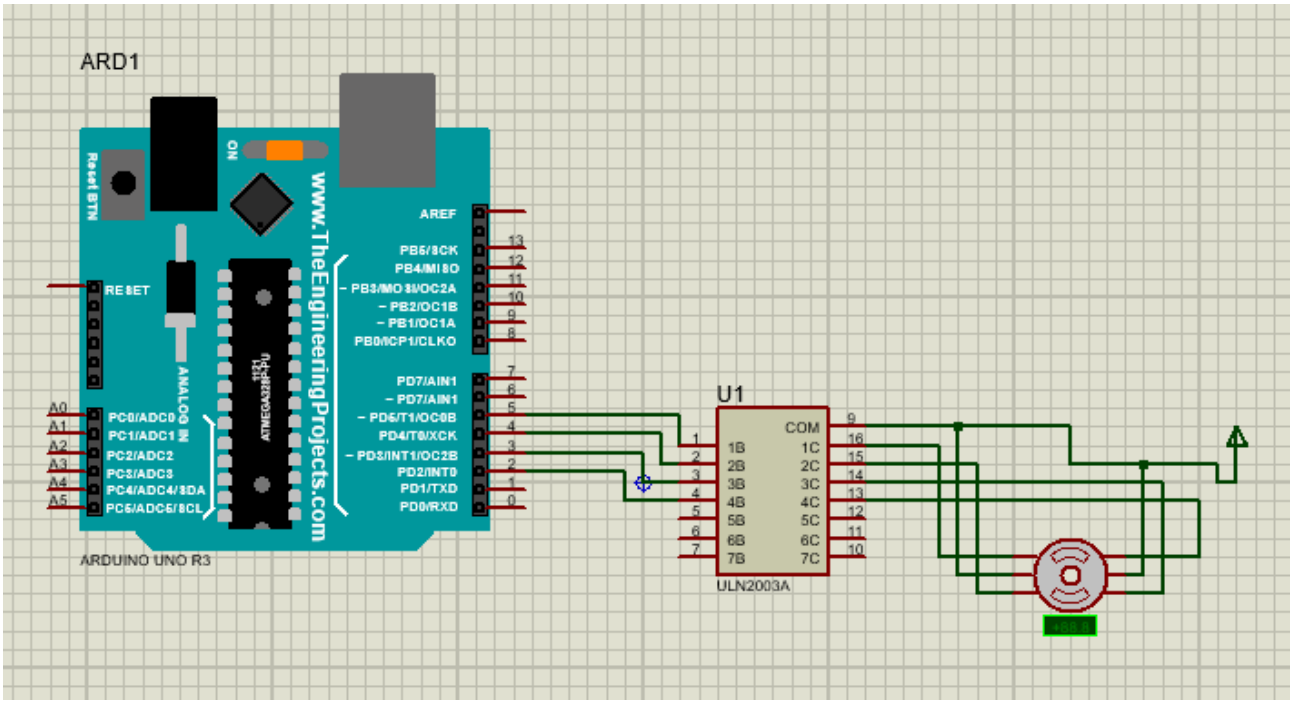


Рисунок 10 – Схема подключения шагового двигателя к контроллеру в proteus

Тестовая программа приведена в приложении А. Там содержатся все необходимые комментарии, так что смысла её комментировать дополнительно нет.

Итог моделирования: программа работает как ожидалось, совершает вращения по часовй, потом против, потом бесконечно по часовой. Доказательство работы показано на рисунке 11. Можно увидеть, что двигатель крутиться

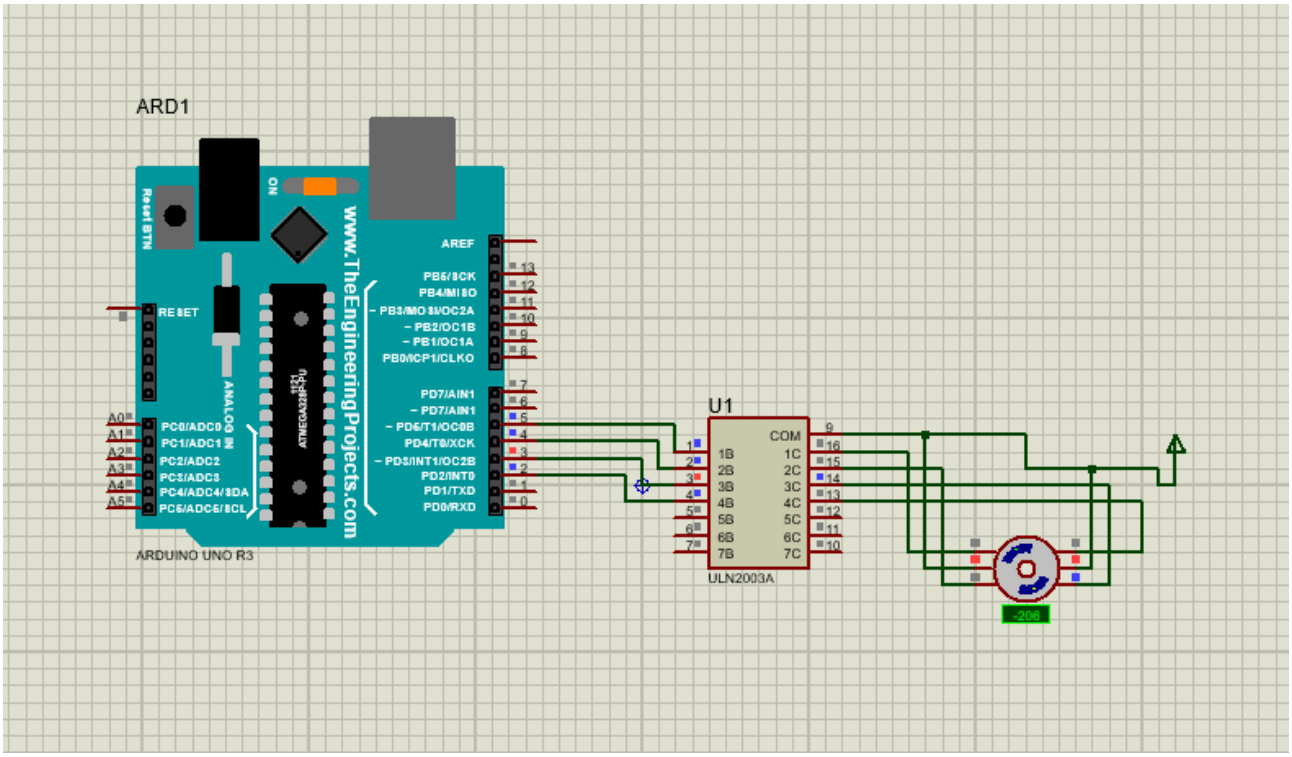


Рисунок 11 – Запущенная тестовая программа

3.2 Моделирование работы семисегментного дисплея

Любой семисегментный дисплей обязательно состоит из семи сегментов. Отсюда и происходит его название. Каждый сегмент – это обычный отдельный светодиод. Мощные семисегментники могут содержать в одном сегменте несколько, как правило, последовательно соединенных светодиодов.

Кроме того в корпусе помимо сегментов находится еще и точка или запятая или другой символ.

С помощью семи сегментов можно изобразить десять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 и некоторые буквы, как латиницы, так и кириллицы.

Светодиоды всех элементов соединяются одноимёнными выводами между собой или анодами, или катодами. Поэтому разделяют семисегментные дисплей с общим анодом и общим катодом.

Вне зависимости от количества разрядов и размеров цифр каждый сегмент имеет название в виде одной из первых букв английского алфавита: a, b, c, d, e, f, g. Точка обозначается dp. Это можно увидеть на рисунке 12

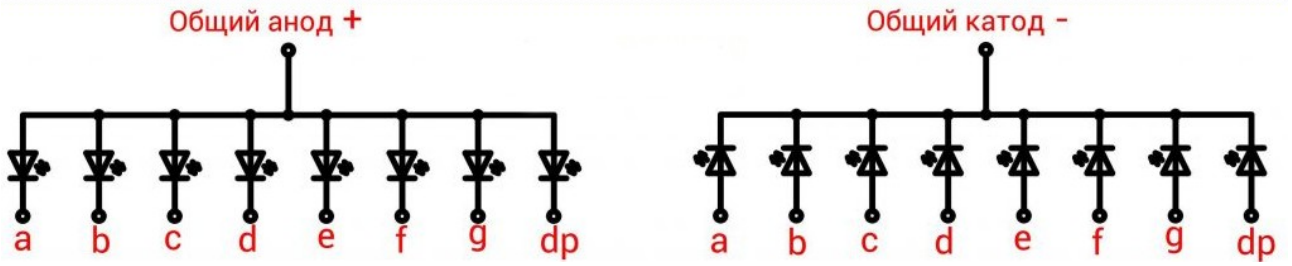


Рисунок 12 – Принципиальные схемы семисегментного дисплея [15]

Чтобы отобразить на дисплее цифру или букву следует засветить несколько сегментов. Например, для отображения единицы 1 задействуются сегменты b и c. При отображении восьмерки 8 задействуются все символы от a до g. Пятерка получается из таких символов: a, c, d, f, g.

Теперь рассмотрим, как подключить семисегментный дисплей к микроконтроллеру ATmega328P. Подключим его к порту D. Данные порт имеет все восемь бит, что очень удобно сочетается с количеством выводов одноразрядного семисегментного дисплея у которого их также восемь с учетом вывода для точки.

Схемы подключения с общим анодом ОА и общим катодом ОК аналогичны, только общий вывод подключается соответственно к плюсу или минусу источника питания [15]. Пример подключения можно увидеть на рисунке 13

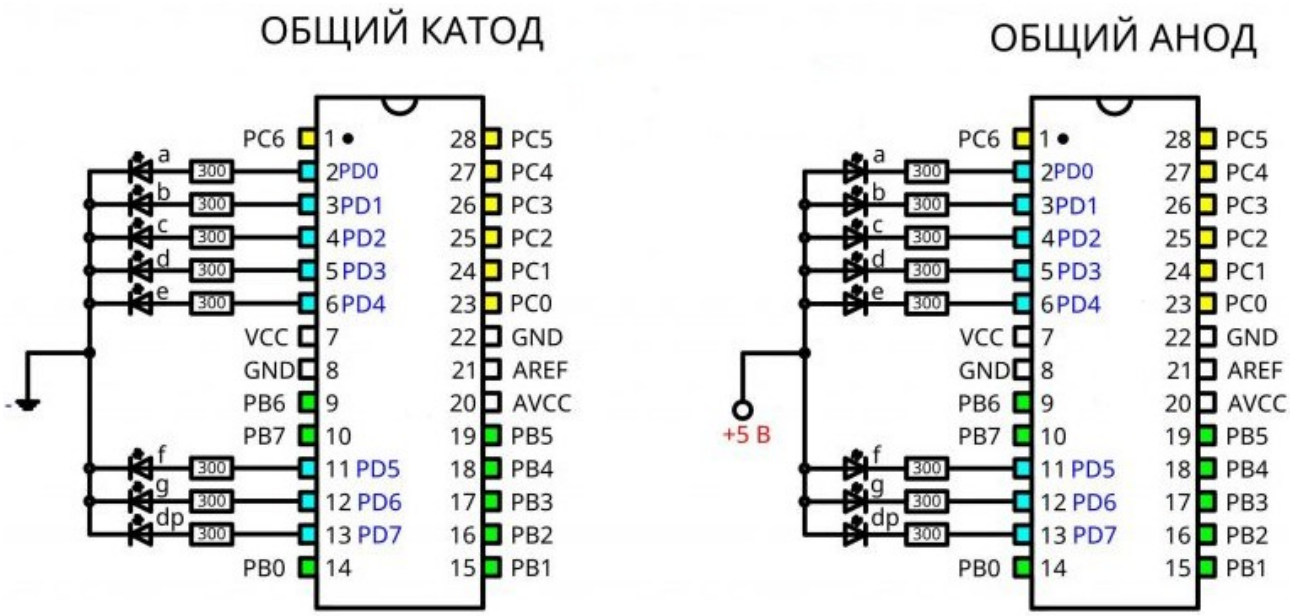


Рисунок 13 – Схема подключения семисегментного дисплея к микроконтроллеру [15]

Для работы с дисплеем будет использована библиотека sevSeg. Использоваться будут следующие команды:

- sevseg.begin() – Инициализирует дисплей с заданными настройками;
- sevseg.setBrightness() – Устанавливает яркость дислея;
- sevseg.setNumber() – Задаёт число, которое должно отобразиться на дисплее;
- sevseg.refreshDisplay() – Непосредственно выводит заданное функцией выше число.

Имея ввиду рисунок 13 смоделируем схему в proteus. Она показана на рисунке 14.

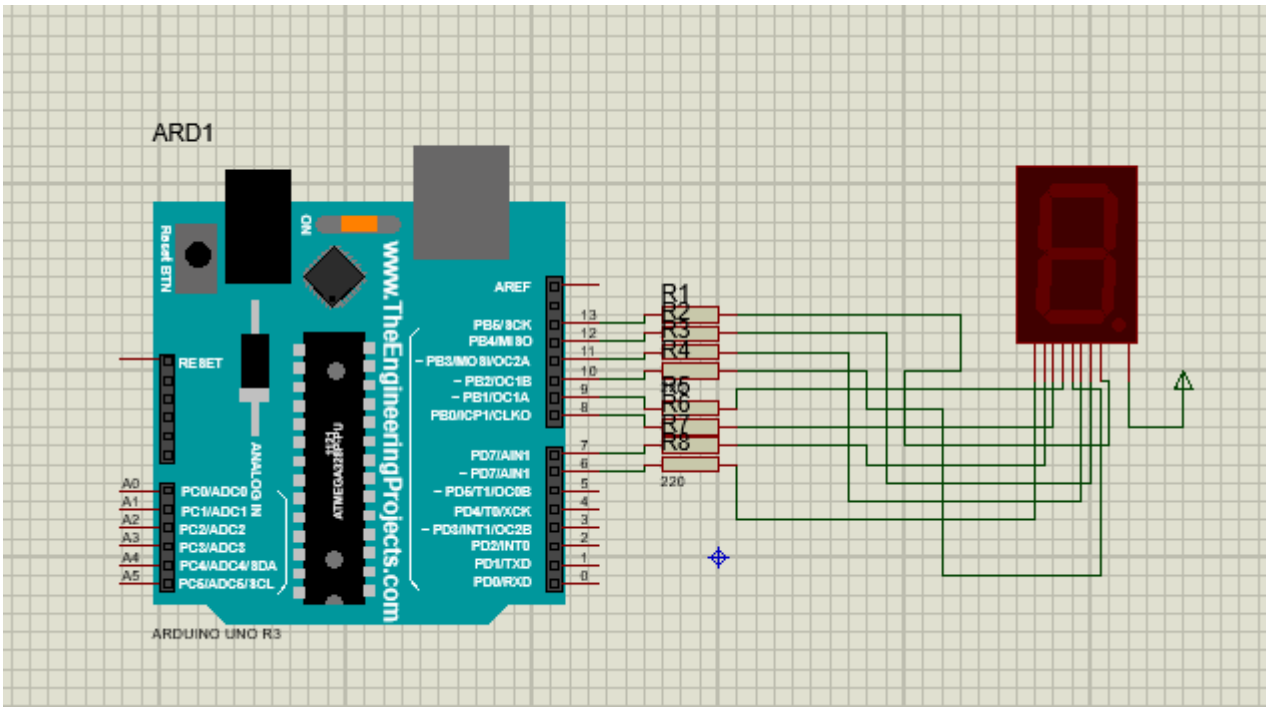


Рисунок 14 – Схема подключения семисегментного дисплея к контроллеру в proteus

Далее загрузим тестовую программу из приложения Б. Т. к. там достаточно комментариев, то рассматривать мы подробно код не будем посмотрим лишь на результат моделирования показанный на рисунке 15.

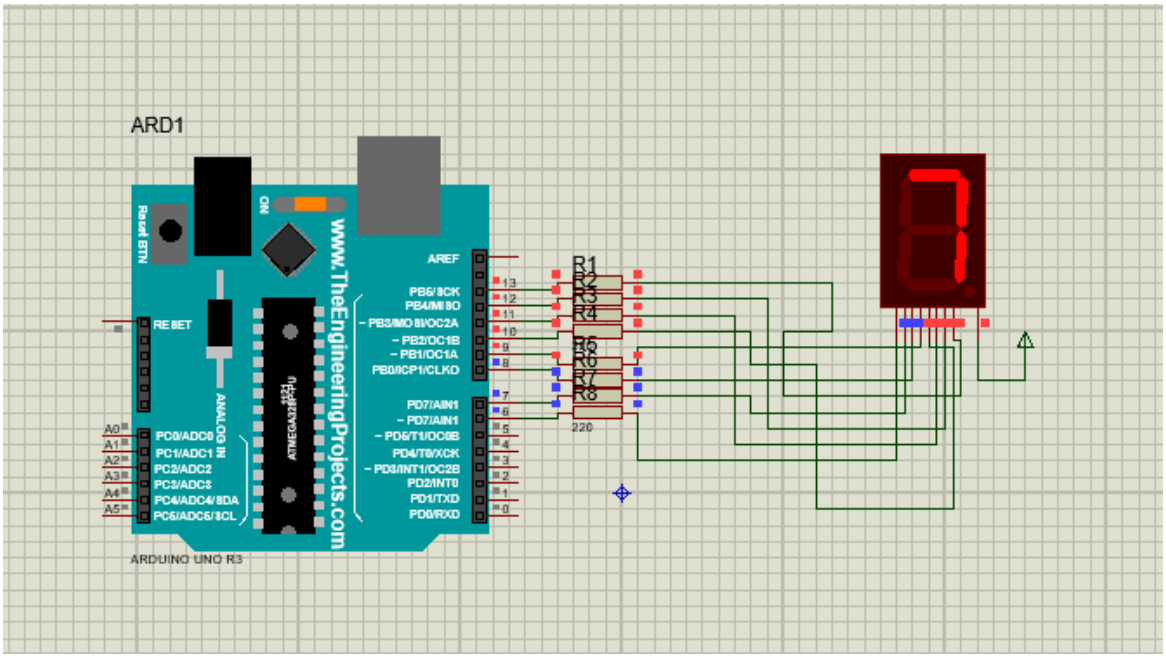


Рисунок 15 – Результат моделирования тестовой программы

4 Разработка устройства

Ознакомившись с нюансами работы устройств можно собирать схему контроллера лифта. Функциональная схема устройства показана на рисунке 16. Резисторы все резисторы номиналом 220 Ом. М1.1 – шаговый двигатель. В схему, помимо рассмотренных в прошлом разделе элементов были добавлены кнопки вызова лифта (SW1.1, SW2.1, SW4.1) и кнопки выбора этажа лифта(SW3.1, SW5.1, SW6.1), так же был добавлен светодиод симулирующий передачу сигнала о начале движения лифта стороннему устройству.

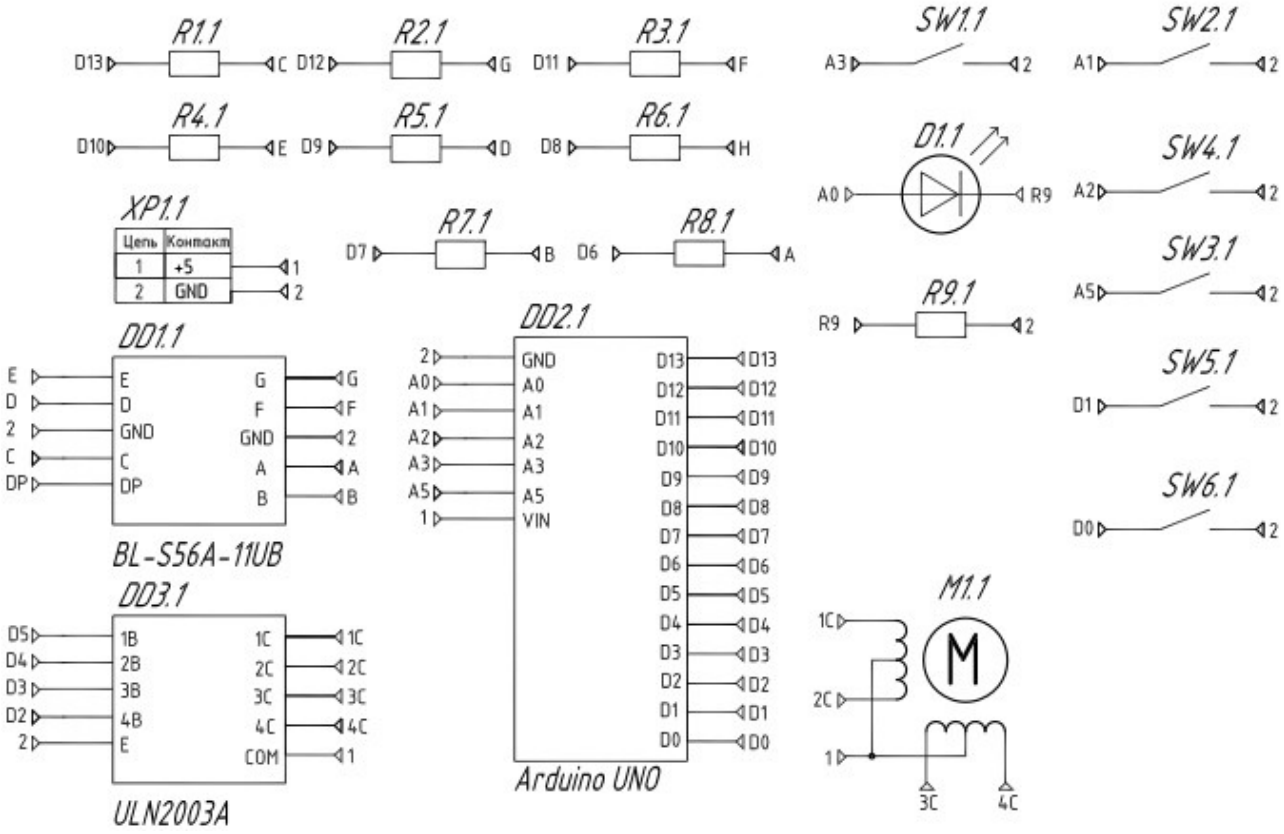


Рисунок 16 – Функциональная схема устройства

На основе функциональной схему устройства была собрана схема в Proteus она показана на рисунке 17. Кнопки с В1 по В3 – отвечают за выбор этажа внутри лифта, а кнопки с В4 по В6 – за вызов лифта на этаж. В1 и В4 – первый этаж, В2 и В5 – второй, В3 и В6 – третий.

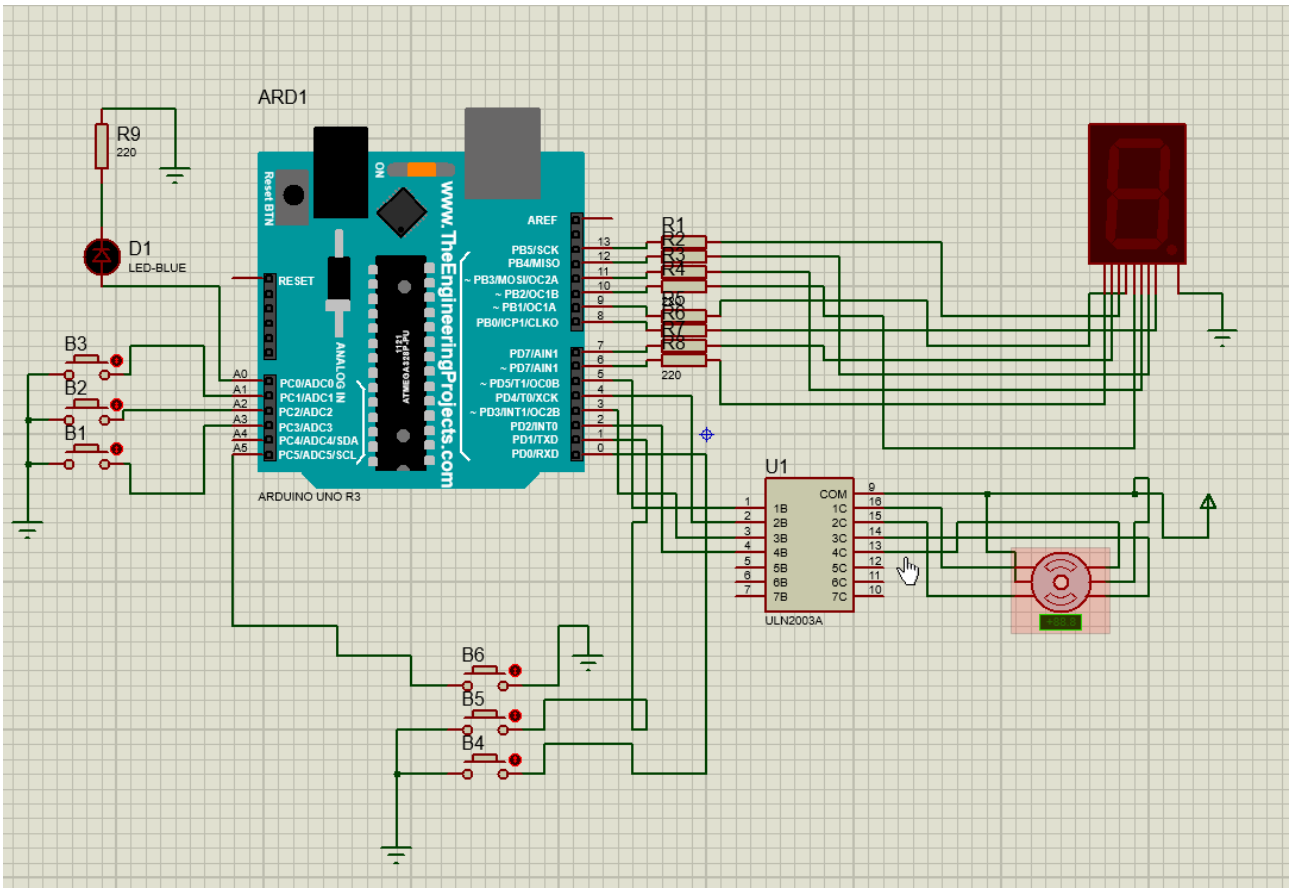


Рисунок 17 – Схема устройства в Proteus

4.1 Написание программы и моделирование

Общий алгоритм работы устройства заключается в следующем: контроллер ждёт пока нажмётся какая-то из кнопок, затем, когда кнопка нажимается, понимает на какой этаж нужно ехать, подаёт сигнал, что он поехал и вращает двигатель столько раз, сколько нужно, чтобы доехать до этажа, а затем устанавливает в 0 порт, сигнализирующий о движении. Полный алгоритм показан в приложении В

Помимо описанных ранее в программе будут использоваться стандартные функции: pinMode() – задаёт режим работы порта, digitalWrite() – подаёт низкое или высокое значение на цифровой выход, digitalRead() – считывает значение с порта.

Имея ввиду всё изложенное ранее пишем код к устройству. Результат

можно посмотреть в приложении Г.

Раз схема в Proteus собрана и программа написана, то можно запускать моделирование. Импортируем hex файл в proteus и включем схему. Запущенная схема показана на рисунке 18.

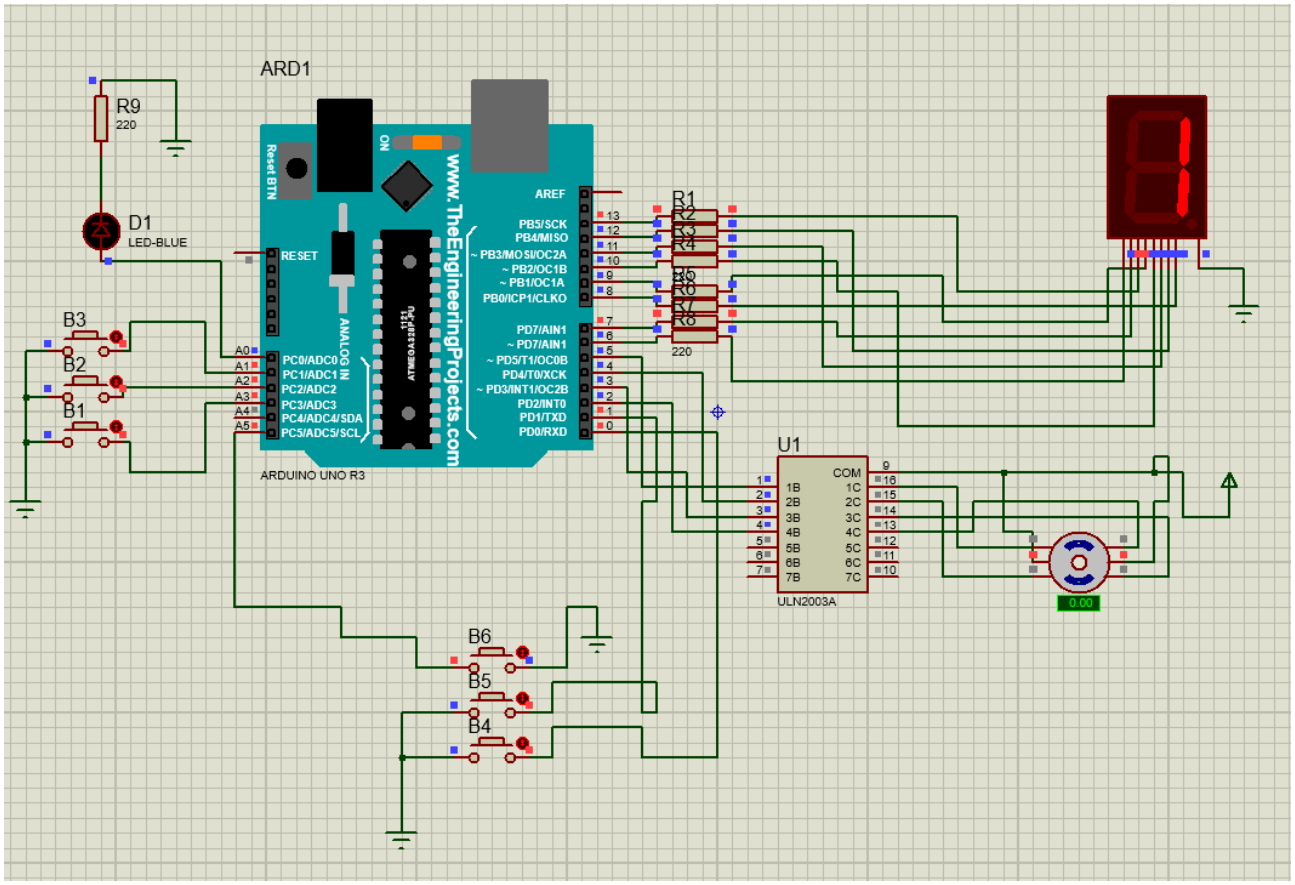


Рисунок 18 – Запущенна схема контроллера лифта в proteus

После нажатия кнопки B3 лифт поедет вверх, на третий этаж. На рисунке 19 можно увидеть как он проезжает 2-ой этаж, а на 20, как он стоит на 3-ем и снова ждёт вызова.

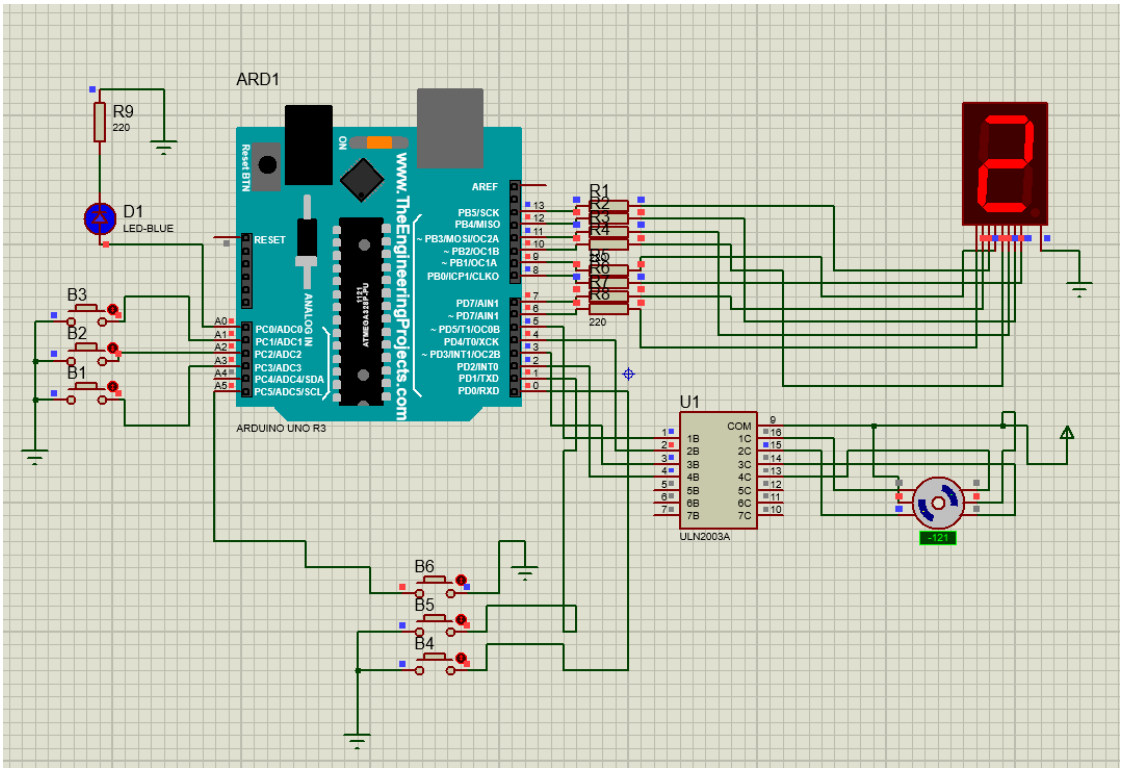


Рисунок 19 – Моделирование ситуации при которой лифт проезжает 2-ой этаж

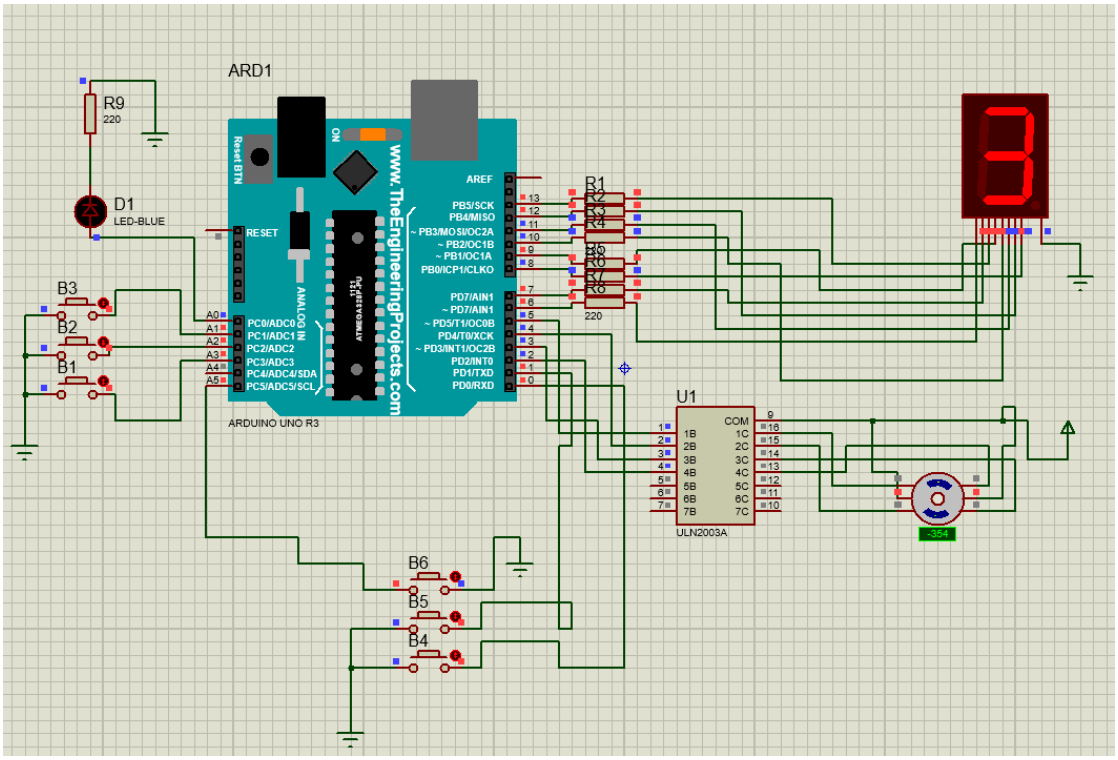


Рисунок 20 – Моделирование ситуации при которой лифт приехал на 3-й этаж и ждёт вызова

4.2 Макетирование

Моделирование показало, что программа работает корректно, значит можно переходить к сборке модели. Для её сборки были использованы детали упомянутые во втором разделе основной части курсовой работы. В таблице 1 приведены стоимость элемента, количество закупленных штук, общая стоимость закупки по позиции и ссылка на позицию в магазине.

В таблице 1 приведены наилучшие позиции подходящие для воспроизведения данной работы. Цены по которым закупались компоненты в рамках работы отличались от табличных, в виду того, что часть элементов у меня уже была.

Для дальнейшего развития проекта некоторые элементы могут быть заменены в зависимости от нужд заказчика. Об этом будет сказано в заключении.

Таблица 1 – Список использованных компонентов

Наименование	Тип	Кол-во, шт	Общая стоимость, Р	Ссылка
Макетная плата	SYS-120	1	450	https://www.chipdip.ru/product/syb-120
Тактовая кнопка	KLS7-TS1204	6	102	https://www.chipdip.ru/product/kls7-ts1204-h1-180-tc-12et
Усилитель	ULN2003A	1	23	https://www.chipdip.ru/product/uln2003a
Семисегментный дисплей	BL-S56A-11UB	1	31	https://www.chipdip.ru/product/bl-s56a-11ub
Отладочная плата	Arduino UNO R3	1	400	https://arduinopro.ru/product/arduino-uno-r3-micro-usb/
Шаговый двигатель	28BYJ-48	1	270	https://www.chipdip.ru/product0/8001779645
Резистор	CF-25	9	27	https://www.chipdip.ru/product0/36485
Светодиод	GNL-3004GD	1	2	https://www.chipdip.ru/product0/8004030774

Итог: 1305 рублей

На рисунке 21 показана сборка проекта в виде макета. Результат работы проекта в макетной сборке можно посмотреть по ссылке: https://imgur.com/a/Ryg4PgM. В видео можно увидеть как после нажатия кнопки B3 шаговый мотор крутиться по часовой стрелке два круга, дисплей показывает сначала цифру 2, а затем и цифру 3, как только движение заканчивается. Потом нажимается кнопка B2 и шаговый двигатель вращается против часовой стрелки один круг, а дисплей меняет число на 2 как только двигатель завершает вращение.

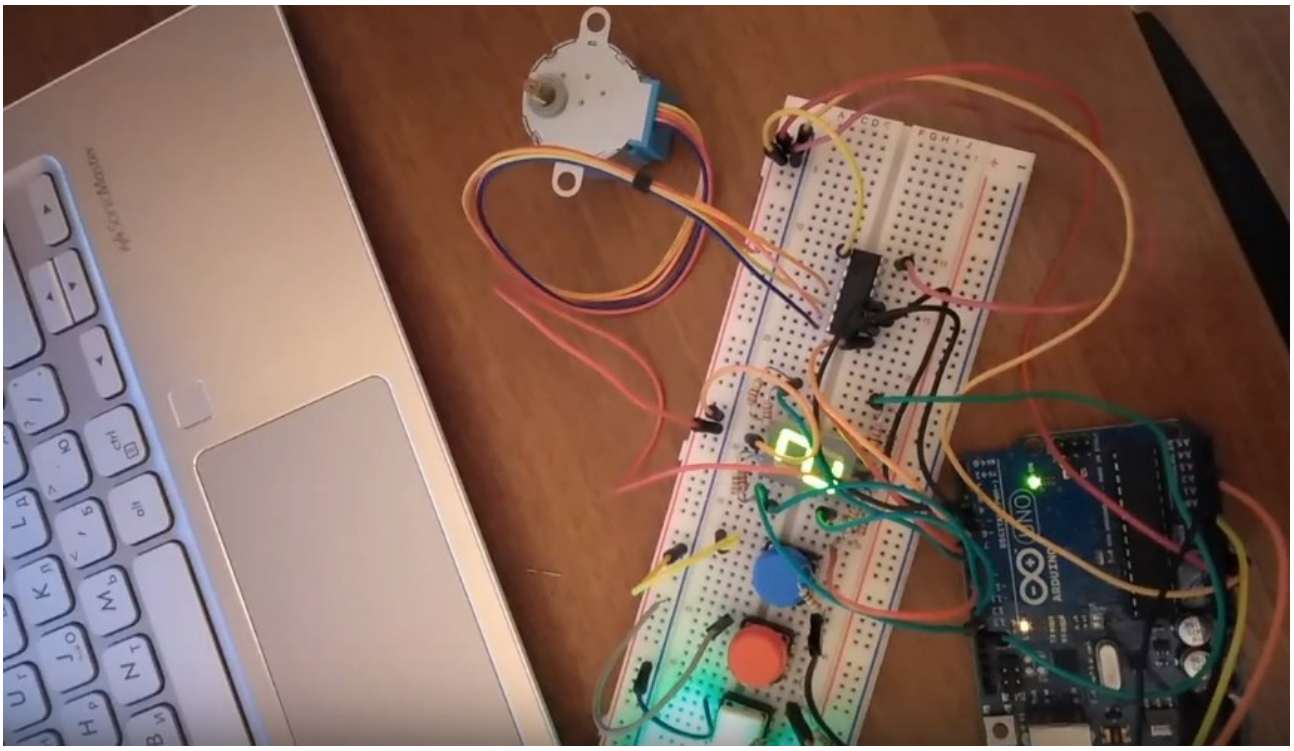


Рисунок 21 – Макет контроллера лифта

ЗАКЛЮЧЕНИЕ

В рамках курсовой работы был спроектирован прототип контроллера лифта, написан код программы контроллера лифта, проведено моделирование в программе Proteus, и наконец, собран макет.

На данном этапе разработки показаны результаты, доказывающие работоспособность устройства. Далее, в зависимости от нужд заказчика, части устройства можно заменить, например перейти к матричным кнопкам, чтобы увеличить количество доступных этажей. Может быть проведена более глубокая интеграция с другими устройствами, которые находятся в лифте. Так же, возможно отказаться от использования отладочной платы в пользу платы с напаянными на неё элементами, в том числе микроконтроллером ATmega328P.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SYB-120, Плата макетная беспаечная. – URL: https://www.chipdip.ru/product/syb-120 (Дата обращения: 18.02.2021)
2. KLS7-TS1204-7.3-180 (TC-12ET) (SWT-9), Кнопка тактовая 12x12. – URL: https://www.chipdip.ru/product/kls7-ts1204-h1-180-tc-12et (Дата обращения: 18.02.2021)
3. LED NUMERIC DISPLAY, 1 DIGIT. BL-S56X-11. – URL: https://static.chipdip.ru/lib/258/DOC000258571.pdf (Дата обращения: 20.02.2021)
4. BL-S56A-11UB, Индикатор синий. – URL: https://www.chipdip.ru/product/bl-s56a-11ub (Дата обращения: 18.02.2021)
5. Шаговый мотор 4-х фазный 28BYJ-48 12В 5.625° 0.32А с редуктором. – URL: https://www.chipdip.ru/product0/9000696962 (Дата обращения: 18.02.2021)
6. Шаговый двигатель 28BYJ-48 5V. – URL: http://wiki.amperka.ru/products:stepper-motor-28byj-48-5v (Дата обращения: 20.02.2021)
7. ULN2003A, Матрица из семи транзисторов Дарлингтона. – URL: https://www.chipdip.ru/product/uln2003a (Дата обращения: 18.02.2021)
8. Arduino Uno R3 [not original], Программируемый контроллер на базе ATmega328. – URL: https://www.chipdip.ru/product/arduino-uno-r3-not-original (Дата обращения: 18.02.2021)
9. Arduino Uno: инструкция, примеры использования и документация. – URL: http://wiki.amperka.ru/products:arduino-uno (Дата обращения: 20.02.2021)
10. Arduino IDE. Установка и запуск. – URL: https://ampermarket.kz/base/arduino_ide/ (Дата обращения: 18.02.2021)
11. Программа Proteus - рисование электронных схем. – URL: https://cxem.net/software/proteus.php (Дата обращения 18.02.2021)

12. Шаговые двигатели. – URL: https://3d-diy.ru/wiki/cnc/stepper-motor/ (Дата обращения 18.02.2021)
13. Обзор шагового двигателя 28BYJ-48 с драйвером ULN2003. – URL: https://robotchip.ru/obzor-28byj-48-s-drayverom-uln2003/ (Дата обращения 18.02.2021)
14. One step at time. – URL: https://www.arduino.cc/en/Tutorial/LibraryExamples/StepperOneStepAtATime (Дата обращения 18.02.2021)
15. Семисегментный индикатор. – URL: https://diodov.net/semisegmentnyj-indikator-programmirovanie-mikrokontrollerov/ (Дата обращения 18.02.2021)

ПРИЛОЖЕНИЕ А. Тестовый скетч, показывающий управление шаговым двигателем с помощью библиотеки CustomStepper

```
#include <CustomStepper.h>            // Подключаем библиотеку управления шаговым двигателем. По умолчанию настроена на двигатель 28BYJ-48-5V
CustomStepper stepper(5, 4, 3, 2);  // Указываем пины, к которым подключен драйвер шагового двигателя

int example = 1;                      // Переменная для демонстрации работы, отвечающая за смену режимов

void setup()
{
  stepper.setRPM(12);                 // Устанавливаем кол-во оборотов в минуту
  stepper.setSPR(48);       // Устанавливаем кол-во шагов на полный оборот. Максимальное значение 4075.7728395
}

void loop()
{
  if (stepper.isDone() and example == 1)  // Когда предыдущая команда выполнена (см. ниже), метод stepper.isDone() возвращает true
  {
    stepper.setDirection(CW);         // Устанавливает направление вращения. Может принимать 3 значения: CW - по часовой, CCW - против часовой, STOP
    stepper.rotate(1);                // Устанавливает вращение на заданное кол-во оборотов
    example = 2;
  }
  if (stepper.isDone() and example == 2)
  {
    stepper.setDirection(CCW);
    stepper.rotateDegrees(90);        // Поворачивает вал на заданное кол-во градусов. Можно указывать десятичную точность (например 90.5), но не принимаются отрицательные значения
    example = 3;
  }
  if (stepper.isDone() and example == 3)
  {
    stepper.setDirection(CW);
    stepper.rotate();                 // Будет вращать пока не получит команду о смене направления или пока не получит директиву STOP
  }
```

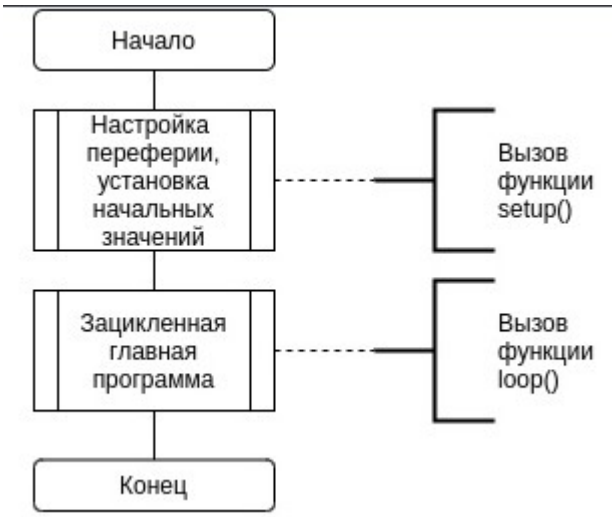
```
  stepper.run();                    // Этот метод обязателен в блоке loop. Он инициирует работу двигателя, когда это необходимо
}
```

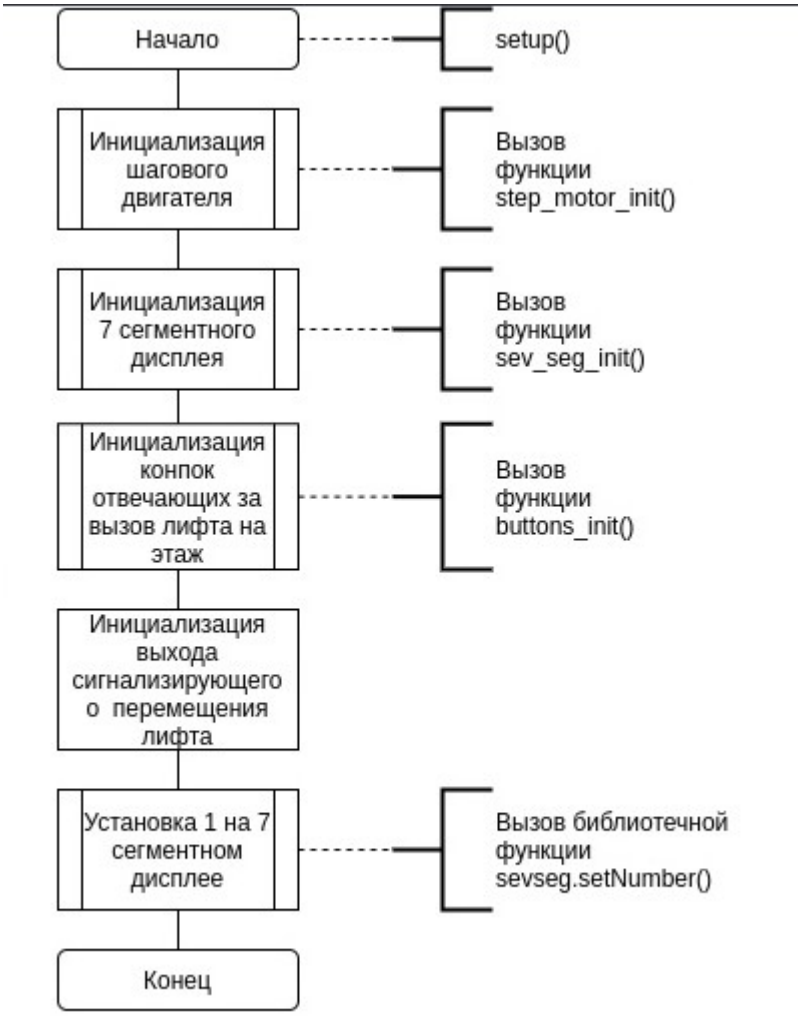
ПРИЛОЖЕНИЕ Б. Тестовый скетч, показывающий управление

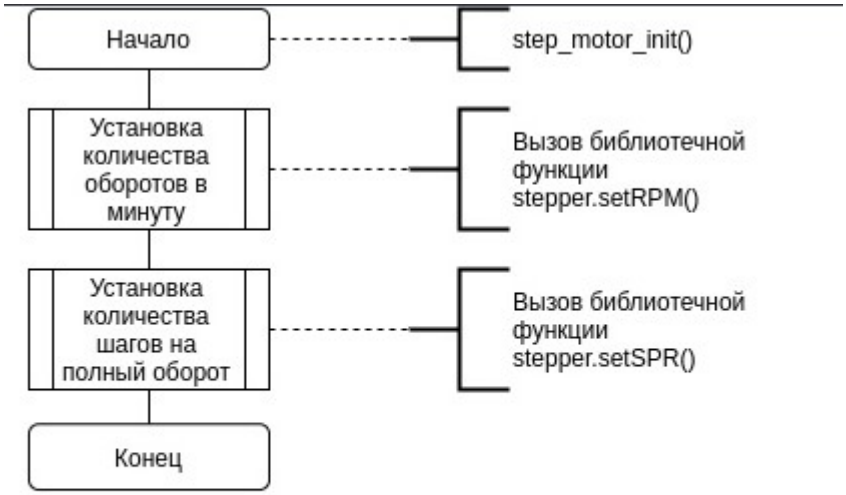
семисегментным дисплеем с помощью библиотеки SevSeg

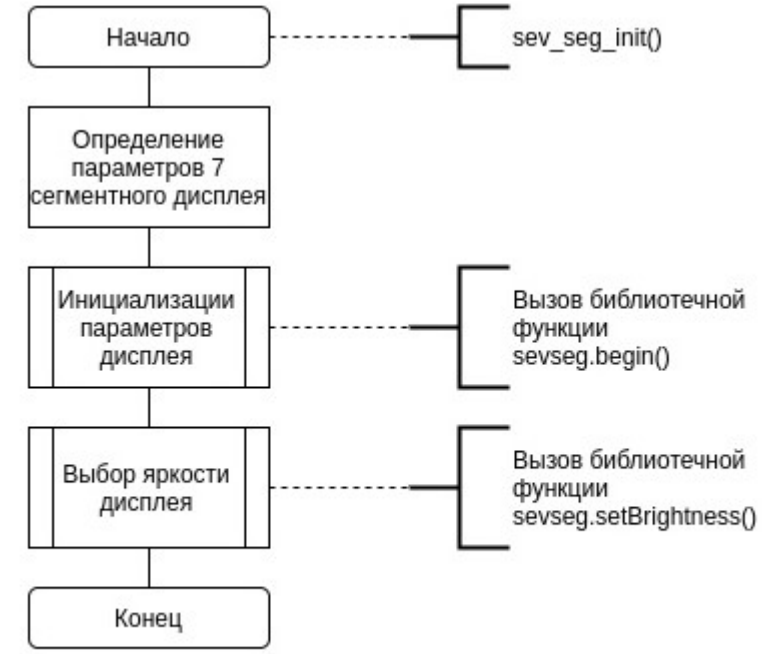
```
#include "SevSeg.h"
SevSeg sevseg;

void setup()
{
  // Сообщаем, что дисплей содержит одно число
  byte numDigits = 1;
  byte digitPins = {};
  // Определяем порты вывода к которым подключён дисплей в следующем
порядке: A, B, C, D, E, F, G, DP
  byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12, 13};
  bool resistorsOnSegments = true;

  // Инициализируем дисплей с записанными ранее настройками
  sevseg.begin(COMMON_ANODE, numDigits, digitPins, segmentPins,
resistorsOnSegments);

  // Устанавливаем яркость дисплея
  sevseg.setBrightness(90);
}

void loop()
{
  {
   // Числа от 0 до 9 будут отображаться с 200мс задержкой
   for(int i = 0; i < 9; i++)
   {
     sevseg.setNumber(i);
     sevseg.refreshDisplay();
     delay(200);
   }
  }
}
```

ПРИЛОЖЕНИЕ В. Алгоритм программы

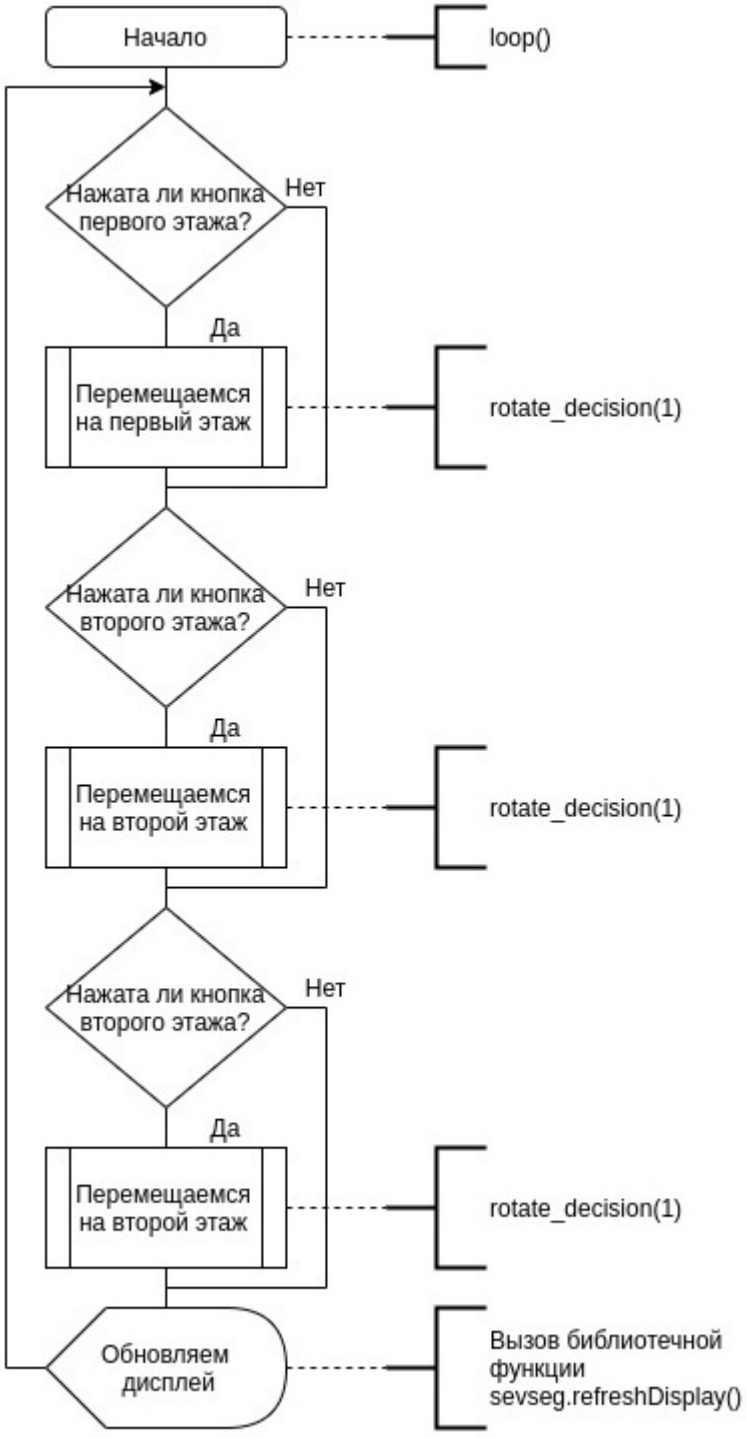


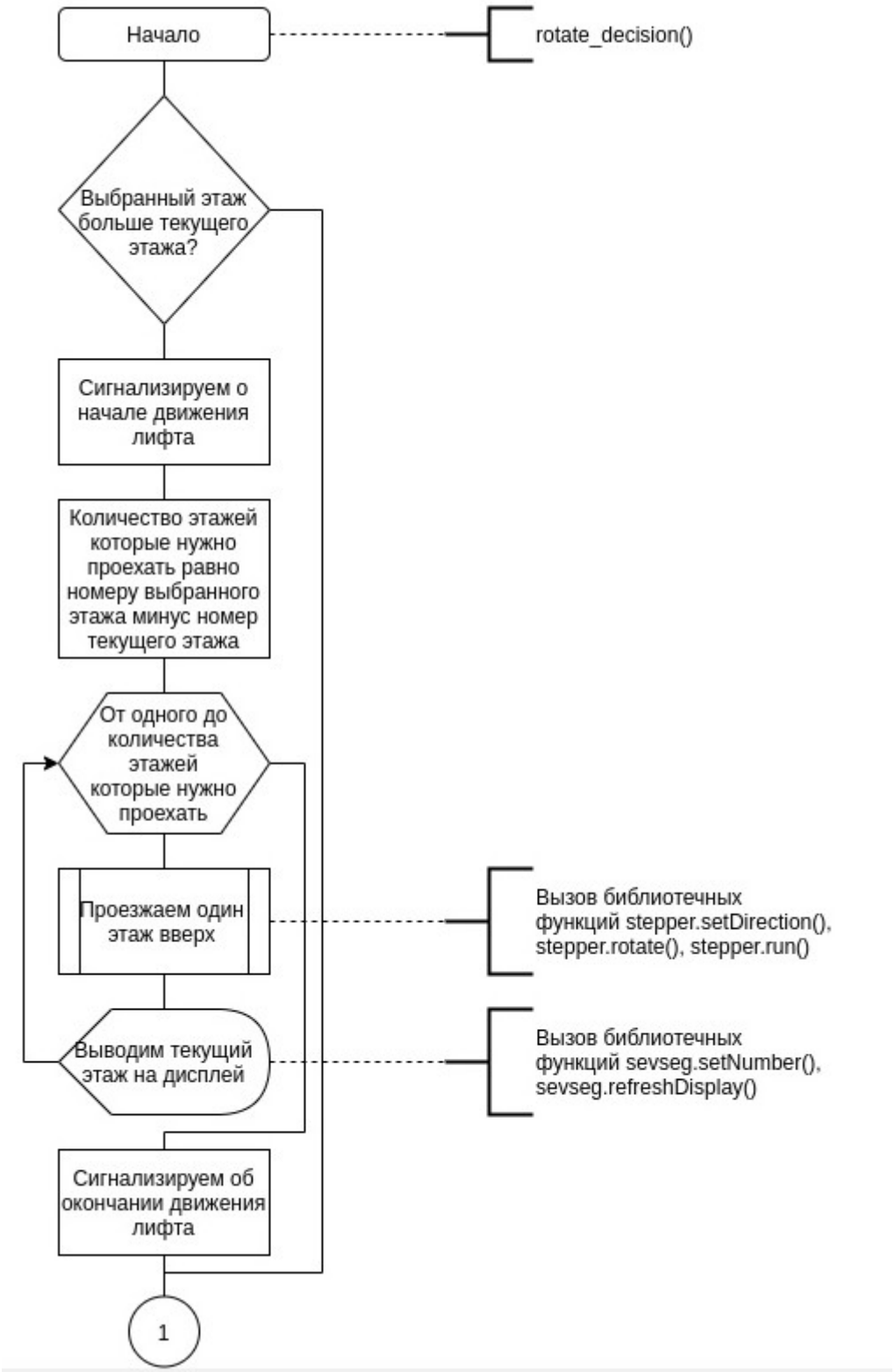


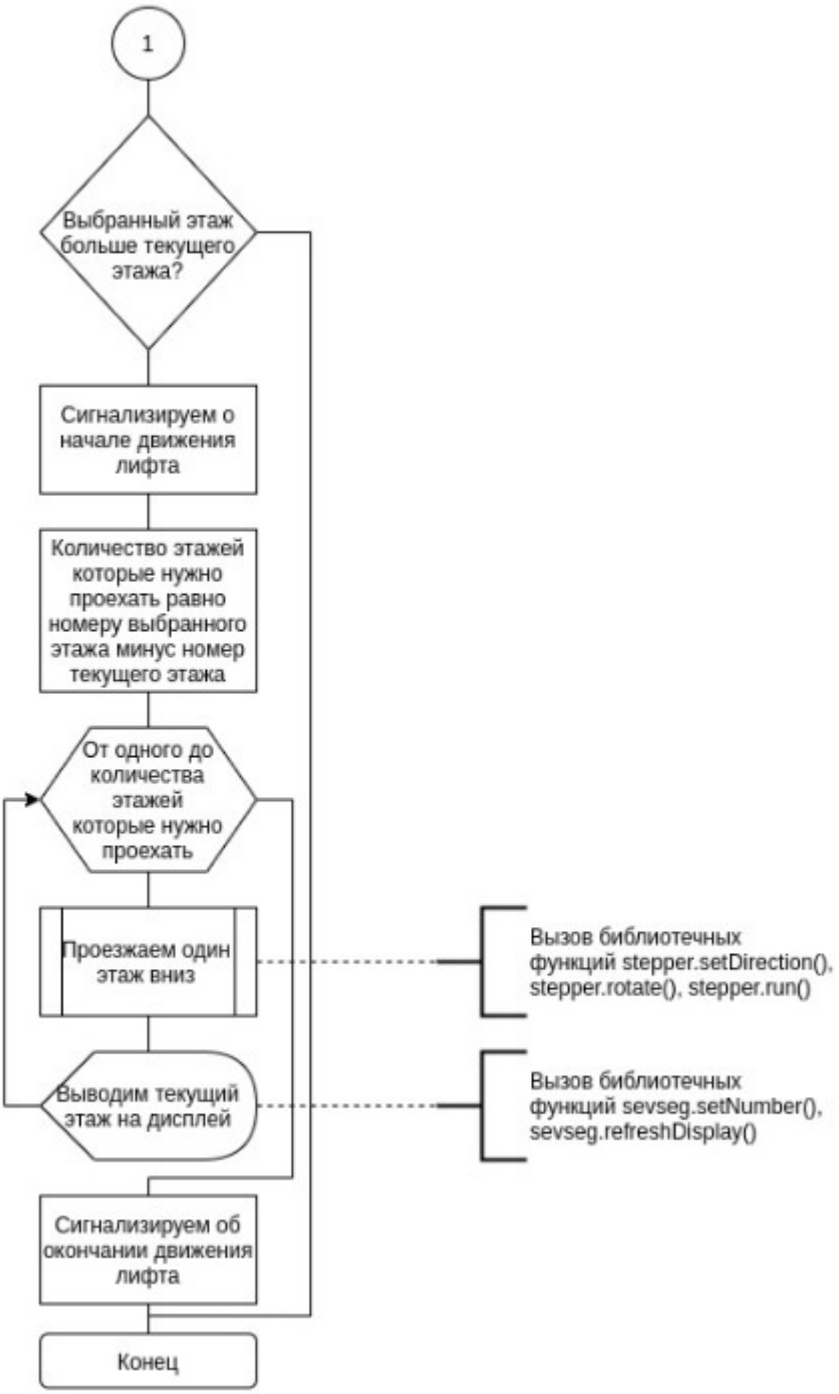

Начало
step_motor_init()
Установка количества оборотов в минуту
Вызов библиотечной функции stepper.setRPM()
Установка количества шагов на полный оборот
Вызов библиотечной функции stepper.setSPR()
Конец


Начало
sev_seg_init()
Определение параметров 7 сегментного дисплея
Инициализации параметров дисплея
Вызов библиотечной функции sevseg.begin()
Выбор яркости дисплея
Вызов библиотечной функции sevseg.setBrightness()
Конец


Начало
Вызов функции button_init()
Определение портов A1, A2, A3, A5, 1 и 0 как работающих на вход, подтянутых к единице
Конец


Начало
loop()
Нажата ли кнопка первого этажа?
Нет
Да
Перемещаемся на первый этаж
rotate_decision(1)
Нажата ли кнопка второго этажа?
Нет
Да
Перемещаемся на второй этаж
rotate_decision(1)
Нажата ли кнопка второго этажа?
Нет
Да
Перемещаемся на второй этаж
rotate_decision(1)
Обновляем дисплей
Вызов библиотечной функции sevseg.refreshDisplay()


Начало
rotate_decision()
Выбранный этаж больше текущего этажа?
Сигнализируем о начале движения лифта
Количество этажей которые нужно проехать равно номеру выбранного этажа минус номер текущего этажа
От одного до количества этажей которые нужно проехать
Проезжаем один этаж вверх
Вызов библиотечных функций stepper.setDirection(), stepper.rotate(), stepper.run()
Выводим текущий этаж на дисплей
Вызов библиотечных функций sevseg.setNumber(), sevseg.refreshDisplay()
Сигнализируем об окончании движения лифта
1


1
Выбранный этаж больше текущего этажа?
Сигнализируем о начале движения лифта
Количество этажей которые нужно проехать равно номеру выбранного этажа минус номер текущего этажа
От одного до количества этажей которые нужно проехать
Проезжаем один этаж вниз
Вызов библиотечных функций stepper.setDirection(), stepper.rotate(), stepper.run()
Выводим текущий этаж на дисплей
Вызов библиотечных функций sevseg.setNumber(), sevseg.refreshDisplay()
Сигнализируем об окончании движения лифта
Конец

ПРИЛОЖЕНИЕ Г. Листинг программы

```
#include "SevSeg.h"
#include <CustomStepper.h>

// Называем выходы более презентабельно, чтобы они легче читались
#define FLOOR_3ST_INNER A1
#define FLOOR_2ST_INNER A2
#define FLOOR_1ST_INNER A3

#define FLOOR_3ST_OUTER A5
#define FLOOR_2ST_OUTER 1
#define FLOOR_1ST_OUTER 0

#define ELEV_DOOR A0

SevSeg sevseg;
CustomStepper stepper(5, 4, 3, 2);

int current_floor = 1;

// Инициализируем семисегментный дисплей
void sev_seg_init(){
  byte numDigits = 1;
  byte digitPins[] = {};
  byte segmentPins[] = {6, 7, 13, 9, 10, 11, 12, 8};
  bool resistorsOnSegments = true;
       sevseg.begin(COMMON_CATHODE, numDigits, digitPins, segmentPins,
resistorsOnSegments);
  sevseg.setBrightness(90);
}

// Инициализируем шаговый мотор
void step_motor_init(){
  stepper.setRPM(12);
  stepper.setSPR(4075.7728395);  //48    4075.7728395
}

// Инициализируем кнопки лифта
void buttons_init(){
  pinMode(FLOOR_3ST_INNER, INPUT_PULLUP);
  pinMode(FLOOR_2ST_INNER, INPUT_PULLUP);
  pinMode(FLOOR_1ST_INNER, INPUT_PULLUP);
```

```
  pinMode(FLOOR_3ST_OUTER, INPUT_PULLUP);
  pinMode(FLOOR_2ST_OUTER, INPUT_PULLUP);
  pinMode(FLOOR_1ST_OUTER, INPUT_PULLUP);
}

void rotate_decision(int selected_floor){
  int floors_to_move;
  int interim_floor;

  // Если заданный этаж выше, то едем вверх
  if (selected_floor > current_floor){
    digitalWrite(ELEV_DOOR, HIGH);
    floors_to_move = selected_floor - current_floor;
    for (interim_floor = 1; interim_floor <= floors_to_move; interim_floor++){
      stepper.setDirection(CW);
      stepper.rotate(1);
      stepper.run();
      while(!stepper.isDone()) stepper.run();
      // Обновляем дисплей
      sevseg.setNumber(current_floor + interim_floor);
      sevseg.refreshDisplay();
    }
    current_floor = current_floor + floors_to_move;
    digitalWrite(ELEV_DOOR, LOW);
  }
  // Если заданный этаж ниже, то едем вниз
  if (selected_floor < current_floor){
    digitalWrite(ELEV_DOOR, HIGH);
    floors_to_move = current_floor - selected_floor;
    for (interim_floor = 1; interim_floor <= floors_to_move; interim_floor++){
      stepper.setDirection(CCW);
      stepper.rotate(1);
      while(!stepper.isDone()) stepper.run();
      // Обновляем дисплей
      sevseg.setNumber(current_floor - interim_floor);
      sevseg.refreshDisplay();
    }
    current_floor = current_floor - floors_to_move;
    digitalWrite(ELEV_DOOR, LOW);
  }
}

void setup() {
```

```
  step_motor_init();
  sev_seg_init();
  buttons_init();
  pinMode(ELEV_DOOR, OUTPUT);
  sevseg.setNumber(1);
}

void loop() {
  // Ждём нажатия кнопок и едем на этаж, когда кнопку нажмут
                if (digitalRead(FLOOR_1ST_OUTER) == LOW or
digitalRead(FLOOR_1ST_INNER) == LOW){
    rotate_decision(1);
  }
                if (digitalRead(FLOOR_2ST_OUTER) == LOW or
digitalRead(FLOOR_2ST_INNER) == LOW){
    rotate_decision(2);
  }
                if (digitalRead(FLOOR_3ST_OUTER) == LOW or
digitalRead(FLOOR_3ST_INNER) == LOW){
    rotate_decision(3);
  }
  sevseg.refreshDisplay();
}
```