МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименования института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки, специальности)

Электроника и робототехника
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему Аппаратно-программный комплекс мини-полигона. Система позиционирования модели автомобиля

Студент А.С. Платонов
(И.О. Фамилия) (личная подпись)

Руководитель А.К. Кудинов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Объем бакалаврской работы 64 стр., 49 рисунков, 4 таблицы, 20 источников, 6 приложений.

В данной бакалаврской работе был разработан действующий мини-полигон способный принимать маршрут от пользователя и отправлять команды на мобильную платформу.

Цель работы: необходимо разработать систему позиционирования мобильного робота на макете мини-полигона с возможностью прокладывания маршрута, посредством человеко-машинного интерфейса.

Задачи работы:

1. разработка структурной схемы;
2. выбор нужных компонентов;
3. разработка электрической принципиальной схемы;
4. разработка алгоритма работы программы для устройства;
5. разработка программной части для устройства;
6. разработка конструкций управляющего модуля и макета мини-полигона;
7. сборка устройства;
8. отладка устройства и экспериментальные исследования

Для оформления чертежей воспользовался программным пакетом КОМПАС-3D V19. Для разработки программы, управляющей логикой работы, использовалась среда разработки Arduino IDE.

Областью применения данной работы являются обучение молодых специалистов при работе с мобильными платформами на производстве и складах.

Abstract

The title of the graduation work is Hardware and software complex of the testing area. Vehicle model positioning system.

The graduation work consists of an explanatory note on 64 pages, introduction, five parts, a conclusion, including 49 figures, 4 tables, the list of 20 references including 5 foreign sources, six appendices and the graphic part on 7 A1 sheets.

The key issue of the thesis is the creation of cheap testing areas with the possibility of training.

The aim of the work is to assemble a testing area.

We start with the statement of the problem and then logically pass over to its possible solutions.

We first design the structural diagram and select necessary components. We then design electric circuit diagram and a program algorithm for the device. Next we design software part for the device, control module and testing area layout. The program code is written in C ++. Finally, we assemble the device and debug it.

In conclusion, we'd like to emphasize that this problem is relevant in designing robotics. The results of the graduation project can be applied in various warehouses and industries, which will greatly facilitate the work of staff.

The work is of interest for wide circle of readers.

Содержание

Введение

Мы живём в современном мире, который развивается каждый день, и люди должны соответствовать ему. Каждое новое поколение более развитое по отношению к предыдущим благодаря накопленным знаниям.

Технологии развиваются, все более интенсивнее внедряются в нашу повседневную жизнь. Многие процессы автоматизируются или упрощаются, появляются новые профессии и новые возможности для людей. То, что когда-то считалось уделом фантастов, сейчас же является обыденностью.

Появление таких платформ, как Arduino или Raspberry, позволило многим заняться робототехникой в виде хобби или создавать на их основе DIY(Do It Yourself)- электронику, например систему умного дома, для собственных нужд. Благодаря простоте и дружелюбности, даже новички без знания схемотехники или программирования могут в скором времени освоить базовые принципы работы с ними. В интернете можно найти множество документаций, статей, учебников, видеоуроков или уже готовых решений, необходимых человеку.

Рисунок 1 – Микроконтроллер Arduino Mega 2560

Именно одна из таких платформ поможет нам в выполнении дипломной работы.

1. Состояние вопроса

1.1. Формулировка актуальности, цели и задач проекта

На современных складах и производстве происходит постепенная автоматизация и роботизация многих процессов. В частности, задачу по перевозке грузов была возложена на мобильные роботизированные платформы.

Комплекс мини-полигона может помочь для оттачивания навыков логистики у только начинающих специалистов, которые ещё не сталкивались на опыте с прокладыванием маршрута для мобильных платформ. Так же данный проект поможет избежать крупных затрат при возникновении аварийных ситуаций во время обучения.

В рамках данной работы необходимо разработать систему позиционирования мобильного робота на макете мини-полигона с возможностью прокладывания маршрута, посредством человеко-машинного интерфейса.

По итогу задачами работы являются:

- разработка структурной схемы;
- выбор нужных компонентов;
- разработка электрической принципиальной схемы;
- разработка алгоритма работы программы для устройства;
- разработка программной части для устройства;
- разработка конструкций управляющего модуля и макета мини-полигона;
- сборка устройства;
- отладка устройства и экспериментальные исследования.

1.2. Обзор существующих решений

Одной из самых первых разработанных систем является глобальная позиционирующая система (GPS), основанная на принципе определения местоположения путём измерения задержки сигнала между спутником и приёмником. Данная технология разрабатывалась исключительно в интересах Министерства обороны Соединённых Штатов Америки. Главной задачей которой представлялось точное обнаружение области по применению военного оружия и отслеживания военной техники.

Не менее популярной является глобальная навигационная спутниковая система, также известная как ГЛОНАССС, применяемая в большинстве своём над территорией Российской Федерации. ГЛОНАСС является аналогом GPS и во многом схожа с ней, но главными отличиями её являются:

- Разный угол наклона спутников, из-за чего эффективность работы в северных широтах намного выше;
- Все российские устройства поддерживают именно систему ГЛОНАСС, что делает её работу независимой от системы GPS;
- Асинхронное движение спутников по отношению к Земле позволяет облегчить работу по управлению системой.

Принцип работы навигационных систем, показанный на рисунке 1.1, основан на отправке спутником закодированного радиосигнала. После получения этого сигнала приёмником, он расшифровывает сообщение, в котором хранится информация о точном местоположении спутника в пространстве. Далее приёмник высчитывает своё метаположение относительно местоположения трёх спутников, опираясь на формулу 1.

$$S = (t - t_i) * c, \quad (1)$$

где S – расстояние между спутником и приёмником, км;

t – время получения сигнала приёмником;

t_i – время отправки сигнала i-го спутника;

с – скорость звука, км/с.

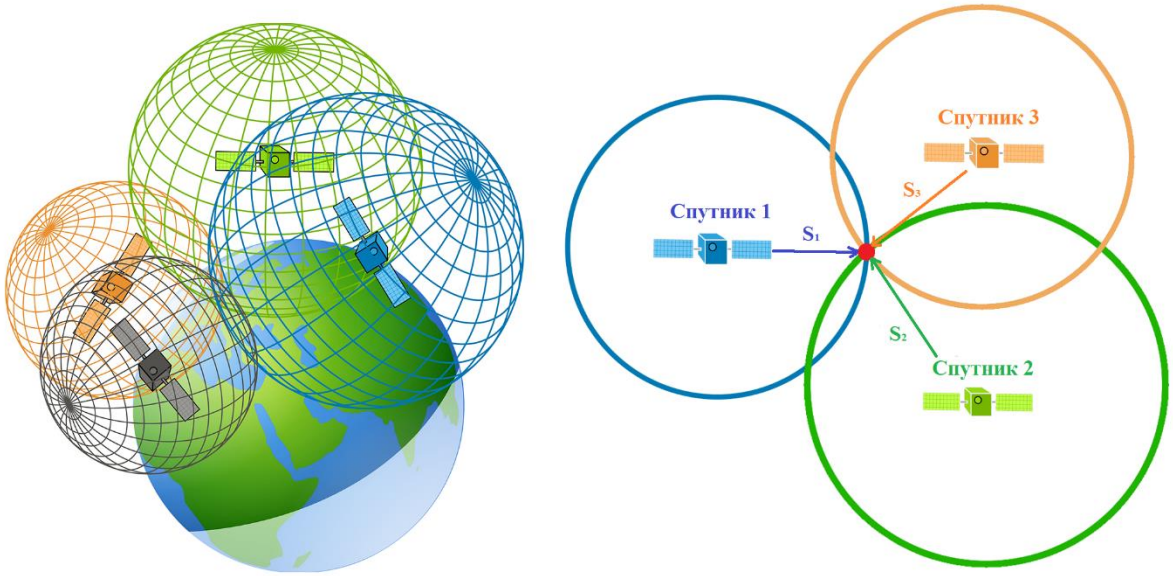

Рисунок 1.1 – Визуализация принципа работы GPS или ГЛОНАСС

Зная расстояние от трёх известных спутников, можно легко ориентировать конкретную точку местоположения приёмника. То есть для корректной работы потребовался некий приёмник с неизвестными координатами и источник сигнала с известным местоположением.

Со временем технологии развивались и системы позиционирования выходили всё на больший уровень. Так со временем подобные системы стали применятся и в гражданском обиходе для отслеживания посылок, контроля перемещения транспортного средства, наблюдения за животными, поиску потерянных устройств и многого другого. Доступность каждому компенсировалось точностью позиционирования вплоть до нескольких километров, дабы не засорять военный частотный диапазон. Из-за ранее упомянутых отличий нецелесообразно использовать какую-то одну систему позиционирования в проекте, так как в разных широтах будут неизбежны ещё большие отклонения в показаниях. Использование же нескольких систем является экономически невыгодным.

Одним из решений по реализации собственной системы может быть система позиционирования, оснащённая компьютерным зрением. Для начала стоит рассмотреть, как работает сама технология компьютерного и только потом система, основанная на этой технологии.

В основе работы компьютерного зрения лежит распознавание изображений с графическим символов. Осуществляется это методом разбития изображения на пиксели и присваивания каждому пикселю некого значения. То есть, если взять чёрно-белое изображение и разбить его на пиксели, то пикселю с белым цветом присваивается значение, зависящее от параметра контрастности. Так белому пикселю присваивается значение 0, в то время как чёрному 255. От сюда можно понять какой именно символ изображён. На рисунке 1.2 можно увидеть наглядно принцип работы компьютерного зрения.

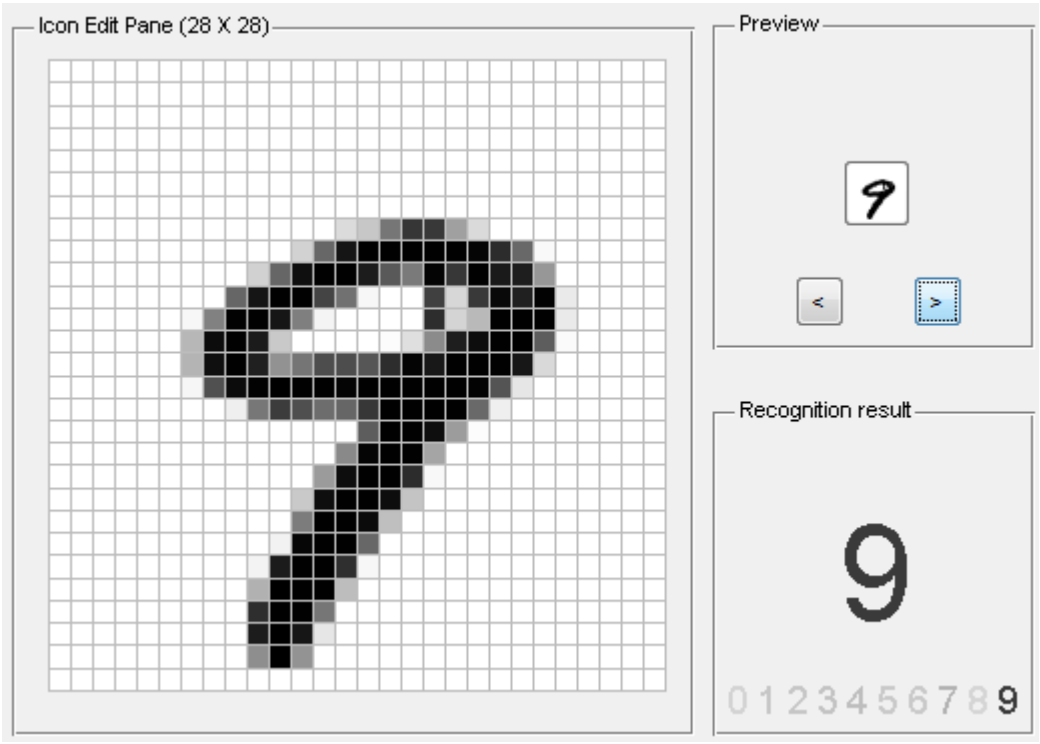

Рисунок 1.2 – Визуализация принципа работы технологии компьютерного зрения

Сама система позиционирования, оснащённая компьютерным зрением, включается в себя набор расположенных в пространстве различных меток в виде графических символов и оптическую камеру для формирования изображения метки. Каждый графический символ в итоге кодируется. Следующим этапом идёт передача сформированного закодированного изображения в микроконтроллер, где его сравнивают. Сравнение осуществляется между полученным изображением и шаблонами, хранящимися во внешней флэш памяти. Сравнения полученного изображения накладывается на шаблоны до тех пор, пока не определится какой именно это символ. После расшифровки символа отслеживаемый объект удаётся найти, потому что понятно на какой именно метке он находится. Принцип работы этой системы продемонстрирован на рисунке 1.3.

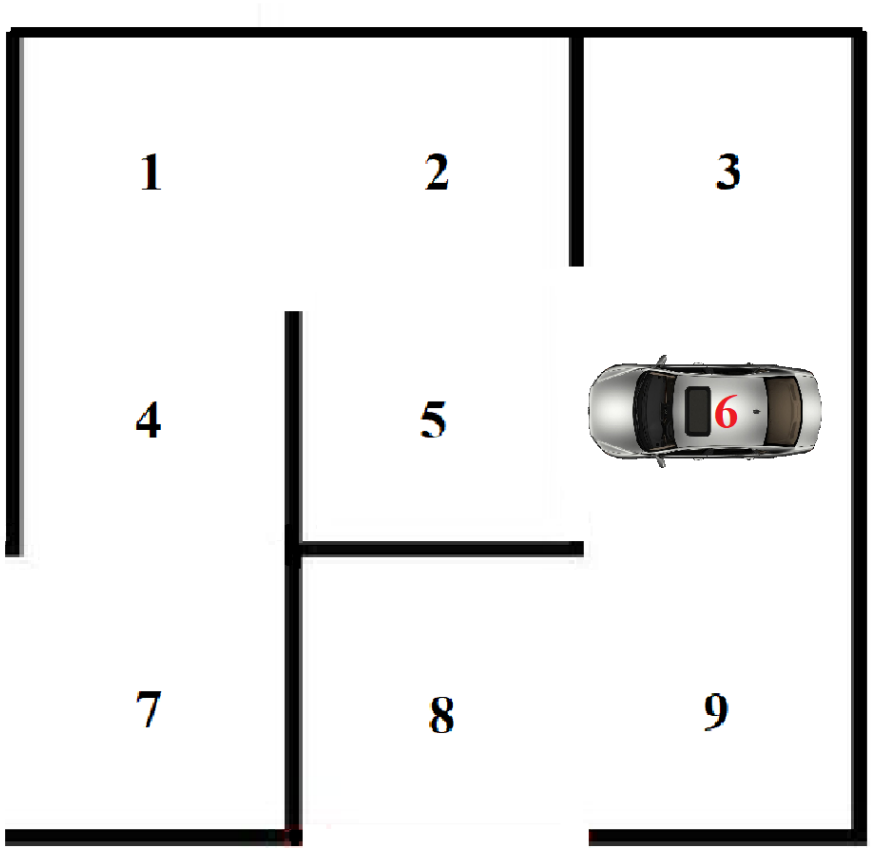

Рисунок 1.3 – Визуализация принципа работы системы позиционирования с компьютерным зрением

Использование данной системы для поставленной цели является неподходящим, так как присутствуют риски возникновении ошибки распознавания символа. Стоит подметить, что расположение меток зависит напрямую от функциональности системы. То есть имеется два выхода из этой ситуации: использование сразу нескольких камер для нахождения меток в разных частях макета, либо расположение одной камеры для считывания метки со строгим расположением на поверхности земли. В первом случае система выйдет довольно затратной и сложно выполнимой. Во втором случае макет мини-полигона будет нуждаться в постоянном обслуживании из-за недолговечности меток, в связи с постоянным загрязнением и стиранием этих меток.

Вдохновившись принципом работы данной системы, выбор пал на создание собственной системы позиционирования по отслеживанию мобильной роботизированной платформы на территории мини-полигона, посредством взаимодействия метки и устройств считывания.

Поэтому грамотным решением было принято в качестве метки взять некий источник воздействия, в то время как под устройством считывания подразумеваются датчики, встроенные в структуру макета мини-полигона.

Выводы по разделу

В данном разделе была поставлена цель, сформулированы задачи для выполнения дипломной работы. Рассмотрены и описаны различные системы навигации, на основании чего был сделан выбор принцип работы разрабатываемой системы отслеживания мобильной платформы.

2. Аппаратная часть

2.1. Разработка структурной схемы

Перед выбором комплектующих необходимо разработать структурную схему, которая будет включать все необходимые компоненты.

Одной из неотъемлемых задач по достижению главной цели реализации аппаратно-программного комплекса мини-полигона, несомненно, является разработка конструкции. Разработка осуществлялась в российской система трёхмерного автоматизированного проектирования КОМПАС-3D. Данное программное обеспечение разрабатывалась изначально для создания чертежей по российским стандартам, из-за чего применяется на большинстве отечественных производств и на территории высших учебных заведений. Тольяттинский государственный университет не является исключением и проводит обучение студентов при помощи этого программного обеспечения, в связи с чем КОМПАС-3D является уже знакомой платформой по написанию чертежей для разработки конструкции.

В разработанной схеме представлены следующие элементы:

- микроконтроллер;
- датчики;
- модуль расширения;
- радиопередатчик;
- реализация человеко-машинного интерфейса;
- электронные индикаторы;
- блок питания.

На рисунке 2.1 представлена структурная схема мини-полигона.

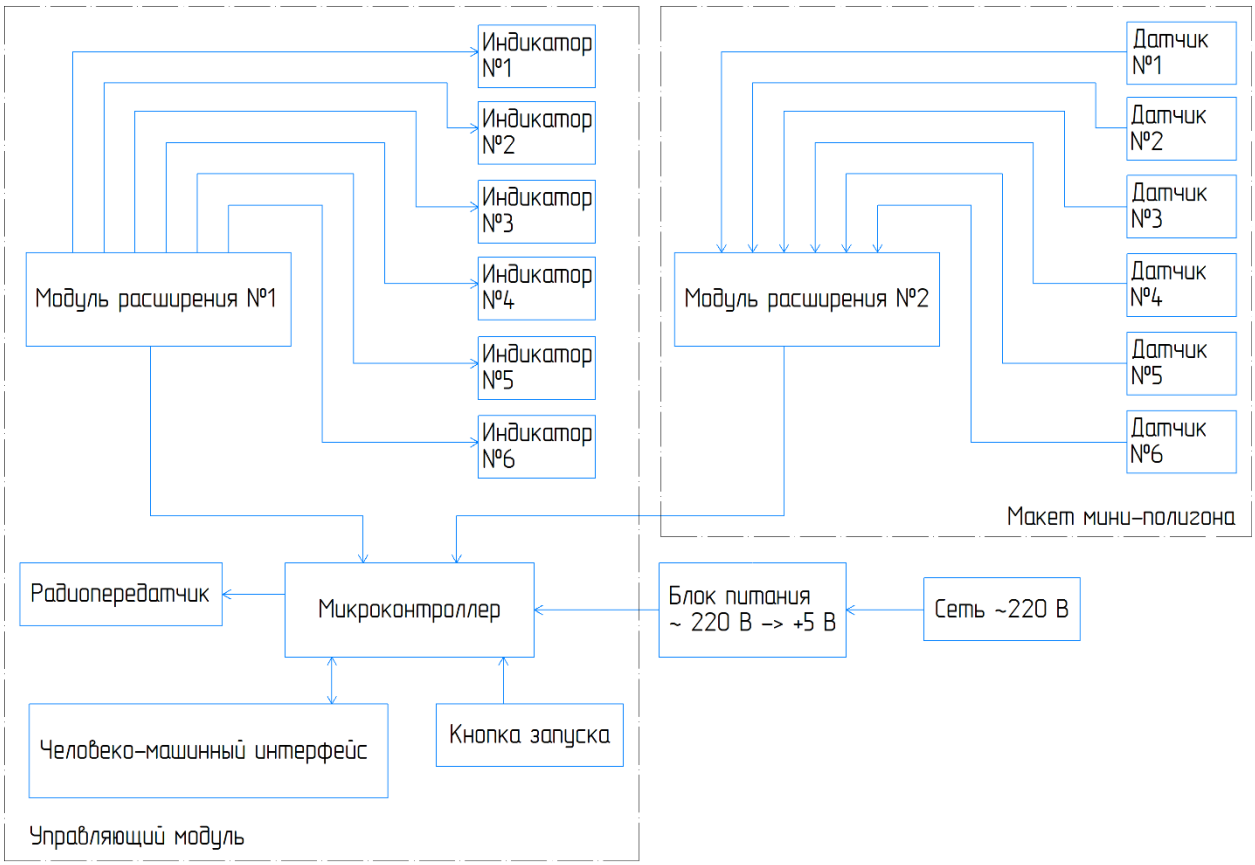

Рисунок 2.1 – Структурная схема мини-полигона

Тем самым данная структурная схема отображает соединения и пути взаимодействия компонентов устройства. Блок питания преобразует напряжение 220 В из сети переменного напряжения в необходимые +5 В постоянного, которое питает через плату с микроконтроллером все электронные компоненты схемы. Микроконтроллер осуществляет управление этими компонентами. Человеко-машинный интерфейс позволяет взаимодействовать с оператором мини-полигона. Благодаря ему предоставляется возможность изменения маршрута мобильной платформы. С помощью кнопки запуска производится запуск и остановка работы макета. С датчиков через второй модуль расширения на микроконтроллер поступает сигнал об приближении мобильного робота. Через первый же модуль расширения с помощью электронных индикаторов отображается для оператора развилка, на которой в данный момент находится мобильная

платформа. Радиопередатчик осуществляет связь с мобильной платформой для передачи управляющих команд.

2.2. Выбор необходимых комплектующих

2.2.1. Выбор микроконтроллера

На рынке существует большое количество разных микроконтроллеров, под различные задачи и ценовые сегменты.

Одним из наиболее популярных микроконтроллеров среди новичков в программировании является Arduino (рисунок 2.2). Он имеет множество модификаций и плат расширений, обладает огромным сообществом, постоянно публикующим полезные библиотеки в свободный доступ. Так же он является легкодоступным для покупателя.

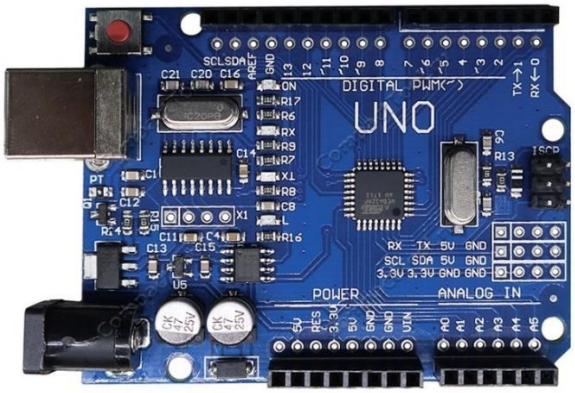

Рисунок 2.2 – микроконтроллер Arduino Mega

Данный микроконтроллер программируется на языке C++. Платы Arduino оснащены наборами цифровых и аналоговых выводов. Платы оснащены интерфейсами последовательной связи, включая универсальную последовательную шину (USB) на некоторых моделях, которые также используются для загрузки программ с персональных компьютеров. Было решено использовать именно микроконтроллер Arduino. Его производительности вполне достаточно для считывания показаний с

датчиков, а большой выбор плат расширений позволяет легко передавать данные на мобильную платформу. А стоимость микроконтроллера Arduino ниже, чем стоимость практически любой подобной платы. Линейка плат Arduino включает в себя множество модификаций. Выбор осуществлялся между моделями Arduino Nano, UNO R3 и Mega. Их стоимости практически одинаковы, поэтому можно не брать их во внимание. Основные характеристики представлены в таблице 2.1:

Таблица 2.1 – Характеристики Arduino Mega, UNO R3 и Nano.

Характеристики	Arduino Mega	Arduino UNO	Arduino Nano
Микроконтроллер	ATmega1280	ATmega328	ATmega328
Тактовая частота	16 МГц	16 МГц	16 МГц
Аналоговые контакты	8	8	8
Цифровые контакты	54	22	22
Диапазон аналоговых значений	От 0 до 5 В	От 0 до 5 В	От 0 до 5 В
Есть ли защита от коротких замыканий	Да	Да	Нет

Из таблицы 2.1 видно, что Arduino Mega отличается большим количеством контактов, однако 22 контактов для данного проекта вполне достаточно. Основное преимущество Arduino Nano в маленьких размерах. Но решающую роль в выборе модели сыграла защита от коротких замыканий. В процессе разработки и тестирования робототехнической платформы, нужно быть уверенным в том, что если произойдет разрыв изоляции проводов, микроконтроллер самостоятельно отключится, а не выйдет из строя. В Arduino UNO R3 в отличие от Arduino Nano такая защита есть. Таким образом, для работы была выбрана плата Arduino UNO R3.

2.2.2. Выбор датчиков

Существует большое количество датчиков, которые могут выступать в роли устройств считывания и сигнализировать об приближении роботизированной платформы.

Первым из рассмотренных вариантов был использование датчиков освещённости (рисунок 2.3), основанные на фоторезисторах, на макете мини-полигона и лазера на роботе.

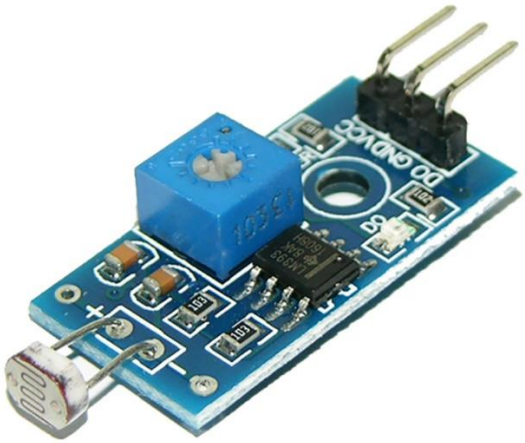

Рисунок 2.3 – Датчик света пороговый

Фоторезистор – это полупроводниковый прибор, сопротивление которого зависит от освещённости его чувствительной поверхности.

«Принцип действия заключается в следующем: между двумя проводящими электродами находится полупроводник, когда полупроводник не освещен – его сопротивление велико, вплоть до единиц МОм. Когда эта область освещена её проводимость резко возрастает, а сопротивление соответственно падает.» [1]

Из-за того, что датчики будут расположены на открытой поверхности, может возникнуть ситуация, когда степень освещённости превысит установленный на датчиках уровень и они одновременно сработают. В связи с этим, данный вариант был отвергнут.

Следующих вариантом на рассмотрение был использование датчиков на основе изменение магнитного поля.

Изначально выбор пал на модуль с герконом (рисунок 2.4).

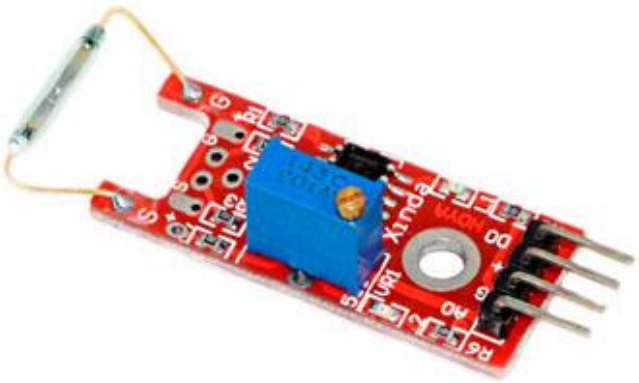

Рисунок 2.4 – Модуль KY-025 с герконом

Геркон (герметизированный контакт) – датчик магнитного поля. Внутри зелёной стеклянной колбы находятся подвижные контакты из особого сплава. Сама колба заполнена инертным газом. При воздействии магнитного поля контакты геркона меняют своё положение.

Данные датчики делятся по принципу своего действия на 3 типа:

- с нормально разомкнутым контактом;
- с нормально замкнутым контактом;
- с переключающимся контактом.

Как следует из названия, у первого типа герконов при воздействии магнитного поля контакты замыкают цепь, а у второго наоборот, размыкают. В случае с переключающихся, в нормальном состоянии подвижный контакт замыкает одну цепь, при воздействии поля переключается на другую.

Значительным недостатком у данных датчиков является хрупкость стеклянного баллона, чувствительность к ударам и вибрациям, что требует специальных мер по амортизации места установки герконов. Так как при наезде мобильной платформы на него или от вибрации во время её езды есть шанс повредить датчик, было решено этот вариант так же отказаться.

Окончательным вариантом было решено рассмотреть датчики Холла.

Датчики Холла работают на основе эффекта, открытого в 1879 году американским ученым Эдвином Холлом. Суть этого явления состоит в том,

что подав постоянное напряжение на противоположные края прямоугольной пластины (A и B на рисунке 2.5) и поместив ее в магнитное поле, возникает разность потенциалов на двух оставшихся краях (C и D).

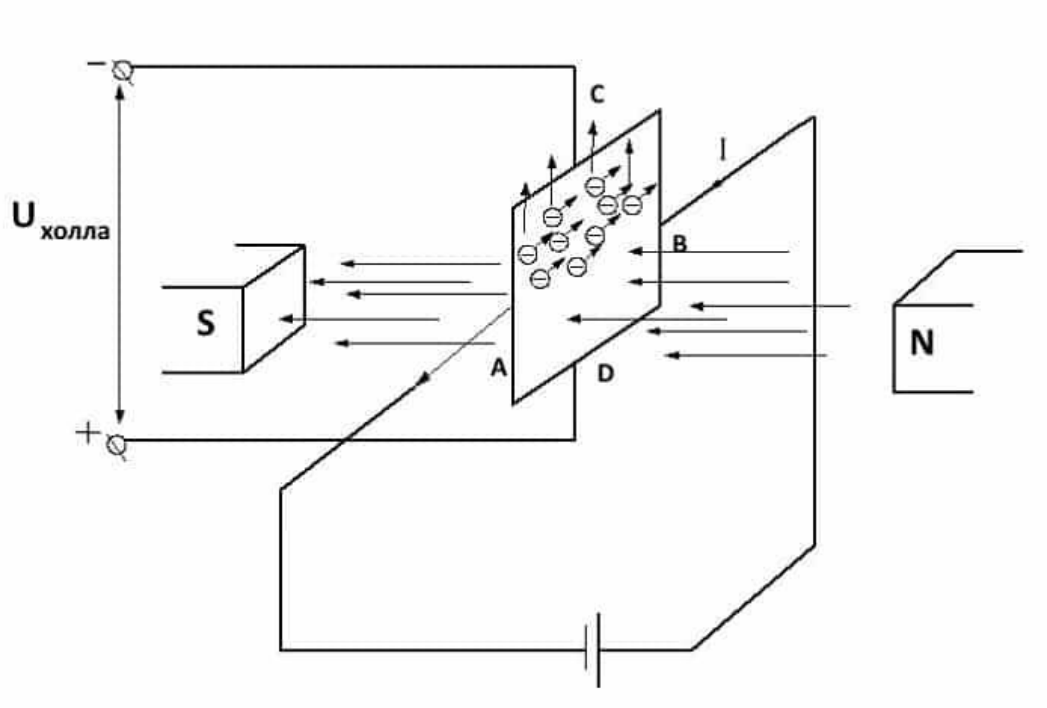

Рисунок 2.5 – Демонстрация работы эффекта Холла

Датчики Холла условно делятся на два вида:

- аналоговые – преобразуют магнитную индукцию в напряжение. Полярность и величина напряжения напрямую зависят от магнитного поля.
- цифровые – имеют всего два устойчивых положения: магнитное поле есть или нет. Срабатывает при достижении интенсивности магнитного поля определённой величины.

В свою очередь, цифровые делится на два подвида:

- униполярный – срабатывает при определенной интенсивности поля, после его понижения возвращается в начальное состояние.
- биполярный – один полюс включает прибор, второй выключает.

Так как магнит на мобильной платформе постоянно закреплен одной стороной, то для правильной работы мини-полигона необходимо поставить

цифровой униполярный датчик Холла. Был выбран SS441A 115G/20G (рисунок 2.6) так как он соответствует критериям и имеет низкую цену.

Рисунок 2.6 – датчик Холла SS441A 115G/20G

Технические характеристики SS441A:

- Тип выходного сигнала: цифровой с ок;
- Тип чувствительного элемента: элемент холла;
- Наличие встроенного магнита: нет;
- Тип чувствительности к полю: омниполярный;
- Индукция вкл, Гаусс: 115;
- Индукция выкл, Гаусс: 20;
- Макс рабочая частота, кГц: 100;
- Мин напряжение питания, В: 3.8;
- Макс напряжение питания, В: 30;
- Макс выходной ток, мА: 20;
- Температурный диапазон, гр. С: -40…150;
- Корпус: ss41;
- Вес, г: 0.12.

Датчик Холла имеет три ножки: питание, земля и информационный. При подключении датчика необходимо соединить ножки с питанием и

информационную подтягивающий резистором на 10 кОм. Схема подключения изображена на рисунке 2.7.

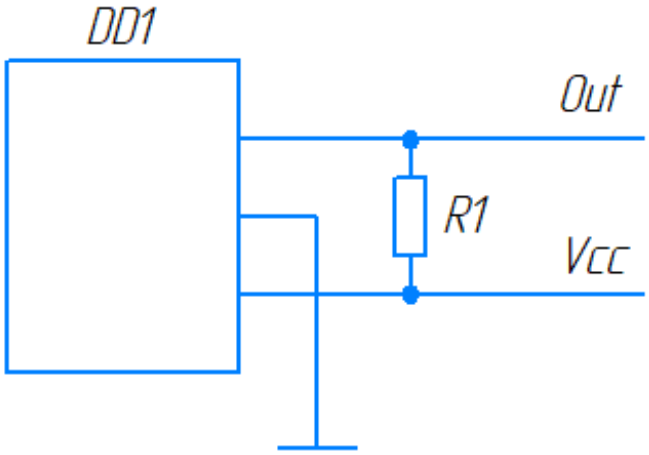

Рисунок 2.7 – Схема подключения датчика Холла

2.2.3. Выбор платы расширения

Чтобы иметь возможность расширять в будущем макет мини-полигона, было решено использовать платы расширения I/O. Был рассмотрен вариант на сдвиговых регистрах, так как они просты в решении, дешевые и многократно описанные, но они имеют ряд недостатков: каждая микросхема занимает дополнительный порт для SELECT, несмотря на то, что и так используется уже 3 для MOSI, MISO, SCK; нельзя их одновременно использовать для ввода или вывода; при монтаже приходится прокладывать 4 провода к каждой из них.

Чтобы не иметь эти недостатки была выбрана плата расширения с чипом PCF8574 на основе двухпроводного интерфейса I2C (рисунок 2.8).

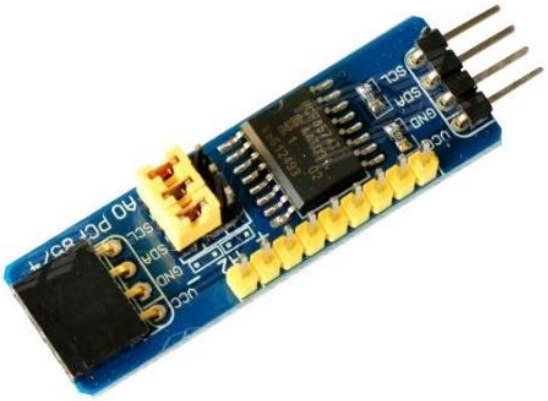

Рисунок 2.8 – плата расширения PCF8574

Данная плата имеет следующие технические характеристики:

- микросхема: PCF8574;
- напряжение питания: 3 – 5.5 В;
- интерфейс: I2C;
- количество портов GPIO: 8;
- размеры: 55 х 15 х 12 мм;
- вес: 5 грамм.

Она позволяет увеличить количество портов на 8 штук. Так же есть возможность увеличение портов до 64 штук при подключении несколько таких же плат. На самом модуле распаяны два подтягивающих резистора на 1 кОм, которые нужны для работы шины I2C.

Изначально на модуле установлен адрес 0x20, который при необходимости изменяется перемычками, что позволяет подключать до восьми устройств. Для его смены, необходимо поменять перемычку на (A0-A2), которая подтягивает линии A0, A1, A2 к питанию или наоборот. Ниже приведена таблица 2.2 с адресами.

Таблица 2.2 – смена адреса модуля

A0	A1	A2	Адрес
-	-	-	0x20
+	-	-	0x21
-	+	-	0x22
+	+	-	0x23
-	-	+	0x24
+	-	+	0x25
-	+	+	0x26
+	+	+	0x27

2.2.4. Выбор радиопередатчика

На данный момент есть несколько технологий передачи данных, которые работают с Ардуино, среди которых наибольшее распространёнными являются Wi-Fi и bluetooth.

Wi-Fi — это «аббревиатура, произошедшая от английского словосочетания Wireless Fidelity, означающая «беспроводная передача данных» или «беспроводная точность». Это система короткого действия, покрывающая десятки метров и которая использует не лицензированные диапазоны частот для обеспечения доступа к сети. Это протокол и стандарт на оборудование для широкополосной радиосвязи, предназначенной для организации локальных беспроводных сетей.» [2]

Основными стандартами Wi-Fi являются:

- IEEE 802.11 — определяет набор протоколов для самых низких скоростей передачи данных и является базовым стандартом WLAN.
- IEEE 802.11a — Протокол не совместим с 802.11b и несет в себе более высокие скорости передачи чем 11b. Использует частотные каналы в спектре 5GHz. Максимальная пропускная способность до 54Мбит/с.
- IEEE 802.11b — стандарт использует более быстрые скорости передачи и вводит больше технологических ограничений. Использует частотные каналы в спектре 2.4GHz. Максимальная пропускная способность до 11Мбит/с.

- IEEE 802.11g — стандарт использует скорости передачи данных эквивалентные 11a. Используются частотные каналы в спектре 2.4GHz. Протокол совместим с 11b. Максимальная пропускная способность до 54Мбит/с.
- IEEE 802.11n — на данный момент это самый передовой коммерческий Wi-Fi стандарт, который использует частотные каналы в спектрах 2.4GHz и 5GHz. Совместим с 11b/11a/11g. Максимальная пропускная способность до 300 Мбит/с.

Bluetooth – это технология беспроводной передачи данных между устройствами на расстоянии от 1 до 100 метров. Его создали компании Ericsson, IBM, Intel, Toshiba и Nokia в 1998 года для беспроводной альтернативы кабелям RS-232. Впоследствии Bluetooth стала частью международного стандарта IEEE 802.15.1.

Работа данной технологии базируется на радиоволнах. Когда включается модуль, активируется радиопередатчик, работающий в диапазоне 2.4 ГГц. После своей активации, он начинает искать все сигналы в данном диапазоне. Когда же он обнаруживает второй радиопередатчик, первый переходит в режим ведущего, в то время как второй в роли ведомого.

Существуют версии данноц данной технологии:

- Bluetooth 2.0 + EDR - скорость передачи данных 2,1 Мбит/с, до сих пор используется в недорогих гарнитурах;
- Bluetooth 2.1 и Bluetooth 2.1 + EDR - то же, что и в первом случае плюс поддержка NFC и уменьшено до 10 раз энергопотребление;
- Bluetooth 3.0 + HS - значительно увеличена скорость передачи данных до 24 Мбит/с, но возросло энергопотребление, популярности эта технология не снискала и в аудио устройствах встречается очень редко;
- Bluetooth 4.0 - уменьшено энергопотребление с сохранением скорости 24 Мбит/с - встречается в относительно дорогих моделях наушников, позволяет сохранить качество звука на высоком уровне;

- Bluetooth 4.1 - появилась защита от перекрестных помех при совместной работе с LTE-модулями, установленными во всех 4G смартфонах;
- Bluetooth 4.2 - увеличение скорости и улучшена защита передачи данных, встречается в относительно дорогих моделях наушников и аудио плеерах Hi-Fi, позволяет сохранить качество звука на высоком уровне;
- Bluetooth 5.0 - по сравнению с предыдущей версией увеличен радиус действия в 4 раза, скорость увеличена в 2 раза.

Среди модулей, работающих на технологии Wi-Fi, выделяются основанные на китайской микросхеме ESP8266. Среди них чаще всего используется ESP 01, который был выбран для сравнения. Среди Bluetooth-модулей были взяты следующие устройства: HC-05, HC-06 и HM-10. Сравнение их характеристик представлена в таблице 2.3.

Таблица 2.3 – Сравнение характеристик HC-05, HC-06 и HM-10

Характеристики	ESP-01	HC-05	HC-06	HM-10
Протокол	Wi-Fi 802.11 b/g/n	Bluetooth 2.0	Bluetooth 2.0	Bluetooth 4.0
Режим AT-команд	Да	Да	Нет	Да
Продолжительная работа	Да	Да	Да	Нет
Ток потребления	220 мА	50 мА	50 мА	50 мА
Дальность	40 м	10 м	10 м	10 м

Модуль ESP-01 не подходит из-за высокого тока потребления, так как вся система будет питаться напрямую от микроконтроллера. У модуля HC-06 отсутствует режим AT-команд, который необходим для перевода в Master-режим. HM-10 был исключен из-за невозможности продолжительной работы. В результате был выбран HC-05 (рисунок 2.9)

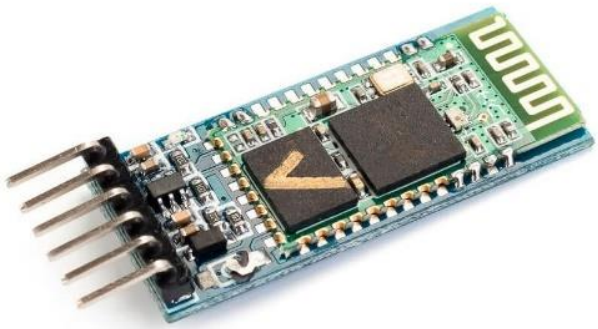

Рисунок 2.9 – Bluetooth-модуль HC-05

Технические характеристики модуля HC05

- чип Bluetooth: HC-05(BC417143);
- диапазон частот радиосвязи: 2,4 – 2,48 ГГц;
- мощность передачи: 0,25 – 2,5 мВт;
- чувствительность: – 80 dBм;
- напряжение питания: 3,3–5 В;
- потребляемый ток: 50 мА;
- радиус действия: до 10 метров;
- интерфейс: последовательный порт;
- режимы: master, slave;
- температура хранения: –40…85 °C;
- рабочий диапазон температур: –25…75 °C;
- габариты: 27 x 13 x 2,2 мм.

Датчик имеет 6 выводов стандарта 2,54 мм (рисунок 2.10):

- VCC – (питание 3,6 – 6 В);
- GND – (земля).
- TXD, RXD – UART интерфейс;
- STATE – индикатор состояния;

- KEY – контакт для входа в режим программирования.

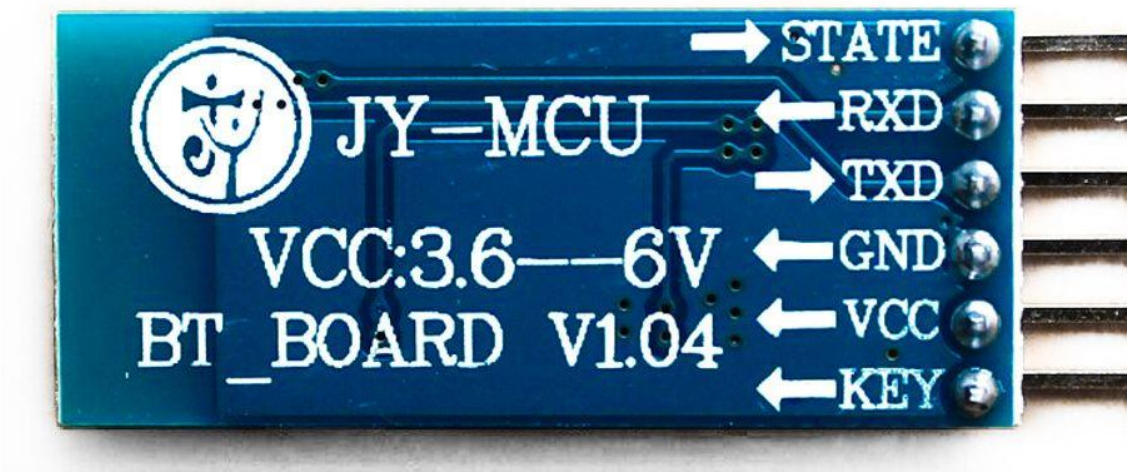

Рисунок 2.10 – Выводы датчика Bluetooth-модуля

2.2.5. Реализация человеко-машинного интерфейса

Существуют различные решения для построения пользовательского интерфейса. Можно использовать LCD-дисплей с джойстиком KY-023 или же составить клавиатуру из тактовых кнопок к нему. Так же можно приобрести готовое решение в виде LCD Keypad Shield.

Выбор пал на последнее в связи с удобством подключения, программирования и экономии места (рисунок 2.11).

Рисунок 2.11 – LCD Keypad Shield

Данный шилд имеет следующие технические характеристики:

- тип дисплея: LCD 1602, символьный, 4-х битный режим.
- разрешение: 16×2 (две строки по 16 символов каждая). Знакоместо 5×8 точек.
- цвет дисплея: синий (возможны варианты с желтым и зеленым цветом). Буквы белого цвета.
- технология: STN, Transflective, Positive.
- контроллер дисплея: HD44780U.
- предельная частота обновления экрана: 5Гц
- питание дисплея: 5 Вольт
- кнопки: 6 кнопок (5 кнопок управления и Reset).
- дополнительные элементы: регулировка яркости подсветки (потенциометр).
- рабочая температура экрана: от -20 °C до +70 °C;
- температура хранения экрана: от -30 °C до +80 °C.

Для работы с экраном используются цифровые пины D4 – D10, для обработки сигнала с кнопок используется лишь один аналоговый пин A0. Значения уровня сигнала этих кнопок расписаны в таблице 2.4.

Таблица 2.4 – Значения уровня сигнала на пине A0 в зависимости от выбранной кнопки

Выбранная кнопка	Значение уровня сигнала
RIGHT	0-100
UP	100-200
DOWN	200-400
LEFT	400-600
SELECT	600-800
Клавиша не нажата	800-1023

2.2.6. Электронные индикаторы

Систему индикации решено было сделать на основе светодиодов, так как система позиционирования отслеживает положения мобильной платформы только на ключевых точках макета мини-полигона.

Светодиоды – это «одни из самых популярных электронных компонентов, использующиеся практически в любой схеме. Являются полупроводниковыми приборами с электронно-дырочным переходом, создающие оптическое излучение при пропускании через них электрического тока в прямом направлении.» [3]

Излучаемый светодиодом свет лежит в узком диапазоне спектра. Диапазон излучения светодиода во многом зависит от химического состава использованных полупроводников.

Светодиод состоит из нескольких частей (рисунок 2.12):

- анод, по которому подается положительная полуволна на кристалл;
- катод, по которому подается отрицательная полуволна на кристалл;
- отражатель;
- кристалл полупроводника;
- рассеиватель.

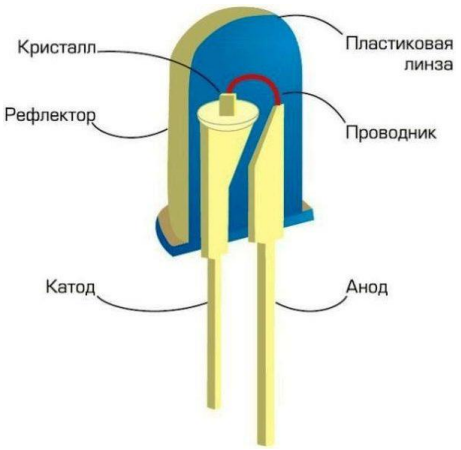

Рисунок 2.12 – части светодиода

В роли светодиодов были выбраны GNL-5013GT (рисунок 2.13) в связи с их оптимальным соотношением цены/максимальной силы света.

Рисунок 2.13 – светодиод GNL-5013GT

Технические характеристики данного светодиода:

- цвет свечения: зеленый;
- длина волны,нм: 565;
- максимальная сила света Iv макс.,мКд: 30;
- при токе Iпр.,мА: 20;
- видимый телесный угол,град: 20;
- цвет линзы: зеленый прозрачный;
- размер линзы,мм: 5;
- вес, г : 0.3.

2.3. Разработка электрической принципиальной схемы

В качестве среды разработки принципиальной схемы была выбрана программа Компас 3D v19 Portable из-за её бесплатной системы распространения и легкости использования. В соответствии с исходными данными, структурной схемой и выбранными компонентами была

разработана схема электрическая принципиальная. Данная схема представлена на рисунке 2.14.

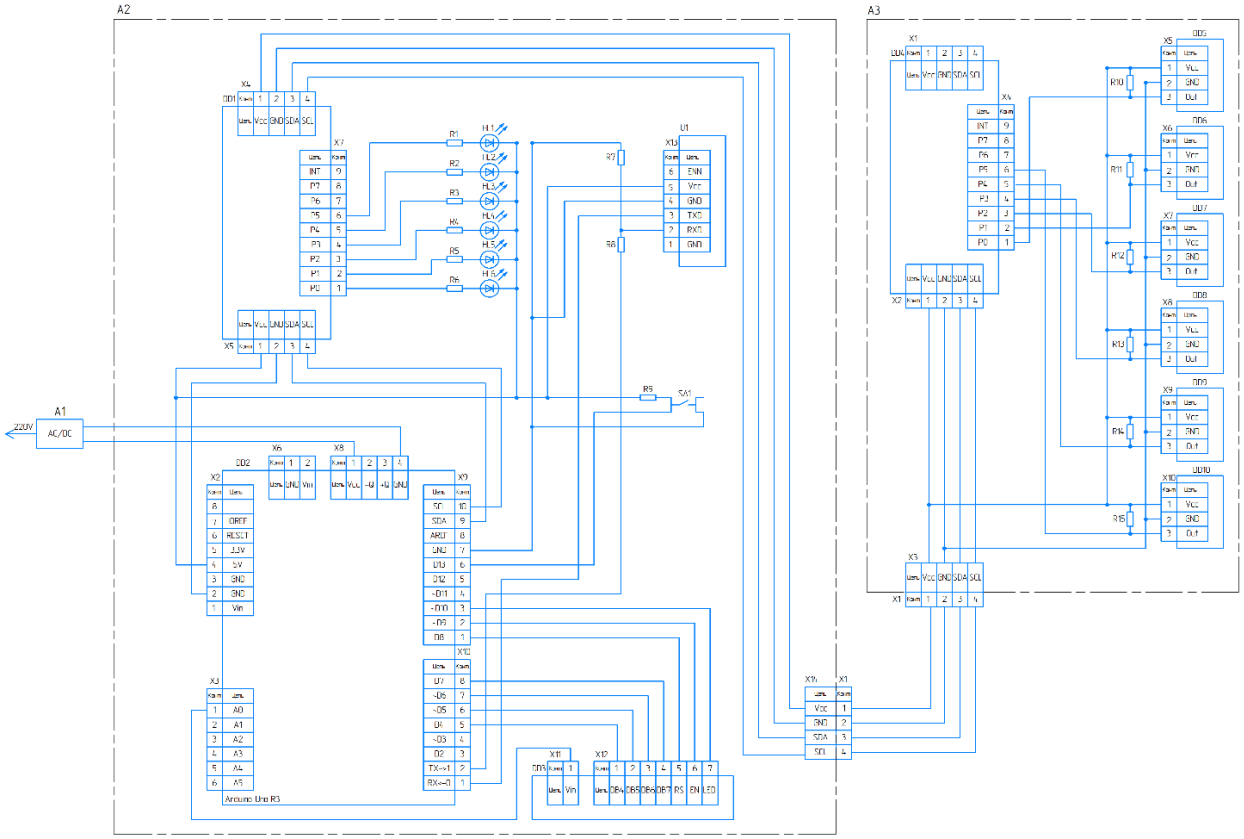

Рисунок 2.14 – Принципиальная схема мини-полигона

Перечень элементов, входящих в электрическую принципиальную схему, приведены в приложении А.

Вывод по разделу

В ходе разработки аппаратной части устройства была составлена структурная схема, которая удовлетворяет заданным условиям, были описаны функциональные узлы, из которых она состоит.

Следом были выбраны необходимые электронные компоненты, а именно: микроконтроллер Arduino UNO для управления системой; датчик Холла SS441A 115G/20G для регистрации приближения робота; плата

расширения PCF8574 для увеличения количества цифровых пинов; радиопередатчик Bluetooth-модуль HC-05 для связи с платформой; LCD Keypad Shield для человеко-машинного интерфейса и светодиод GNL-5013GT в качестве электронного индикатора.

После всей проделанной ранее работы, удалось создать электрическую принципиальную схему, состоящую из управляющей части, где задаётся маршрут для мобильного робота, и макета, который осуществляет сбор данных.

3. Разработка конструкции

Так же, как и в прошлом разделе для разработки опорных чертежей, на основании которых будет собираться конструкция, используется программное обеспечение КОМПАС-3D.

Основным решением было создание двух разных частей конструкции для простоты реализации аппаратно-программного комплекса. Первой частью являлся управляющий модуль, хранящий в себе основные компоненты по взаимодействию с пользователем. Второй же частью конструкции являлся сам макет, состоящий лишь из датчиков, платы и модуля расширения. Такое решение было принято во избежание одной большой многоуровневой конструкции, где при неисправности, например, одного датчика пришлось бы разбирать целую структуру.

Следующим остростоящим вопросом по разработке был выбор материала. Для реализации макета в реальность был выбран способ панельного сбора, подразумевающий строительный процесс макета при помощи панелей и плит. Самым приоритетным критерием по конструированию является оценочная стоимость материала.

Наиболее дешёвый вариантом материала является древесноволокнистая плита (ДВП) – листовый материал, в состав которого входят деревянистые волокна, в ходе изготовки которого они обдаются сушке и плотно спрессовываются. Главный недостаток этого материала заключается в его слабой огнестойкости из-за чего оказывается неподходящим.

Следующий рассматриваемый материал то же состоит из дерева, а точнее из деревянных щепок, которые спрессовываются в несколько слоёв и проклеиваются различными смолами и борной кислотой. Этот материал носит название ориентированно-стружечной плиты (ОСП), выдерживает высокое горение и не поддаётся деформациям на излом. Мобильная роботизированная платформа оснащена системой беспилотного управления,

в основании которой лежит устройство лидара с инфракрасным излучением, что делает материал не пригодным к изготовлению, потому что данный тип излучения поглощается деревом.

Ещё одним материалом может являться металл. Самым распространённым металлом в листовом варианте на рынке является алюминий, который не только может воздействовать на высокие температуры, но и обладает высокими антикоррозийные свойствами, имеет малый вес, высокую стойкость и пластичность. Все перечисленные свойства металла делают его стоимость выше остальных ранее перечисленных, поэтому материал так же не подходит.

Из рассмотренных материалов стоит подметить, что стоимость хоть и приоритетный критерий, но безопасность и возможность взаимодействия с мобильной платформой являются ещё одними ключевыми параметрами по выбору материала. Основываясь на всех характеристиках, удалось подобрать самый подходящий вариант, а именно пластик АБС, получивший своё название от трёх базовых мономенов входящих в его состав: акрилонитрила, бутадиена и стирола. Стоит подметить, что данный пластик легко поддаётся обработке и применим в изготовлении.

Конструкция управляющего модуля делается в виде полого блока для размещения внутри микроконтроллера и связанных с ним элементов. Как можно увидеть на рисунке 3.1 каждый элемент конструкции пронумерован и имеет собственное предназначение. Так на верхней поверхности расположено отверстие 1 предназначенное для доступа к регулятору яркости LCD дисплея, который будет располагаться в отверстии 2. Под цифрой 3 располагается шесть отверстий для светоиндикаци сработанного датчика, выполненная посредством светодиодов. В отверстие 4 будет крепится тактовая кнопка по запуску программы мини-полигона. Оставшийся набор кнопок по заданию маршрута будет располагаться в отверстии 5. Оставшиеся три отверстия сделаны с торца блока для проведения проводки. Провод (USB

А) ответственный за основное питание и загрузку программы с целью корректировки будет проходить через отверстие 6 и подключатся на прямую к микроконтроллеру. Для альтернативного же питания (JACK 5,5x2,1) сделано отверстие 7. При последующей передачи информации на макетную часть используется разъём DB9, который является встраиваемым в отверстие 8.

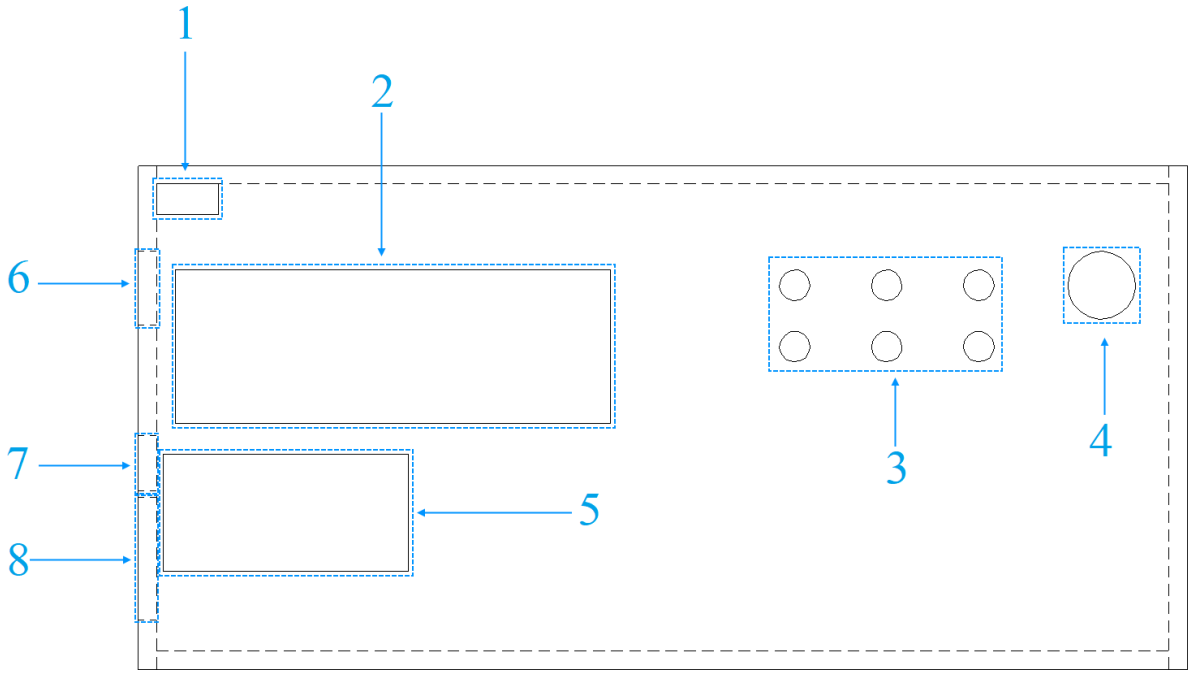

Рисунок 3.1 – Конструкция части управляющего модуля

На рисунке 3.2 можно увидеть точное расположение элементов, которые продублированы относительно осевой линии и полностью симметричны. Макетная часть конструкции оснащена гнездом DB9 под подключение экранированного кабель, идущего с управляющего модуля, для которого предназначено единственное отверстие 9. Главной особенностью этого кабеля является помехозащищённость от внешних факторов. Ещё макетная часть оснащена специальными площадками 10 для въезда/выезда мобильной платформа. Чтобы сформировать перекрёстки, требуются участки 11 по созданию коридоров. И последними не менее важными элементами были отверстия на поверхности дороги, которые предназначались для

лучшего взаимодействия датчиков Холла с магнитным полем. Стоить подметить что под каждый датчик сделано своё собственное отверстие и на рисунке выглядит как 11.

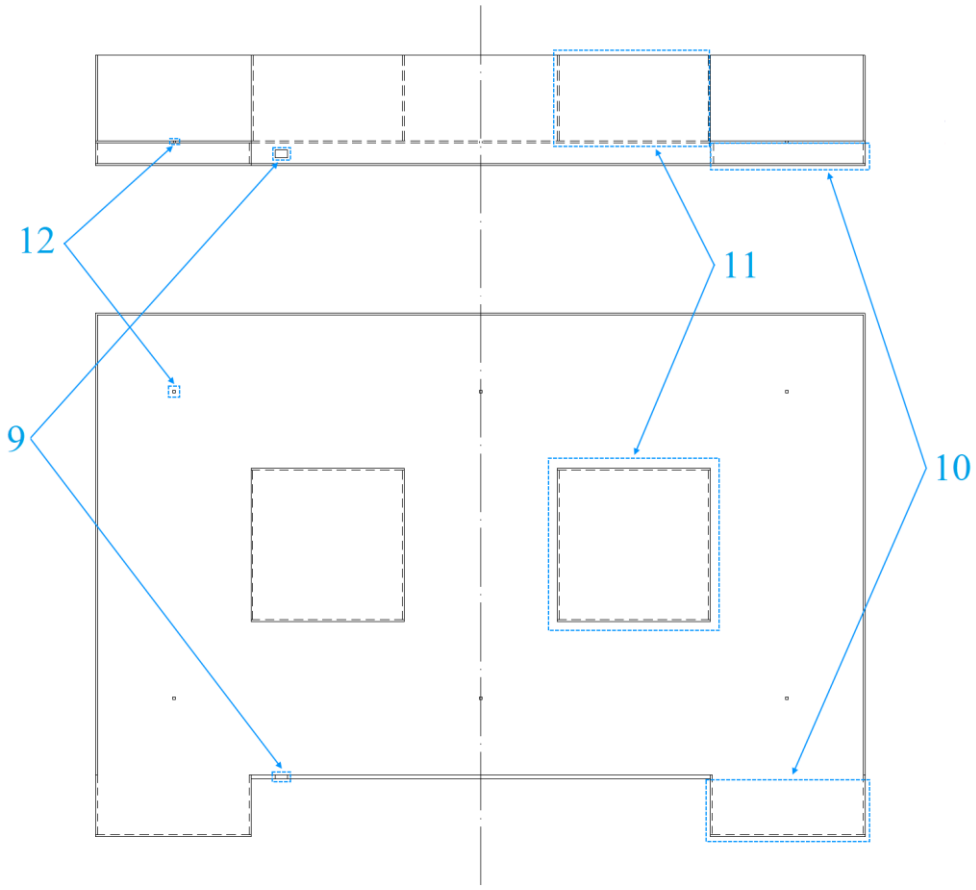

Рисунок 3.2 – Конструкция макетной части

Вывод по разделу

В данном раздели получилось определиться наиболее подходящим материалом и способом по разработке конструкции. В создании чертежей для персональной разработки, в очередной раз, использовалась САПР КОМПАС-3D из-за знакомого функционала и удобства интерфейса. Ключевым решением было разделить конструкцию на две части с дальнейшим описанием каждого элемента. Описание помогло наглядно понять, где будет располагаться тот или иной элемент.

4. Программная часть

4.1. Алгоритмизация

Алгоритмизацией является процесс составления разных алгоритмов для решения различных поставленных прикладных задач.

Сам же алгоритм есть не что иное, как точный набор инструкций, представляющих порядок действий некоторого исполнителя для достижения результата, решения некоторой задачи за конечное число шагов.

По замыслу, изначально нужно задать с помощью пользовательского интерфейса сторону поворота на каждом из перекрестков. После того, как все значения будут заданы, программа будет ждать нажатия кнопки Старт. При её нажатии на мобильную платформу отправляется команда запуска, затем запускается цикл проверки состояния каждого датчика Холла, которые должны реагировать на магнит, прикрепленный ко дну корпуса робота. Если какой-то из них сработал, то на микроконтроллер подается сигнал об изменении его состояния. Через Bluetooth-модуль отправляются данные с заранее заданным поворотом на мобильную платформу, а на управляющем модуле загорается соответствующий светодиод. При повторном нажатии на кнопку Старт, отправляется команда об завершении работы робота, цикл завершается и возвращается к началу первого. На рисунке 4.1 изображена блок-схема разработанного алгоритма.

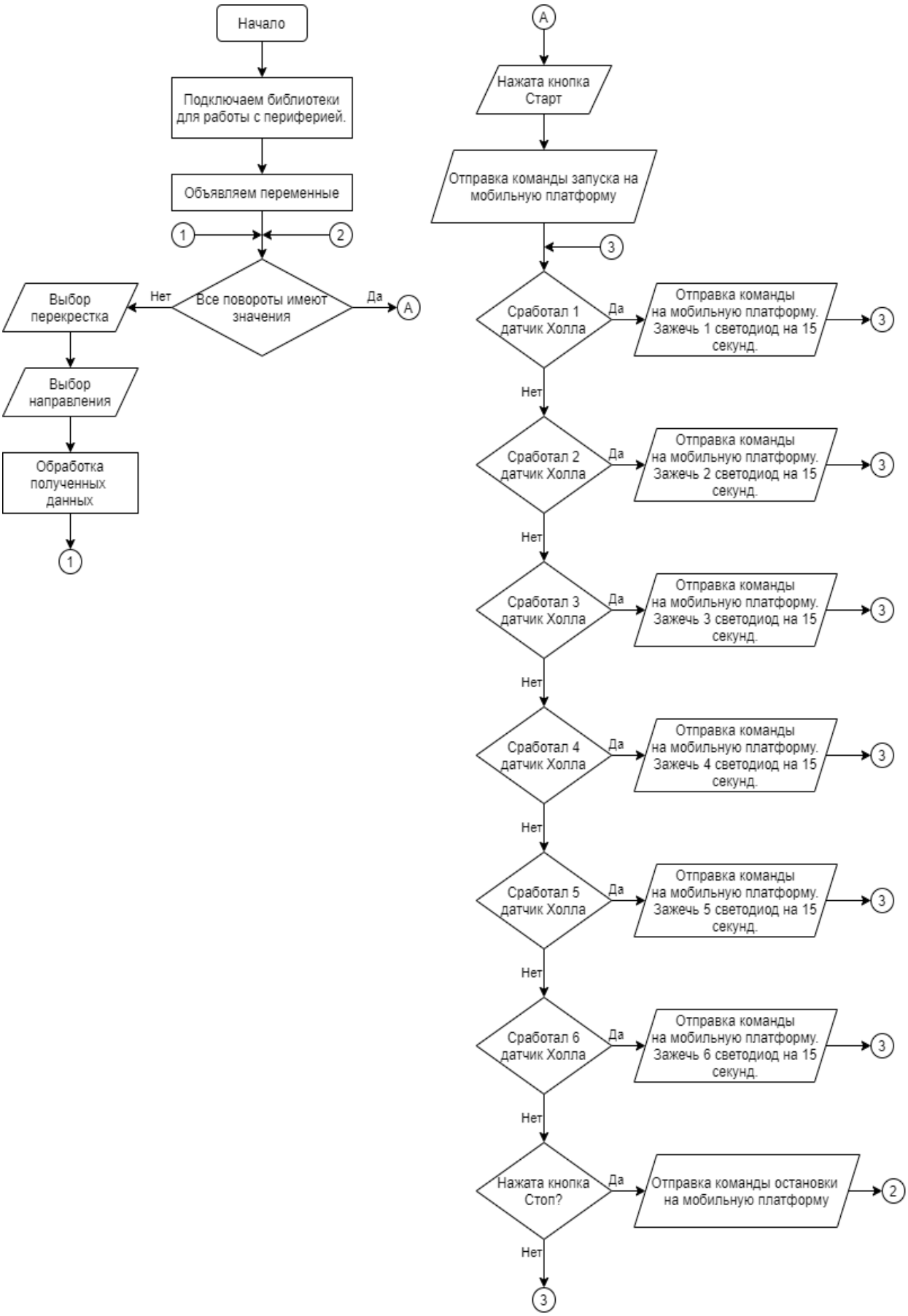

Рисунок 4.1 – Блок-схема алгоритма работы управляющего модуля мини-полигона

4.2. Разработка управляющей программы для устройства

Платформа Arduino использует язык программирования C++ с фреймворком Wiring. Само программирование происходит в программной оболочке Arduino IDE. Полностью написанная программа называется скетч, которая имеет расширение *.ino. В комплекте с оболочкой идут стандартные библиотеки, с прописанными функциями и объектами внутри. Так же есть возможность подключение своих библиотек, написанных на C++.

Код программы обязательно должен включать в себя функции setup() и loop(). Первая функция вызывается при запуске скетча, инициализирует переменные, определяет режимы работы выводов, запускает используемые библиотеки и т.д. Срабатывает один раз, после чего запускается вторая функция, которая работает в цикле и используется для активного управления платой Arduino.

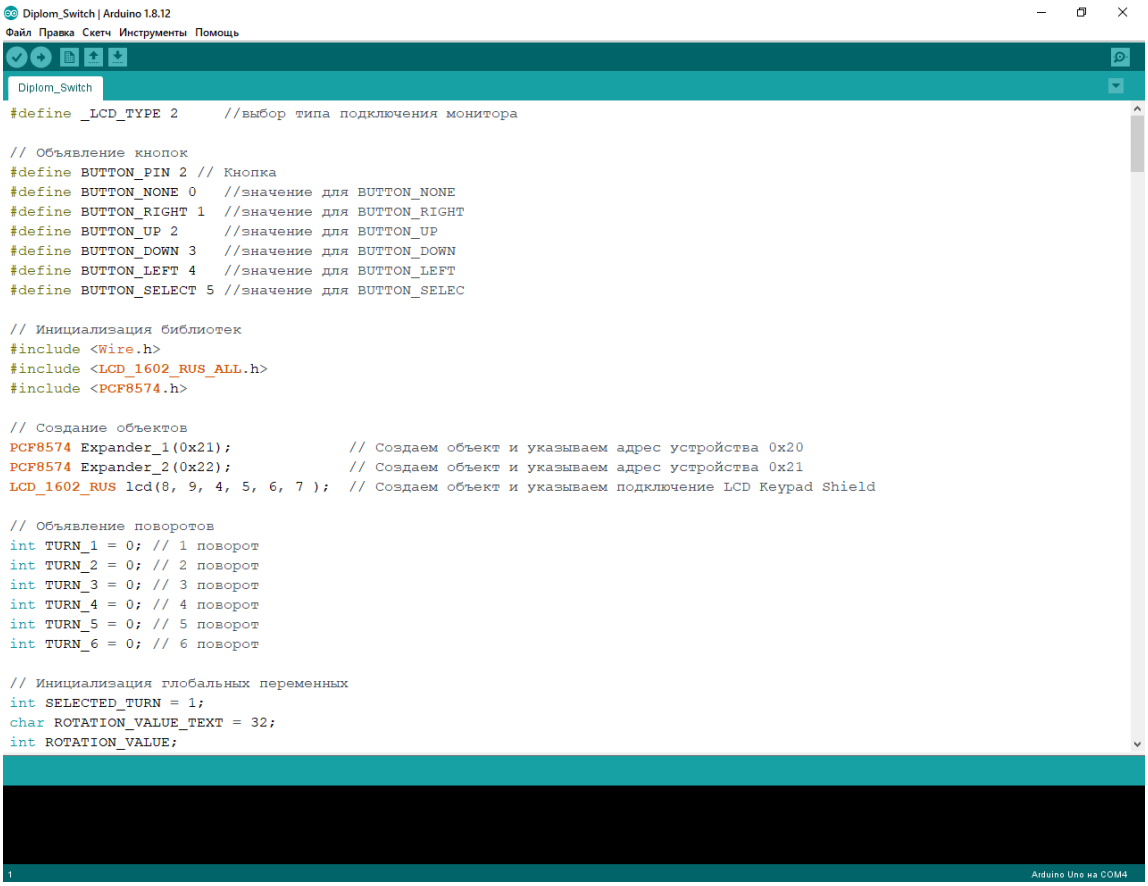

Рисунок 4.2 – Среда программирования Arduino IDE

Чтобы подключить нужный COM-порт, следует выбрать «Сервис» – «Последовательный порт» – «COM». Сам же микроконтроллер подключается через кабель USB-A – USB-B.

Для скетча были использованы следующие библиотеки:

- Wire.h – необходима для работы с устройствами по шине I2C;
- LCD_1602_RUS_ALL.h – служит для подключения и управления дисплея, позволяющая использовать русский язык в полном объёме;
- PCF8574.h – нужна для взаимодействия с модулем с чипом PCF8574.

На рисунке 4.3 представлены все необходимые для работы переменные.

```
#define _LCD_TYPE 2     // Выбор типа подключения монитора

// Объявление кнопок
#define BUTTON_PIN 2    // Тактовая кнопка
#define BUTTON_NONE 0   // Значение для BUTTON_NONE
#define BUTTON_RIGHT 1  // Значение для BUTTON_RIGHT
#define BUTTON_UP 2     // Значение для BUTTON_UP
#define BUTTON_DOWN 3   // Значение для BUTTON_DOWN
#define BUTTON_LEFT 4   // Значение для BUTTON_LEFT
#define BUTTON_SELECT 5 // Значение для BUTTON_SELEC

// Объявление поворотов
int TURN_1 = 0; // 1 поворот
int TURN_2 = 0; // 2 поворот
int TURN_3 = 0; // 3 поворот
int TURN_4 = 0; // 4 поворот
int TURN_5 = 0; // 5 поворот
int TURN_6 = 0; // 6 поворот

// Инициализация глобальных переменных
int SELECTED_TURN = 1;                  // Выбранный поворот
char ROTATION_VALUE_TEXT = 32;          // Вывод значение поворота
int ROTATION_VALUE;                     // Значение поворота
boolean TURN_SELECTION_MENU_GO = true; // Переменная для запуска меню выбора поворота
boolean ROTATION_VALUE_MENU_GO = true; // Переменная для запуска меню выбора направления поворота
boolean START_MENU_GO = true;           // Переменная для запуска меню старта
boolean START_GO = false;               // Переменная для запуска меню работы макета мини-полигона
```

Рисунок 4.3 – Объявление переменных

Были созданы объекты для модулей расширения и LCD-дисплея (рисунок 4.4). Для объектов модулей были прописаны адреса, для дисплея необходимые пины.

```
// Создание объектов
PCF8574 Expander_1(0x21);             // Создаем объект и указываем адрес устройства 0x20
PCF8574 Expander_2(0x22);             // Создаем объект и указываем адрес устройства 0x21
LCD_1602_RUS lcd(8, 9, 4, 5, 6, 7 );  // Создаем объект и указываем подключение LCD Keypad Shield
```

Рисунок 4.4 – Создание объектов

В базовой функции Setup (рисунок 4.5) была запущена работа последовательного порта, установлены режимы пинов для 1 и 2 расширителя и тактовой кнопки, а также уровни пинов на 1. Дальше запущена работа дисплея, после чего выведены имя автора, группа и название проекта.

```
void setup()
{
  Serial.begin(9600);

  Expander_1.pinMode(P0, OUTPUT);
  Expander_1.pinMode(P1, OUTPUT);
  Expander_1.pinMode(P2, OUTPUT);
  Expander_1.pinMode(P3, OUTPUT);
  Expander_1.pinMode(P4, OUTPUT);
  Expander_1.pinMode(P5, OUTPUT);

  Expander_1.digitalWrite(P0, HIGH);
  Expander_1.digitalWrite(P1, HIGH);
  Expander_1.digitalWrite(P2, HIGH);
  Expander_1.digitalWrite(P3, HIGH);
  Expander_1.digitalWrite(P4, HIGH);
  Expander_1.digitalWrite(P5, HIGH);

  Expander_2.pinMode(P0, INPUT_PULLUP);
  Expander_2.pinMode(P1, INPUT_PULLUP);
  Expander_2.pinMode(P2, INPUT_PULLUP);
  Expander_2.pinMode(P3, INPUT_PULLUP);
  Expander_2.pinMode(P4, INPUT_PULLUP);
  Expander_2.pinMode(P5, INPUT_PULLUP);

  pinMode(BUTTON_PIN, INPUT);

  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("Платонов А.С.");
  lcd.setCursor(0, 1);
  lcd.print("Элб-1702а");
  delay(2000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Мини-полигон");
  lcd.setCursor(0, 1);
  lcd.print("Версия 0.2.1");
  delay(2000);
  lcd.clear();
}
```

Рисунок 4.6 – Функция Setup

На рисунке 4.6 была прописана переменная для обработки сигнала с кнопок на LCD Keypad Shield. В переменную заносятся данные с аналогового пина, после чего результат сравнивается с заданными в документации на шилд значениями.

```
int getPressedButton()
{
  int buttonValue = analogRead(0);
  delay(180);

  if (buttonValue < 50) {
    return BUTTON_RIGHT;
  }
  else if (buttonValue < 150) {
    return BUTTON_UP;
  }
  else if (buttonValue < 300) {
    return BUTTON_DOWN;
  }
  else if (buttonValue < 500) {
    return BUTTON_LEFT;
  }
  else if (buttonValue < 800) {
    return BUTTON_SELECT;
  }
  return BUTTON_NONE;
}
```

Рисунок 4.6 – Переменная обработки сигнала кнопок

В базовой функции Loop была вызвана функция меню выбора поворота (рисунок 4.7).

```
void loop()
{
  TURN_SELECTION_MENU();
}
```

Рисунок 4.7 – Функция Loop

На рисунке 4.8 представлен код функции меню выбора поворота.

```
void TURN_SELECTION_MENU()
{
  if (TURN_1 != 0 && TURN_2 != 0 && TURN_3 != 0 && TURN_4 != 0 && TURN_5 != 0 && TURN_6 != 0)
  {
    START_MENU();
  }
  else
  {
    while (TURN_SELECTION_MENU_GO == true)
    {
      int BUTTON_TURN_SELECTION = getPressedButton();
      switch (BUTTON_TURN_SELECTION)
      {
        case BUTTON_UP:
          if (SELECTED_TURN  < 6)
          {
            SELECTED_TURN ++;
          }
          else
          {
            TURN_WRONG_UP();
          }
          break;
        case BUTTON_DOWN:
          if (SELECTED_TURN  > 1)
          {
            SELECTED_TURN --;
          }
          else
          {
            TURN_WRONG_DOWN();
          }
          break;
        case BUTTON_SELECT:
          TURN_SELECT();
          BUTTON_TURN_SELECTION = 0;
          break;
        default:
          lcd.setCursor(0, 0);
          lcd.print("Выберите поворот:");
          lcd.setCursor(0, 1);
          lcd.print(SELECTED_TURN, DEC);
      }
    }
  }
  ROTATION_VALUE_MENU();
}
```

Рисунок 4.8 – Функция меню выбора поворота

В начале функции проверяются состоянию переменных поворотов. Если все они не равны нулю, то есть пользователь выбрал направление для каждого из них, то вызывается функция запуска мини-полигона. Затем запускается цикл, который работает, пока значение переменной работы меню выбора поворота является истинной. Внутри данного цикла изначально

присваиваем значение новой переменной значение нажатой кнопки, после чего запускаем конструкцию с четырьмя блоками значений.

Если была нажата кнопка вверх, то проверяется значение переменной выбранного поворота. Если она ниже шести, то мы повышаем её значение на единицу. Если же выше, то вызывается функция сообщения об ошибке (рисунок 4.9).

```
void TURN_WRONG_UP()
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Значение");
  lcd.setCursor(0, 1);
  lcd.print("выше 6!");
  delay(2000);
  lcd.clear();
}
```

Рисунок 4.9 – Функция ошибки кнопки вверх

При нажатии кнопки вниз происходит та же проверка, что была описана выше, только условием является ниже единицы, в следствии чего значение переменной уменьшается на единицу. При несоответствии с условием вызывается следующая функция ошибки (рисунок 4.10).

```
void TURN_WRONG_DOWN()
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Значение ниже 1!");
  delay(2000);
  lcd.clear();
}
```

Рисунок 4.9 – Функция ошибки кнопки вниз

При нажатии на кнопку выбора вызывается функция обработки выбора поворота (рисунок 4.10). На дисплей выводится номер выбранного поворота,

после чего экран очищается. После чего значение переменной работы меню выбора поворота выставляется в ложное, в следствии чего прерывается работа цикла меню. Так же обнуляется состояние переменной нажатой кнопки.

```
void TURN_SELECT()
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Выбран поворот:");
  lcd.setCursor(0, 1);
  lcd.print(SELECTED_TURN, DEC);
  delay(1500);
  lcd.clear();
  TURN_SELECTION_MENU_GO = false;
}
```

Рисунок 4.10 – Функция обработки выбора поворота

Если же ни одна из кнопок не нажата, то на дисплей выводится сообщение об выбранной стороне на данный момент.

Когда цикл прерывается, вызывается функция меню выбора значения поворота (рисунок 4.11). В начале запускается цикл, который работает, пока значение переменной работы меню выбора значения поворота является истинной. Внутри цикла изначально присваиваем значение новой переменной значение нажатой кнопки, после чего запускаем конструкцию с шестью блоками значений. В первых четырех блоках при нажатии определенной кнопки переменным значения поворота и вывода значения поворота на дисплей присваивается соответствующее значение.

```
void ROTATION_VALUE_MENU()
{
  while (ROTATION_VALUE_MENU_GO == true){
    int BUTTON_ROTATION_VALUE_MENU = getPressedButton();

    switch (BUTTON_ROTATION_VALUE_MENU){
      case BUTTON_DOWN:
        ROTATION_VALUE_TEXT = 95;
        ROTATION_VALUE = 4;
        break;
      case BUTTON_UP:
        ROTATION_VALUE_TEXT = 94;
        ROTATION_VALUE = 1;
        break;
      case BUTTON_RIGHT:
        ROTATION_VALUE_TEXT = 62;
        ROTATION_VALUE = 2;
        break;
      case BUTTON_LEFT:
        ROTATION_VALUE_TEXT = 60;
        ROTATION_VALUE = 3;
        break;
      case BUTTON_SELECT:
        ROTATION_SELECT();
        BUTTON_ROTATION_VALUE_MENU = 0;
        ROTATION_VALUE_MENU_GO = false;
        break;
      default:
        lcd.setCursor(0, 0);
        lcd.print("Сторона поворота:");
        lcd.setCursor(0, 1);
        lcd.write(ROTATION_VALUE_TEXT);
    }
  }
  TURN_SELECT(ROTATION_VALUE);
}
```

Рисунок 4.10 – Функция меню выбора значения поворота

При нажатии на кнопку выбора вызывается функция обработки выбора направления поворота (рисунок 4.11). На дисплей выводится направление поворота, после чего экран очищается. Затем значение переменной работы меню выбора направления поворота выставляется в ложное, в следствии чего

прерывается работа цикла меню. Так же обнуляется состояние переменной нажатой кнопки.

```
void ROTATION_SELECT()
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Выбрана сторона:");
  lcd.setCursor(0, 1);
  lcd.write(ROTATION_VALUE_TEXT);
  delay(1500);
  lcd.clear();
}
```

Рисунок 4.11 – Функция обработки выбора направления поворота

Пока пользователь не взаимодействует с кнопками, на дисплее выводится сообщение о выбранной на данной момент стороне.

После окончания цикла вызывается функция обработки введённых данных и передаётся значение переменной значения поворота (рисунок 4.12). Создаётся конструкция из шести блоков, где при некоторой принимаемой цифре соответствующий ей поворот принимает значение выбранного направления. Затем значения переменной работы меню выбора поворота и переменной работы меню выбора направления поворота выставляются в истинное. Далее вызывается функция меню выбора поворота, пока пользователь не выберет направления для всех поворотов, после чего вызывается функция меню старта.

```
void TURN_SELECT(int VALUE)
{
  switch (SELECTED_TURN)
  {
    case 1: TURN_1 = VALUE; break;
    case 2: TURN_2 = VALUE; break;
    case 3: TURN_3 = VALUE; break;
    case 4: TURN_4 = VALUE; break;
    case 5: TURN_5 = VALUE; break;
    case 6: TURN_6 = VALUE; break;
  }

  ROTATION_VALUE_MENU_GO = true;
  TURN_SELECTION_MENU_GO = true;
  TURN_SELECTION_MENU();
}
```

Рисунок 4.12 – Функция обработки введённых данных

В данной функции (рисунок 4.13) запускается цикл, работающий до тех пор, пока значение переменной работы меню старта является истинной. В нём создаётся новая переменная для тактовой кнопки и присваивается ей значение с цифрового пина кнопки. Следом создаётся конструкция из 2 блоков. Пока кнопка не нажата, на дисплее показывается надпись об ожидании запуска. Если же она была нажата, то на дисплей отправляется сообщение об запуске мини-полигона. Переменная для запуска меню работы полигона объявляется истиной, в то время как переменная для работы меню старта ложной. Впоследствии запускается функция меню работы мини-полигона.

```
void START_MENU()
{
  while (START_MENU_GO == true)
  {
    int ButtonState = 0;
    ButtonState = digitalRead(BUTTON_PIN);
    switch (ButtonState)
    {
      case HIGH:
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Мини-полигон");
        lcd.setCursor(0, 1);
        lcd.print("запущен!");
        delay(2500);
        lcd.clear();
        START_GO = true;
        START_MENU_GO = false;
        break;
      default:
        lcd.setCursor(0, 0);
        lcd.print("Запуск полигона: ");
        lcd.setCursor(0, 1);
        lcd.print("СТАРТ");
    }
  }
  Start();
}
```

Рисунок 4.13 – Функция меню старта мини-полигона

В этой функции (рисунок 4.14) первоначально отсылается на мобильную платформу команда старта. Позже запускается цикл, работающий при значении истина переменной запуска меню работы мини-полигона. Затем создаётся переменная для отслеживания нажатия на тактовую кнопку. Следом переменные для записи состояния датчиков Холла. Дальше выводим на дисплей сообщение о работе полигона и вызываем функцию обработки данных от датчиков и передаём ей значения.

```
void START() {
  Serial.print(0);
  while (START_GO == true) {
    int ButtonState = 0;
    ButtonState = digitalRead(BUTTON_PIN);

    uint8_t val0x22 = Expander_2.digitalRead(P0);
    uint8_t val1x22 = Expander_2.digitalRead(P1);
    uint8_t val2x22 = Expander_2.digitalRead(P2);
    uint8_t val3x22 = Expander_2.digitalRead(P3);
    uint8_t val4x22 = Expander_2.digitalRead(P4);
    uint8_t val5x22 = Expander_2.digitalRead(P5);

    lcd.setCursor(0, 0);
    lcd.print("Работаем...");

    SENSOR_HALL(val0x22, val1x22, val2x22, val3x22, val4x22, val5x22);

    if (ButtonState == HIGH) {
      STOP_START();
      TURN_SELECTION_MENU();
    }
  }
}
```

Рисунок 4.14 – Функция меню старта

Данная функция (рисунок 4.15) проверяет состояние каждого датчика Холла. Если у некоторого из них оно поменялось, то на управляющем модуле загорается соответствующий светодиод, а на робота отправляется направление надлежащего поворота. После чего идёт ожидание, пока платформа проедет в заданном направлении, чтобы не возникло ситуации повторной отправки. Далее светодиод выключается.

```
void SENSOR_HALL(uint8_t SEN_0, uint8_t SEN_1, uint8_t SEN_2, uint8_t SEN_3, uint8_t SEN_4, uint8_t SEN_5)
{
  if (SEN_0 == 0) {
      Expander_1.digitalWrite(P0, LOW);
      Serial.print(TURN_1);
      delay(15000);
      Expander_1.digitalWrite(P0, HIGH);
    }
    if (SEN_1 == 0) {
      Expander_1.digitalWrite(P1, LOW);
      Serial.print(TURN_2);
      delay(15000);
      Expander_1.digitalWrite(P1, HIGH);
    }
    if (SEN_2 == 0) {
      Expander_1.digitalWrite(P2, LOW);
      Serial.print(TURN_3);
      delay(15000);
      Expander_1.digitalWrite(P2, HIGH);
    }
    if (SEN_3 == 0) {
      Expander_1.digitalWrite(P3, LOW);
      Serial.print(TURN_4);
      delay(15000);
      Expander_1.digitalWrite(P3, HIGH);
    }
    if (SEN_4 == 0) {
      Expander_1.digitalWrite(P4, LOW);
      Serial.print(TURN_5);
      delay(15000);
      Expander_1.digitalWrite(P4, HIGH);
    }
    if (SEN_5 == 0) {
      Expander_1.digitalWrite(P5, LOW);
      Serial.println(TURN_6);
      delay(15000);
      Expander_1.digitalWrite(P5, HIGH);
    }
}
```

Рисунок 4.15 – Функция обработки данных от датчиков

Когда были проверены все состояния датчиков, проверяется состояние кнопки. Если она была нажата, то вызывается функция остановки меню старта (рисунок 4.16). На платформу отправляется команда прекращении работы, после чего на дисплей выводится надпись об остановки работы мини-полигона. Все переменные для поворотов обнуляются, переменной выбранного поворота присваивается значение единицы, переменная запуска меню старта принимает значение истина, в то время как переменная запуска меню работы мини-полигона ложь. После чего вызывается функция меню выбора поворота и вся программа приходит к исходному состоянию.

```
void STOP_START()
{
  Serial.print(4);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Мини-полигон");
  lcd.setCursor(0, 1);
  lcd.print("остановлен!");
  delay(2500);
  lcd.clear();
  TURN_1 = TURN_2 = TURN_3 = TURN_4 = TURN_5 = TURN_6 = 0;
  SELECTED_TURN = 1;
  START_MENU_GO == true;
  START_GO = false;
}
```

Рисунок 4.16 – Функция остановки меню старта

В приложении Б приводится код разработанной программы.

Вывод по разделу

В данном разделе был разработан алгоритм работы управляющего модуля мини-полигона для более понятного отображения работы скетча. На основании него была составлена соответствующий скетч для микроконтроллера при помощи программной оболочки Arduino IDE. В нём описан процесс выбора перекрестка и поворота, запуск мини-полигона и его работа.

5. Конструкторско-экспериментальный раздел

Перед сборкой мини-полигона было решено разработать конструкторский чертеж для макета и сборочный для управляющего модуля. Для их разработки использовался Компас-3D.

Сборочный чертежом является конструкторским документом, который содержит изображение сборочной единицы и другие данные, необходимые для ее сборки (изготовления) и контроля. На рисунке 5.1 представлен результат работы.

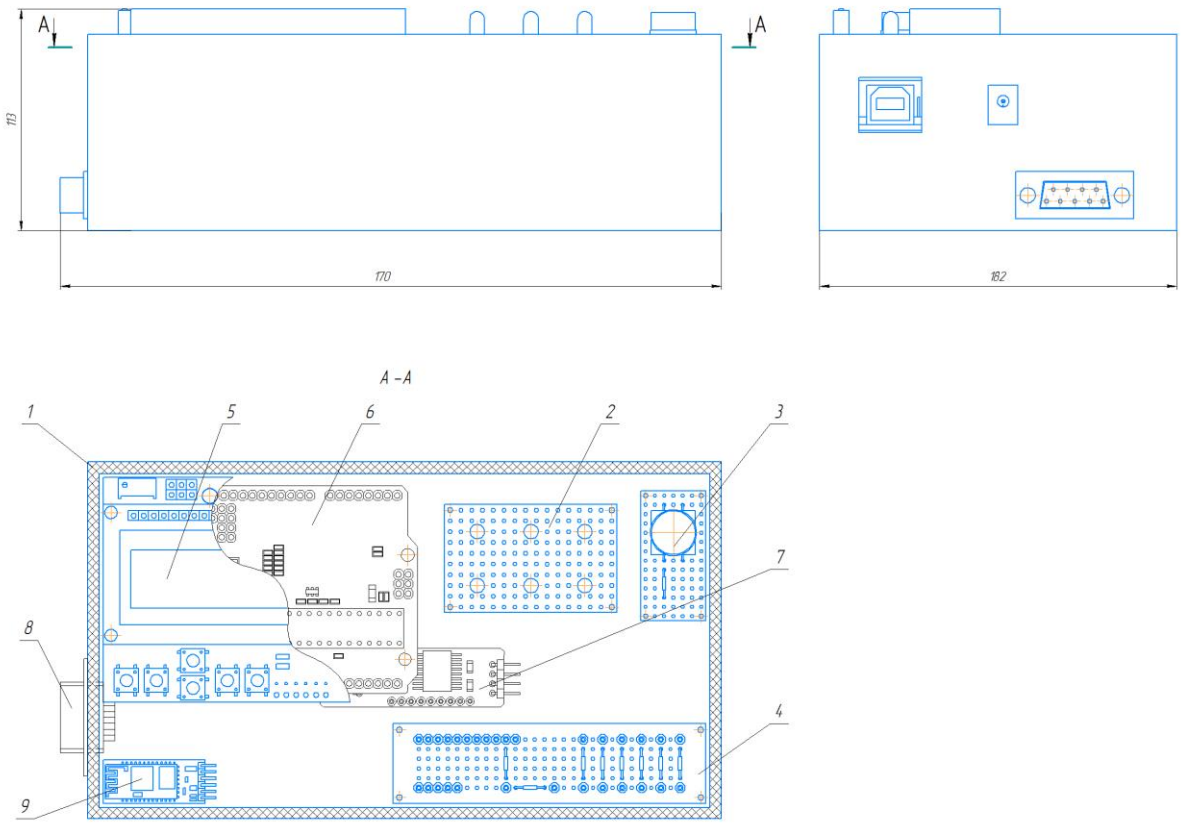

Рисунок 5.1 – Сборочный чертеж управляющего модуля

Модуль содержит 8 элементов, включая корпус, и имеет габариты 337x182x113 мм. Номером 1 является корпус блока, его чертёж детали представлен на рисунке 5.2 со всеми нужными для сборки размерами.

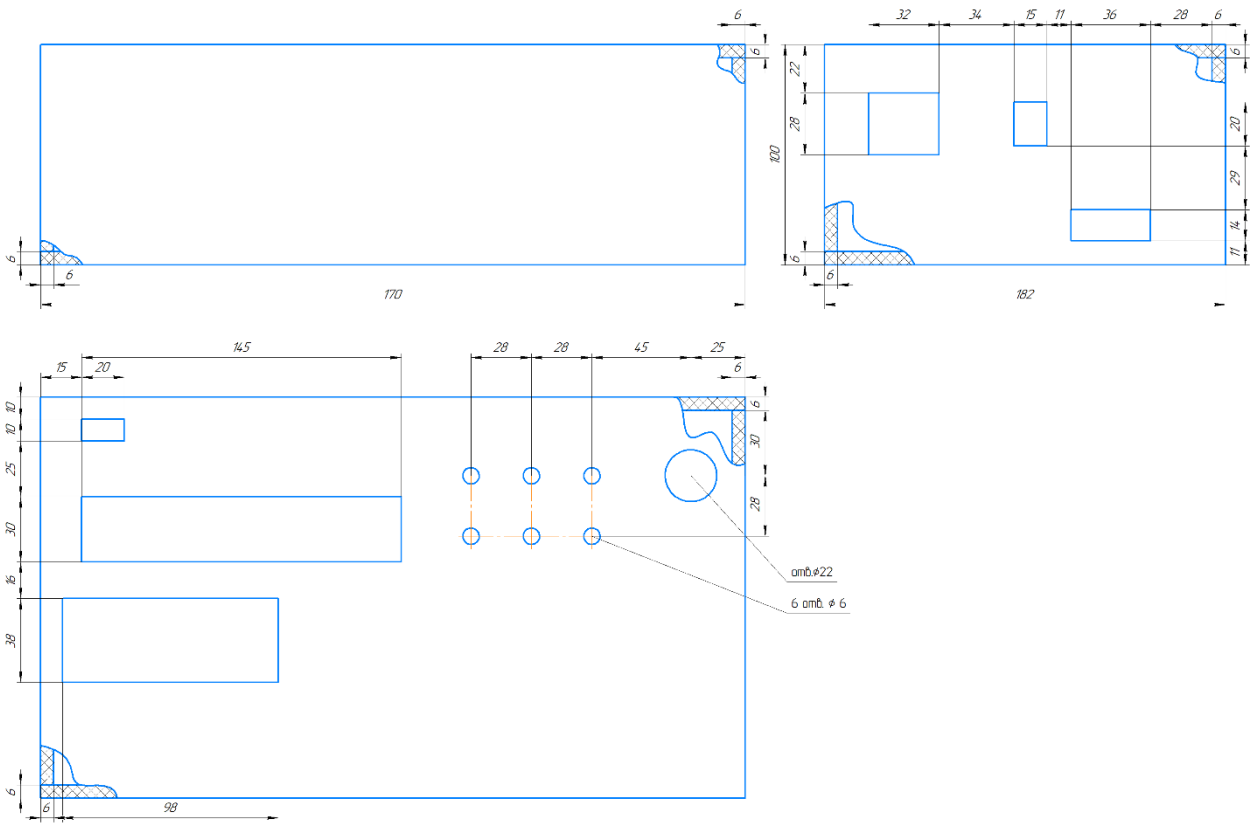

Рисунок 5.2 – Чертёж детали управляющего модуля мини-полигона

3 и 4 являются платами светодиодной (рисунок 5.3) и кнопочной (рисунок 5.4) соответственно. Все детали крепятся на макетные платы необходимого размера, после чего припаиваются. Перечень элементов, входящих в светодиодную плату, приведены в приложении Г. Для кнопочной платы приведены в приложении Д.

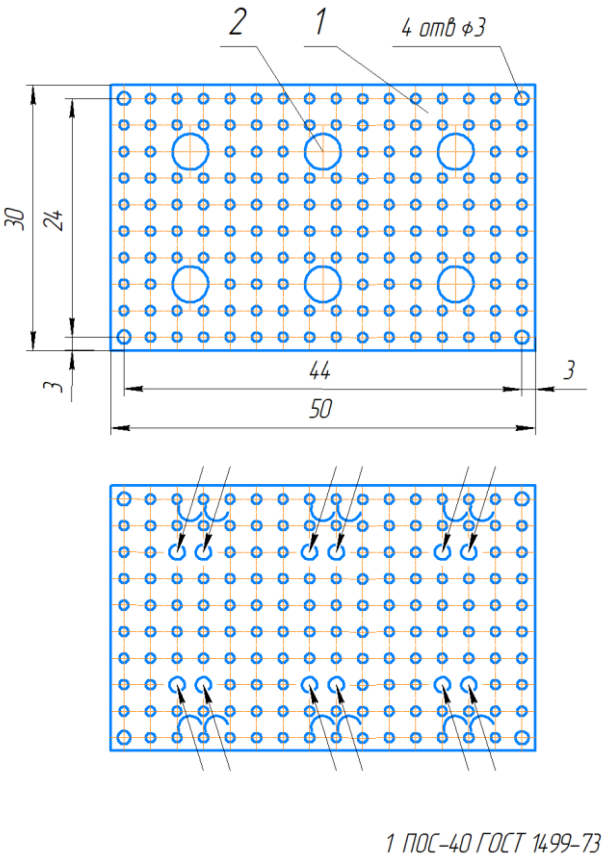

Рисунок 5.3 – Сборочный чертеж платы светодиодной

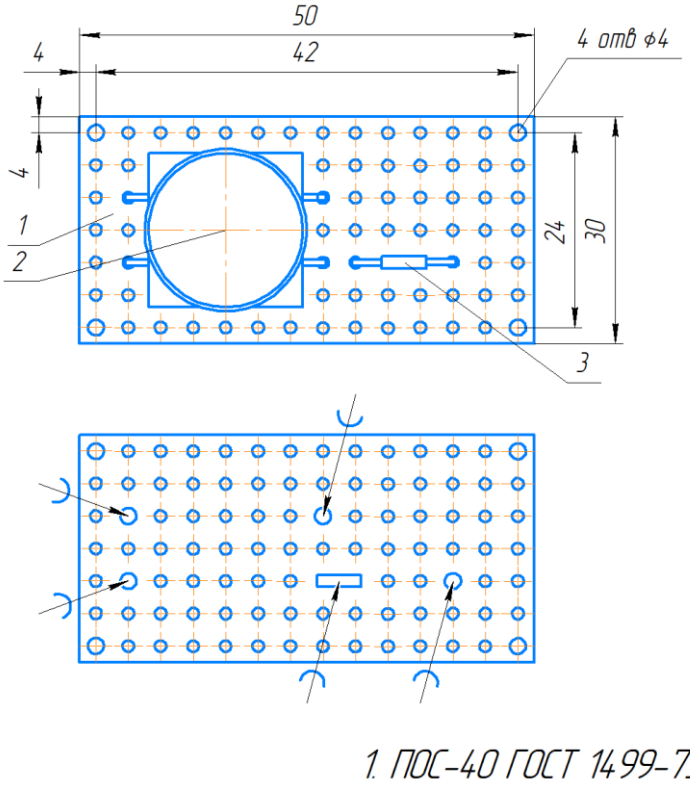

Рисунок 5.4 – Сборочный чертеж платы кнопочной

Под номером 5 представлен LCD Keypad Shield, который соединён с номером 6 Arduino UNO R3.

Все вышеперечисленные элементы крепятся к верхней крышке блока путём склеивания раствором из ацетона и стружки ABS.

Под номером 3 обозначена платой расширения (рисунок 5.5), собираемая из штырьков и резисторов. Перечень элементов, входящих в светодиодную плату, приведены в приложении Е. 7 является модуль расширения PCF8574, 9 представляет из себя Bluetooth-модулем HC-05. Данные элементы крепятся к нижней крышке аналогичным способом.

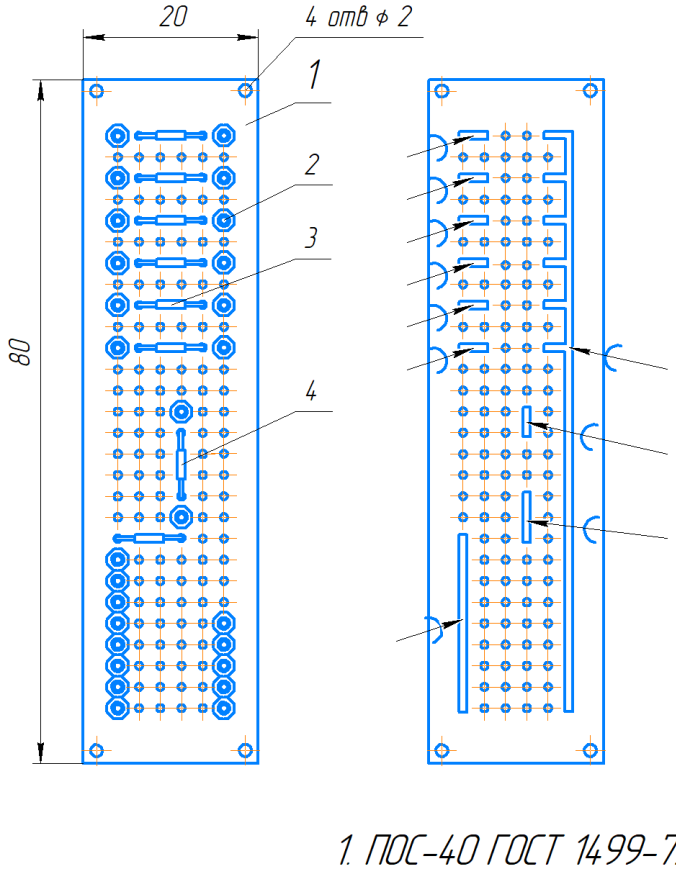

Рисунок 5.5 – Сборочный чертеж платы расширения

Номером 7 является разъём DB9, который крепится во входное отверстие на боковой стенке.

Перечень элементов, входящих в сборочный чертеж, приведены в приложении В.

На рисунке 5.6 представлен чертёж детали макета мини-полигона со всеми нужными для сборки размерами.

Исходя из чертежей было решено закупить 2 листа ABS с габаритами 2100x1500x2.

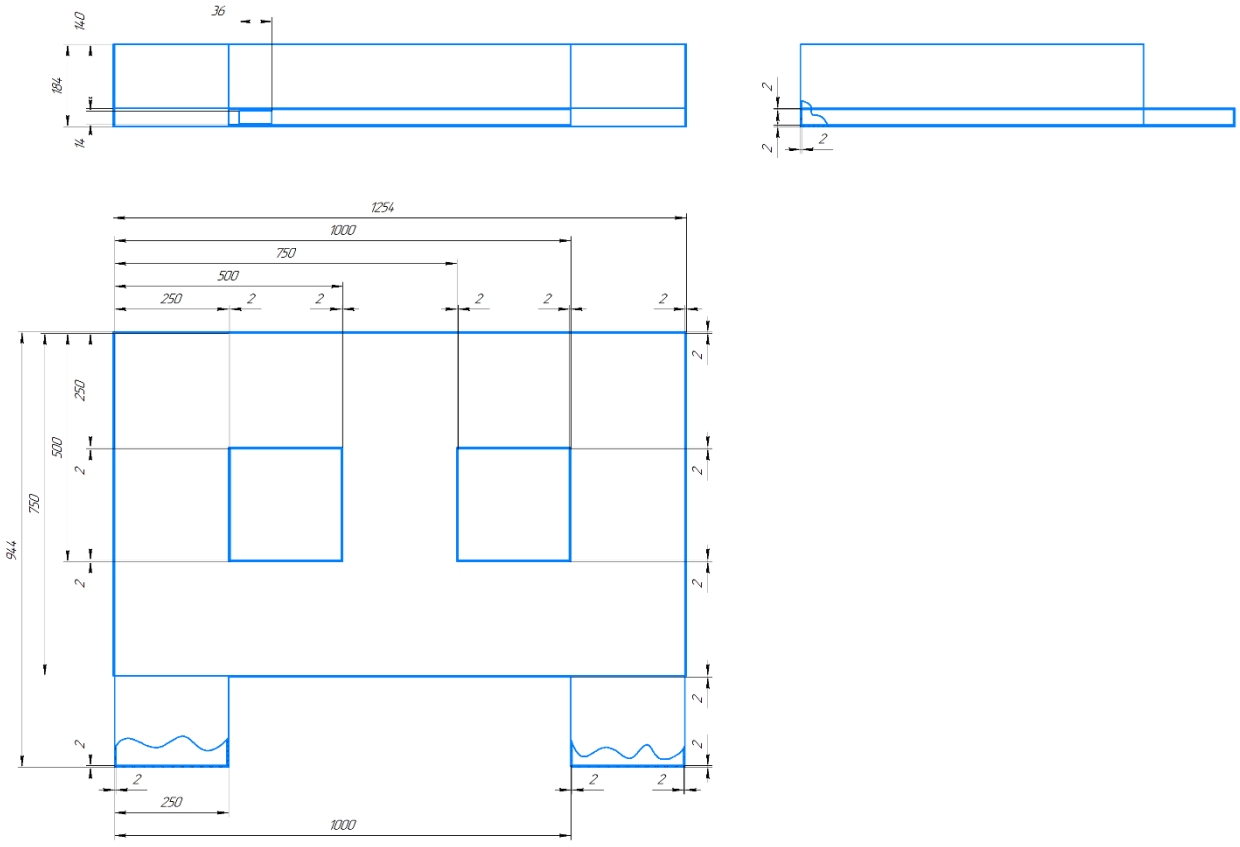

Рисунок 5.6 – Габаритный чертеж макета мини-полигона

5.1. Сборка управляющего модуля

Первым делом было решено собрать воедино всю электронику и протестировать работоспособность программы. Благодаря разработанной во 2 разделе принципиальной схеме удалось собрать воедино все компоненты устройства (рисунок 5.7). На рисунке 5.8 представлена работа устройства. При поднесении магнита к датчику Холла срабатывает соответствующий светодиод.

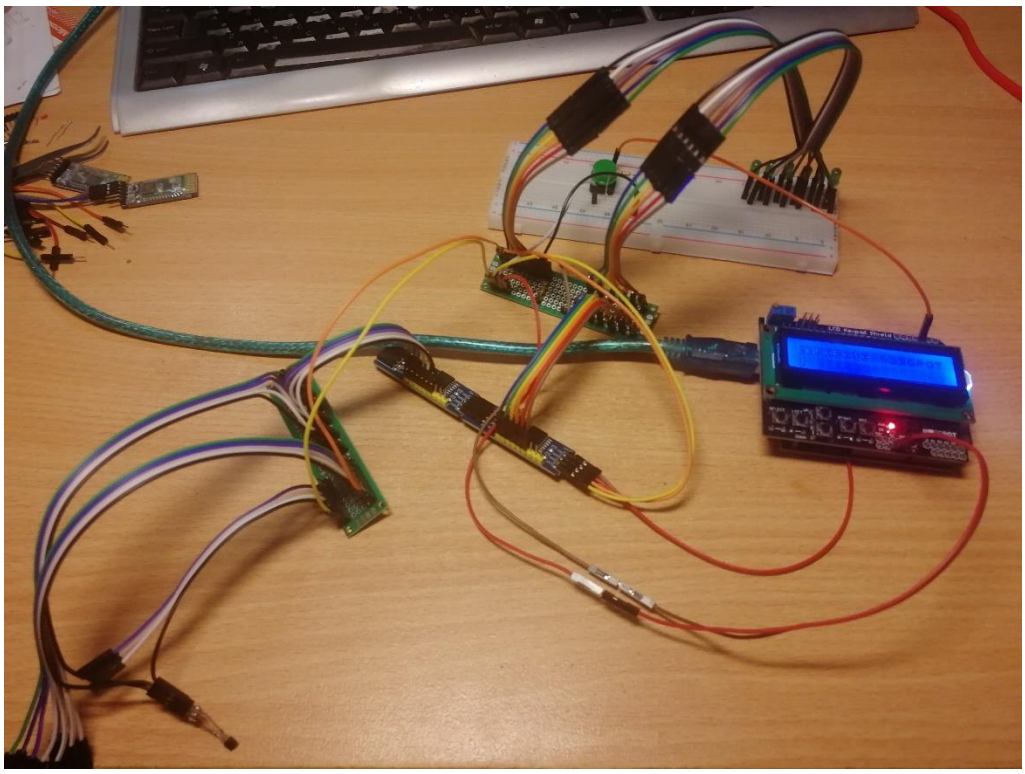

Рисунок 5.7 – Сборка всех компонентов устройства

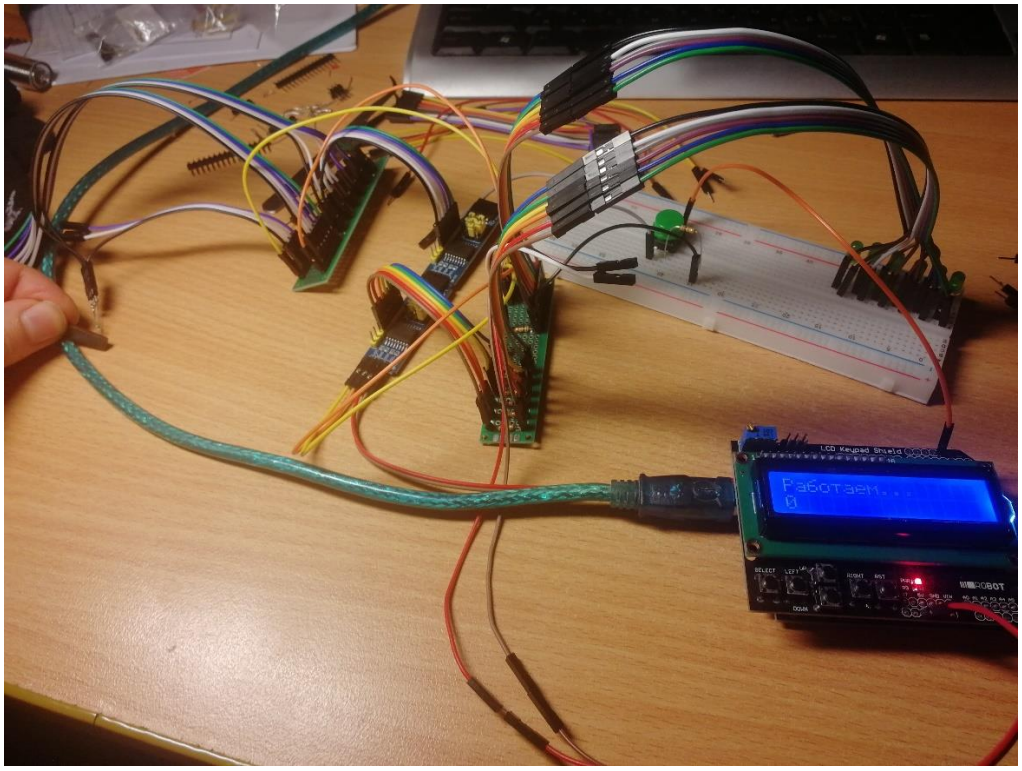

Рисунок 5.8 – Проверка работоспособности устройства

Для сборки модуля управления понадобилось вырезать 6 плоскостей. Так как у них малые размеры, было решено отрезать их при помощи ножниц для металла. Для получения отверстий для светодиодов, тактовой кнопки и разъёмов на Arduino UNO R3 использовалось переходное сверло, для остальных делались отверстия по углам прямоугольников, после чего периметр вырезался электролобзиком. Неровности выравнивались канцелярским ножом. После завершения работ края обрабатывались лобзиком от фаски, впоследствии зачищались шкуркой шлифованной. К верхней поверхности были приклеены все необходимы элементы. На рисунке 5.9 представлены результаты работы.

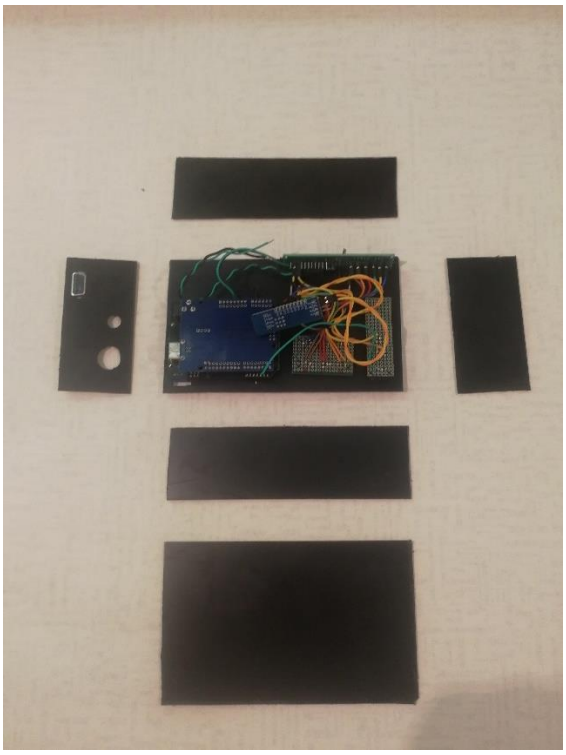

Рисунок 5.9 – Детали корпуса управляющего модуля

Сборка коробки началась с крепежом на нижнюю поверхность всех боковых частей, после чего каждая сторона выравнивалась относительно другой на 90 градусов. Затем к разъёму DB9 были припаяны все необходимые провода. Впоследствии была закреплена верхняя крышка и обработаны углы. Результат работы представлен на рисунке 5.10.

Рисунок 5.10 – Управляющий модуль в сборке

5.2. Сборка макета мини-полигона

Для сборки макета было необходимо нарезать 26 плоскостей. Так как они достаточно длинные, было решено использовать электролобзик. После его использования оставалась нагретая стружка, которая снималась канцелярским ножом.

После нарезки были установлены плоскости, что размещались по верхнему периметру макета. Все крепления поверхностей производились клеем на основе раствора ацетона и стружки пластик ABS. Для большей устойчивости было решено использовать внешние рёбра жёсткости.

Затем были склеены две коробки, которые были размещены внутри верхнего периметра на расстоянии 25 см от его плоскостей.

Впоследствии были установлены стенки для нижнего периметра. В передней стенке было сделано отверстие под разъем DB9.

Когда основная часть макета была собрана, по центру перекрестков были сделаны отверстия с помощью свёрла под датчики Холла. Под поверхностью полигона была проложена проводка к ним от модуля

расширения, что был на ней же закреплен вместе с модулем расширения PCF8574.

Последними были установлены пьедесталы для мобильной платформы для заезда оной.

Результат работы представлен на рисунке 5.11.

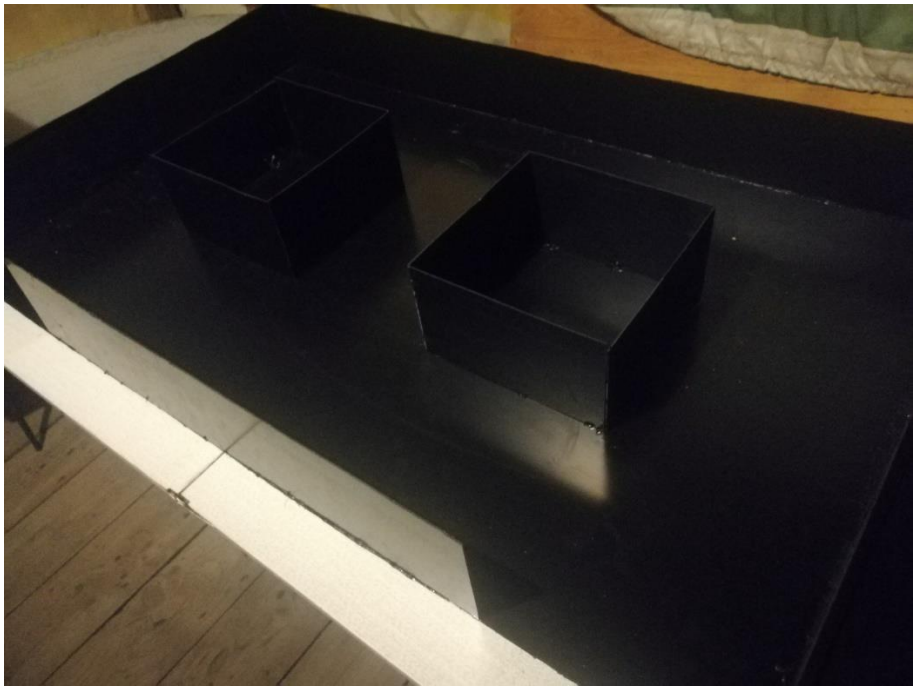

Рисунок 5.11 – Макет мини-полигона в сборке

Вывод по разделу

В результате проделанной работы в данном разделе, был разработан сборочный чертеж для управляющего модуля и деталировка к нему, после чего был собран сам модуль. Так же для макета был разработан конструкторский чертеж, на основании которого были произведены сборочные работы.

Заключение

В результате выполнения бакалаврской работы были выполнены все поставленные цели и задачи, в результате чего был разработан мини-полигон, способный задавать маршрут для мобильной платформы и менять его посредством человеко-машинного интерфейса.

В первом разделе были рассмотрены различные системы позиционирования и выбран метод, наиболее подходящий для решения нашей задачи.

Во втором разделе была составлена структурная схема, подобраны электронные компоненты и разработана принципиальная схема.

В третьем разделе был выбран материал для сборки управляющего модуля и макета мини-полигона, были разработаны чертежи для персональной разработки к каждому из них.

В четвертом разделе была разработана блок-схема и соответствующая ей программа на языках C/C++ для Arduino UNO R3.

В шестом разделе представлены результаты конструкторской деятельности и экспериментальных испытаний, созданы сборочный и конструкторские чертежи, была произведена сборка мини-полигона, выявлены и исправлены недочеты программной части.

Таким образом, готовый комплексный проект представляет собой мини-полигон, состоящий из макета и управляющего модуля к нему, который позволяет задавать маршрут при помощи выбора направления движения мобильной платформы на перекрестках.

Список используемой литературы

1. Блум Джереми. Изучаем Arduino: инструменты и методы технического волшебства: Пер. с англ. – СПб.: БВХ-Петербург, 2015. – 336 с.: ил. ISBN 978-5-9775-3585-4
2. Глибин Е.С. Программирование электронных устройств: электронное учебное пособие / Е.С. Глибин, А.В. Прядилов, - Тольятти: Издво ТГУ, 2014
3. Достойные аналоги Ардуино: Teensy, Netduino и другие [Электронный ресурс].URL: https://arduinoplus.ru/vse-analogi-arduino/ (дата обращения: 20.05.2021)
4. Изменение адреса дисплея по шине I2C [Электронный ресурс]. URL: https://flprog.ru/uchebnyj-centr/articles/podkljuchenie-displeev-i-indikatorov/izmenenie-adresa-displeja-po-shine-i2c/ (дата обращения: 16.05.2021)
5. Как работают светодиоды и их виды, полярность и расчет резистора [Электронный ресурс]. URL: https://arduinomaster.ru/datchiki-arduino/printsip-raboty-i-vidy-svetodiodov/ (дата обращения: 12.05.2021)
6. Подключение шилда LCD Keypad Shield 1602 к Arduino [Электронный ресурс]. URL: https://arduinomaster.ru/platy-arduino/arduino-lcd-keypad-shield/ (дата обращения: 22.05.2021)
7. Программирование на языке С для AVR и PIC микроконтроллеров./ Сост. Ю.А. Шпак – К.: «МК-Пресс», 2006. – 400 с., ил. ISBN 966-8806-16-6
8. Среда разработки Arduino IDE [Электронный ресурс]. URL: https://arduinoplus.ru/ (дата обращения: 11.04.2021)
9. Устройство и принцип работы Wi-Fi сети (преимущества и недостатки) [Электронный ресурс]. URL: /https://hobbyits.com/ustrojstvo-i-princip-raboty-wi-fi-seti/ (дата обращения: 11.05.2021)

10. Фоторезистор: устройство, принцип работы, характеристики [Электронный ресурс]. URL: https://samelectrik.ru/chto-takoe-fotorezistory.html (дата обращения: 11.04.2021)
11. Arduino IDE [Электронный ресурс]: «Download the Arduino IDE» URL: https://www.arduino.cc/en/main/software (дата обращения: 20.02.2021).
12. Bluetooth модуль HC-05 [Электронный ресурс].URL: https://3d-diy.ru/wiki/arduino-moduli/bluetooth-modul-hc-05/ (дата обращения: 22.05.2021)
13. Connecting 2 Arduinos by Bluetooth using a HC-05 and a HC-06: Pair, Bind, and Link [Электронный ресурс]. URL: http://www.martyncurrey.com/connecting-2-arduinos-by-bluetooth-using-a-hc-05-and-a-hc-06-pair-bind-and-link/ (дата обращения: 10.04.2021)
14. Explain Algorithm and Flowchart with Examples [Электронный ресурс]. URL: https://www.edrawsoft.com/explain-algorithm-flowchart.html (дата обращения: 17.05.2021)
15. GitHub - ssilver2007/LCD_1602_RUS: Arduino LCD 16x02 display RUSSIAN with NO CYRILLIC symbols set [Электронный ресурс]. URL: https://github.com/ssilver2007/LCD_1602_RUS (дата обращения: 15.05.2021)
16. GNL-5013GT, Светодиод зеленый 20° d=5мм 100-300мКд 565нМ (Green) [Электронный ресурс].URL: https://www.chipdip.ru/product/gnl-5013gt (дата обращения: 21.05.2021)
17. GPS System : Working, Types,Trackers & Its Applications [Электронный ресурс]. URL: https://www.elprocus.com/how-gps-system-works/#:~:text=The%20working%2Foperation%20of%20the,the%20receiver%20on%20the%20earth. (дата обращения: 13.05.2021)
18. I2C расширитель портов PCF8574 / Деталька / Сообщество EasyElectronics.ru [Электронный ресурс]. URL:

http://we.easyelectronics.ru/part/i2c-rasshiritel-portov-pcf8574.html (дата обращения: 22.05.2021)

19. SS441A Datasheet, PDF - https://www.alldatasheet.com/ [Электронный ресурс]. URL: https://www.alldatasheet.com/view.jsp?Searchword=Ss441a%20datasheet&gclid=CjwKCAjwhYOFBhBkEiwASF3KGYsNbmKEg5kfANlAWv-MdPPNRn7j0h3WJ371AsCUZC-4c_t1YAnrXBoCvP4QAvD_BwE (дата обращения: 30.04.2021)
20. SS441A датчик Холла однопол. цифровой 115G/20G sip3 [Электронный ресурс]. URL: https://www.voltmaster-samara.ru/catalog/datchiki_magnitnogo_polya/ss441a_datchik_kholla_odnopol_tsifrovoy_115g_20g_sip3/ (дата обращения: 22.05.2021)

Приложение А

Перечень элементов к схеме принципиальной электрической

Поз	Наименование	Кол	Примечание
A1	Блок питания B12-1000	1	
A2	Управляющий модуль	1	
A3	Макет мини-полигона	1	
	Микросхемы		
DD1, DD4	Модуль расширения PCF8574	2	
DD2	Микроконтроллер Arduino UNO R3	1	
DD3	LCD Keypad Shield	1	
U1	Bluetooth-модуль HC-05	1	
	Резисторы		
R1-R6, R8-R15	МЛТ-0,25-10 кОм±10%	14	
R7	МЛТ-0,25-4,7 кОм±10%	1	
	Светодиоды		
VD1-VD6	АЛ5013А	6	
SA1	Кнопка тактовая 1571627-1 (FSM103A)	1	

Изм	Лист	№ докум	Подп	Дата	21-110304 53/09 215 02 ПЭ3		
Разраб		Платонов А С			Мини-полигон Перечень элементов	Лит	Лист	Листов
Пров		Кудинов А К						
Н контр		Кудинов А К				ТГУ, ЭЛб-1702а		
Утв		Шевцов А А						

Приложение Б

Код программы мини-полигона

```
#define _LCD_TYPE 2     // Выбор типа подключения монитора

// Объявление кнопок
#define BUTTON_PIN 2    // Тактовая кнопка
#define BUTTON_NONE 0   // Значение для BUTTON_NONE
#define BUTTON_RIGHT 1  // Значение для BUTTON_RIGHT
#define BUTTON_UP 2     // Значение для BUTTON_UP
#define BUTTON_DOWN 3   // Значение для BUTTON_DOWN
#define BUTTON_LEFT 4   // Значение для BUTTON_LEFT
#define BUTTON_SELECT 5 // Значение для BUTTON_SELEC

// Инициализация библиотек
#include <Wire.h>
#include <LCD_1602_RUS_ALL.h>
#include <PCF8574.h>

// Создание объектов
PCF8574 Expander_1(0x21);            // Создаем объект и указываем адрес устройства 0x20
PCF8574 Expander_2(0x22);            // Создаем объект и указываем адрес устройства 0x21
LCD_1602_RUS lcd(8, 9, 4, 5, 6, 7 );  // Создаем объект и указываем подключение LCD Keypad Shield

// Объявление поворотов
int TURN_1 = 0; // 1 поворот
int TURN_2 = 0; // 2 поворот
int TURN_3 = 0; // 3 поворот
int TURN_4 = 0; // 4 поворот
int TURN_5 = 0; // 5 поворот
int TURN_6 = 0; // 6 поворот

// Инициализация глобальных переменных
int SELECTED_TURN = 1;               // Выбранный поворот
char ROTATION_VALUE_TEXT = 32;       // Вывод значение поворота
int ROTATION_VALUE;                  // Значение направления поворота
boolean TURN_SELECTION_MENU_GO = true; // Переменная для запуска меню выбора поворота
boolean ROTATION_VALUE_MENU_GO = true; // Переменная для запуска меню выбора направления поворота
boolean START_MENU_GO = true;        // Переменная для запуска меню старта
boolean START_GO = false;            // Переменная для запуска меню работы макета мини-полигона
```

Продолжение Приложения Б

```
int getPressedButton() // Инициализация переменной
{
  int buttonValue = analogRead(0); // Чтение значения с аналогового входа
  delay(180);                    // Защита от дребезга

  if (buttonValue < 50) {          // Если при нажатии кнопки значение меньше 50
    return BUTTON_RIGHT;           // Значит нажата кнопка BUTTON_RIGHT
  }
  else if (buttonValue < 150) {    // Если при нажатии кнопки значение меньше 150
    return BUTTON_UP;              // Значит нажата кнопка BUTTON_UP
  }
  else if (buttonValue < 300) {    // Если при нажатии кнопки значение меньше 300
    return BUTTON_DOWN;            // Значит нажата кнопка BUTTON_DOWN
  }
  else if (buttonValue < 500) {    // Если при нажатии кнопки значение меньше 500
    return BUTTON_LEFT;            // Значит нажата кнопка BUTTON_LEFT
  }
  else if (buttonValue < 800) {    // Если при нажатии кнопки значение меньше 800
    return BUTTON_SELECT;          // Значит нажата кнопка BUTTON_SELECT
  }
  return BUTTON_NONE;              // Иначе, нажата кнопка BUTTON_NONE
}

void setup()
{
  Serial.begin(9600);              // Запускаем последовательный порт и задаём скорость
обмена данными

  // Установление режимов пинов на расширителе 0 на вывод данных
  Expander_1.pinMode(P0, OUTPUT);
  Expander_1.pinMode(P1, OUTPUT);
  Expander_1.pinMode(P2, OUTPUT);
  Expander_1.pinMode(P3, OUTPUT);
  Expander_1.pinMode(P4, OUTPUT);
  Expander_1.pinMode(P5, OUTPUT);

  // Установление уровня пинов на расширителе 0 на высокий
  Expander_1.digitalWrite(P0, HIGH);
  Expander_1.digitalWrite(P1, HIGH);
  Expander_1.digitalWrite(P2, HIGH);
  Expander_1.digitalWrite(P3, HIGH);
  Expander_1.digitalWrite(P4, HIGH);
  Expander_1.digitalWrite(P5, HIGH);
```

```
  // Установление режимов пинов на расширителе 0 на принятие данных
  Expander_2.pinMode(P0, INPUT_PULLUP);
  Expander_2.pinMode(P1, INPUT_PULLUP);
  Expander_2.pinMode(P2, INPUT_PULLUP);
  Expander_2.pinMode(P3, INPUT_PULLUP);
  Expander_2.pinMode(P4, INPUT_PULLUP);
  Expander_2.pinMode(P5, INPUT_PULLUP);

  pinMode(BUTTON_PIN, INPUT);          // Установление режима пина для тактовой кнопки на принятие данных

  lcd.begin(16, 2);                    // Инициализация дисплея: 2 строки по 16 символов

  // Вывод автора и название устройства
  lcd.setCursor(0, 0);
  lcd.print("Платонов А.С.");
  lcd.setCursor(0, 1);
  lcd.print("Элб-1702а");
  delay(2000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Мини-полигон");
  lcd.setCursor(0, 1);
  lcd.print("Версия 0.2.1");
  delay(2000);
  lcd.clear();
}

// Функция ошибки кнопки вверх
void TURN_WRONG_UP()
{
  // Если значение больше 6, то вывести предупреждение
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Значение выше 6!");
  delay(2000);
  lcd.clear();
}

// Функция ошибки кнопки вниз
void TURN_WRONG_DOWN()
{
  // Если значение меньше 1, то вывести предупреждение
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
  lcd.print("Значение ниже 1!");
  delay(2000);
  lcd.clear();
}

// Функция обработки выбора поворота
void TURN_SELECT()
{
  // Вывести номер выбранного поворота
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Выбран поворот:");
  lcd.setCursor(0, 1);
  lcd.print(SELECTED_TURN, DEC);
  delay(1500);
  lcd.clear();
}

void TURN_SELECTION_MENU() // Функция меню выбора поворота
{
  // Проверка на запылённость значений
  if (TURN_1 != 0 && TURN_2 != 0 && TURN_3 != 0 && TURN_4 != 0 && TURN_5 != 0
&& TURN_6 != 0)
  {
    START_MENU(); // Запуск функции меню старта
  }
  else
  {
    while (TURN_SELECTION_MENU_GO == true) // Пока значение переменной запуска
меню выбора поворота верно
    {
      int BUTTON_TURN_SELECTION = getPressedButton(); // Получение нажатой кнопки
      switch (BUTTON_TURN_SELECTION)
      {
        case BUTTON_UP: // Если нажата кнопка вверх
          if (SELECTED_TURN  < 6) // Если значение меньше 6
          {
           SELECTED_TURN ++; // Увеличить значение переменной для выбора поворота на 1
          }
          else
          {
            TURN_WRONG_UP(); // Вызов функции ошибки кнопки вверх
          }
          break;
```

```
  case BUTTON_DOWN: // Если нажата кнопка вниз
        if (SELECTED_TURN  > 1) // Если значение больше 1
        {
         SELECTED_TURN --; // Уменьшить значение переменной для выбора поворота на 1
        }
        else
        {
          TURN_WRONG_DOWN(); // Вызов функции ошибки кнопки вниз
        }
        break;
       case BUTTON_SELECT: // Если нажата кнопка выбора
        TURN_SELECT(); // Вызов функции обработки выбора
        BUTTON_TURN_SELECTION = 0; // Обнуление значения переменной поворота
        TURN_SELECTION_MENU_GO = false; // Выключение функции показа меню
выбора поворота
        break;
       default:
        // Вывести на экран текст ожидания
        lcd.setCursor(0, 0);
        lcd.print("Выберите поворот:");
        lcd.setCursor(0, 1);
        lcd.print(SELECTED_TURN, DEC);
     }
   }
 }
 ROTATION_VALUE_MENU(); // Запустить меню выбора значения поворота
}

// Функция обработки выбора направления поворота
void ROTATION_SELECT()
{
 // Вывести выбранную сторону поворота
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Выбрана сторона:");
 lcd.setCursor(0, 1);
 lcd.write(ROTATION_VALUE_TEXT);
 delay(1500);
 lcd.clear();
}

void ROTATION_VALUE_MENU() // Меню выбора значения поворота
{
   while (ROTATION_VALUE_MENU_GO == true) // Пока значение переменной запуска
меню выбора направления поворота верно
```

Продолжение Приложения Б

```
 {
   int BUTTON_ROTATION_VALUE_MENU = getPressedButton(); //получение нажатой кнопки

   switch (BUTTON_ROTATION_VALUE_MENU)
   {
    case BUTTON_DOWN: // Если нажата кнопка вверх
      ROTATION_VALUE_TEXT = 95; // Присвоить переменной для вывода значения поворота символ "_"
      ROTATION_VALUE = 4; // Присвоить переменной значения поворота число 4
      break;
    case BUTTON_UP: // Если нажата кнопка вниз
      ROTATION_VALUE_TEXT = 94; // Присвоить переменной для вывода значения поворота символ "^"
      ROTATION_VALUE = 1; // Присвоить переменной значения поворота число 1
      break;
    case BUTTON_RIGHT: // Если нажата кнопка вправо
      ROTATION_VALUE_TEXT = 62; // Присвоить переменной для вывода значения поворота символ ">"
      ROTATION_VALUE = 2; // Присвоить переменной значения поворота число 2
      break;
    case BUTTON_LEFT: // Если нажата кнопка влево
      ROTATION_VALUE_TEXT = 60; // Присвоить переменной для вывода значения поворота символ "<"
      ROTATION_VALUE = 3; // Присвоить переменной значения поворота число 3
      break;
    case BUTTON_SELECT: // Если нажата кнопка выбора
      ROTATION_SELECT(); // Вызов функции обработки выбора направления поворота
      BUTTON_ROTATION_VALUE_MENU = 0; // Обнуление значения переменной
      ROTATION_VALUE_MENU_GO = false; // Выключение функции показа меню выбора направления поворота
      break;
    default:
      // Вывести на экран текст ожидания
      lcd.setCursor(0, 0);
      lcd.print("Сторона поворота:");
      lcd.setCursor(0, 1);
      lcd.write(ROTATION_VALUE_TEXT);
   }
  }
  TURN_SELECT(ROTATION_VALUE); // Запустить обработку введенных данных и передать значение переменной значения поворота
}
```

Продолжение Приложения Б

```
void TURN_SELECT(int VALUE) // Обработка введённых данных
{
 switch (SELECTED_TURN)
 {
  case 1: TURN_1 = VALUE; break; // Если полученное значение равно 1, то присвоить значение переменной VALUE переменной 1 поворота
  case 2: TURN_2 = VALUE; break; // Если полученное значение равно 2, то присвоить значение переменной VALUE переменной 2 поворота
  case 3: TURN_3 = VALUE; break; // Если полученное значение равно 3, то присвоить значение переменной VALUE переменной 3 поворота
  case 4: TURN_4 = VALUE; break; // Если полученное значение равно 4, то присвоить значение переменной VALUE переменной 4 поворота
  case 5: TURN_5 = VALUE; break; // Если полученное значение равно 5, то присвоить значение переменной VALUE переменной 5 поворота
  case 6: TURN_6 = VALUE; break; // Если полученное значение равно 6, то присвоить значение переменной VALUE переменной 6 поворота
 }

 ROTATION_VALUE_MENU_GO = true; // Включение функции показа меню выбора направления поворота
 TURN_SELECTION_MENU_GO = true; // Включение функции показа меню выбора поворота
 TURN_SELECTION_MENU(); // Запустить функцию меню выбора поворота
}

void START_MENU() // Меню запуска мини-полигона
{
 // Вывод значений выбранных поворотов для тестирования
 lcd.setCursor(0, 0);
 lcd.print("Выбраны стороны:");
 lcd.setCursor(0, 1);
 lcd.print(TURN_1, DEC);
 lcd.setCursor(2, 1);
 lcd.print(TURN_2, DEC);
 lcd.setCursor(4, 1);
 lcd.print(TURN_3, DEC);
 lcd.setCursor(6, 1);
 lcd.print(TURN_4, DEC);
 lcd.setCursor(8, 1);
 lcd.print(TURN_5, DEC);
 lcd.setCursor(10, 1);
 lcd.print(TURN_6, DEC);
 delay(10000);
 lcd.clear();
```

Продолжение Приложения Б

```
 while (START_MENU_GO == true) // Пока значение переменной запуска меню старта верно
  {
    int ButtonState = 0; // Переменная состояния кнопки
    ButtonState = digitalRead(BUTTON_PIN); // Считываем состояние кнопки с цифрового пина и присваем значение переменной
    switch (ButtonState)
    {
      case HIGH: // Если состояние HIGH
        // Вывести сообщение об запуске
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Мини-полигон");
        lcd.setCursor(0, 1);
        lcd.print("запущен!");
        delay(2500);
        lcd.clear();
        START_GO = true; // Включение функции показа меню работы мини-полигона
        START_MENU_GO = false; // Выключение функции показа меню запуска мини-полигона
        break;
      default:
        // Вывести на экран текст ожидания
        lcd.setCursor(0, 0);
        lcd.print("Запуск полигона: ");
        lcd.setCursor(0, 1);
        lcd.print("СТАРТ");
    }
  }
  START(); // Запустить функцию работы мини-полигона
}

// Функция обработки данных от датчиков
void SENSOR_HALL(uint8_t SEN_0, uint8_t SEN_1, uint8_t SEN_2, uint8_t SEN_3, uint8_t SEN_4, uint8_t SEN_5)
{
  if (SEN_0 == 0) // Если сработал 1 датчик Холла
  {
    Expander_1.digitalWrite(P0, LOW); // Зажечь 1 светодиод
    Serial.print(TURN_1); // Отправить направление 1 поворота на мобильную платформу
    delay(15000);
    Expander_1.digitalWrite(P0, HIGH); // Выключить 1 светодиод
  }
  else if (SEN_1 == 0) // Если сработал 2 датчик Холла
  {
```

```
  Expander_1.digitalWrite(P1, LOW); // Зажечь 2 светодиод
  Serial.print(TURN_2); // Отправить направление 2 поворота на мобильную платформу
  delay(15000);
  Expander_1.digitalWrite(P1, HIGH); // Выключить 2 светодиод
 }
 else if (SEN_2 == 0) // Если сработал 3 датчик Холла
 {
  Expander_1.digitalWrite(P2, LOW); // Зажечь 3 светодиод
  Serial.print(TURN_3); // Отправить направление 3 поворота на мобильную платформу
  delay(15000);
  Expander_1.digitalWrite(P2, HIGH); // Выключить 3 светодиод
 }
 else if (SEN_3 == 0) // Если сработал 4 датчик Холла
 {
  Expander_1.digitalWrite(P3, LOW); // Зажечь 4 светодиод
  Serial.print(TURN_4); // Отправить направление 4 поворота на мобильную платформу
  delay(15000);
  Expander_1.digitalWrite(P3, HIGH); // Выключить 4 светодиод
 }
 else if (SEN_4 == 0) // Если сработал 5 датчик Холла
 {
  Expander_1.digitalWrite(P4, LOW); // Зажечь 5 светодиод
  Serial.print(TURN_5); // Отправить направление 5 поворота на мобильную платформу
  delay(15000);
  Expander_1.digitalWrite(P4, HIGH); // Выключить 5 светодиод
 }

 else if (SEN_5 == 0) // Если сработал 6 датчик Холла
 {
  Expander_1.digitalWrite(P5, LOW); // Зажечь 6 светодиод
  Serial.println(TURN_6); // Отправить направление 6 поворота на мобильную платформу
  delay(15000);
  Expander_1.digitalWrite(P5, HIGH); // Выключить 6 светодиод
 }
}

// Функция остановки меню старта
void STOP_START()
{
 Serial.print(4); // Отправить команду остановки на мобильную платформу
 // Вывести сообщение об прекращении работы мини-полигона
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Мини-полигон");
 lcd.setCursor(0, 1);
```

Продолжение Приложения Б

```
  lcd.print("остановлен!");
  delay(2500);
  lcd.clear();
  TURN_1 = TURN_2 = TURN_3 = TURN_4 = TURN_5 = TURN_6 = 0; // Обнулить
значения у всех поворотов
  SELECTED_TURN = 1; // Присвоить переменной выбранного поворота значение 1
  START_MENU_GO == true; // Включение функции показа меню запуска мини-полигона
  START_GO = false; // Выключение функции показа меню работы мини-полигона
}

void START() // Меню работы мини-полигона
{
  Serial.print(0); // Отправить команду запуска на мобильную платформу
  while (START_GO == true) // Пока значение переменной запуска меню работы макета
мини-полигона верно
  {
    int ButtonState = 0; // Переменная состояния кнопки
    ButtonState = digitalRead(BUTTON_PIN); // Считываем состояние кнопки с цифрового
пина и присваиваем значение переменной

    // Считываем состояние пинов на расширителе и присваиваем значение переменным
    uint8_t val0x22 = Expander_2.digitalRead(P0);
    uint8_t val1x22 = Expander_2.digitalRead(P1);
    uint8_t val2x22 = Expander_2.digitalRead(P2);
    uint8_t val3x22 = Expander_2.digitalRead(P3);
    uint8_t val4x22 = Expander_2.digitalRead(P4);
    uint8_t val5x22 = Expander_2.digitalRead(P5);

    // Вывести сообщение о работе мини-полигона
    lcd.setCursor(0, 0);
    lcd.print("Работаем...");

    SENSOR_HALL(val0x22, val1x22, val2x22, val3x22, val4x22, val5x22); // Запустить
функцию обработки данных от датчиков и передать значения

    if (ButtonState == HIGH)  { // Если состояние тактовой кнопки HIGH
      STOP_START(); // Запустить функцию остановки меню старта
      TURN_SELECTION_MENU(); // Запустить функцию меню выбора поворота
    }
  }
}

void loop(){
  TURN_SELECTION_MENU(); // Запустить функцию меню выбора поворота
}
```

Приложения В

Спецификация к сборочному чертежу управляющего модуля

Формат	Зона	Поз	Обозначение	Наименование	Кол	Приме-чание
				Документация		
А1			21-110304.53/09.215.04 СБ	Управляющий модуль. Сборочный чертеж		
				Сборочные единицы		
А2		1	21-110304.53/09.215.04.01 ГБ	Корпус Габаритный чертеж	1	
				Детали		
А4		2	21-110304.53/09.215.04.02 СБ	Плата светодиодная	1	
А4		3	21-110304.53/09.215.04.03 СБ	Плата кнопочная	1	
А4		4	21-110304.53/09.215.04.04 СБ	Плата расширения	1	
				Стандартные изделия		
		5		LCD Keypad Shield	1	
		6		Arduino UNO R3	1	
		7		Модуль расширения PCF8574	1	
		8		Разъем DB9	1	
		9		HC-05	1	

Перв примен | Справ № | Подп и дата | Инв № дубл | Взам инв № | Подп и дата | Инв № подл

Изм	Лист	№ докум	Подп	Дата	21-110304.53/09.215.04		
Разраб		Платонов А.С.			Управляющий модуль	Лит	Лист	Листов
Пров		Кудинов А.К.						
Н контр		Кудинов А.К.			Спецификация	ТГУ, ЭЛб-1702а		
Утв		Шевцов А.А.						

Приложения Г

Спецификация к сборочному чертежу платы светодиодной

Формат	Зона	Поз	Обозначение	Наименование	Кол	Приме-чание
				Документация		
			21-110304.53/09.215.04.02 СБ	Плата светодиодная		
				Сборочный чертеж		
				Детали		
		1		ПЛАТА МАКЕТНАЯ PCB (шаг 2.54мм)	1	
				Стандартные изделия		
		3		Светодиоды		
		2		AL1503A	6	

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Платонов А.С.		
Пров.		Кудинов А.К.		
Н. контр.		Кудинов А.К.		
Утв.		Шевцов А.А.		

21-110304.53/09.215.04.02

Плата светодиодная

Спецификация

Лит.	Лист	Листов
		1

ТГУ, ЭЛб-1702а

Приложения Д

Спецификация к сборочному чертежу платы кнопочной

Формат	Зона	Поз	Обозначение	Наименование	Кол	Приме-чание
				Документация		
А4			21-110304.53/09.215.04.03 СБ	Плата кнопочная.		
				Сборочный чертеж		
				Детали		
		1		ПЛАТА МАКЕТНАЯ PCB (шаг 2.54мм)	1	
				Стандартные изделия		
		2		Кнопка тактовая 1571627-1 (FSM103A)	1	
				Ристоры		
		3		МЛТ-0,25-10 кОм±10%	1	

Перв. примен.

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм	Лист	№ докум.	Подп.	Дата
Разраб.		Платонов А.С.		
Пров.		Кудинов А.К.		
Н. контр.		Кудинов А.К.		
Утв.		Шевцов А.А.		

21-110304.53/09.215.04.03

Плата кнопочная

Спецификация

Лит.	Лист	Листов
		1

ТГУ, ЭЛб-1702а

Приложения Е

Спецификация к сборочному чертежу платы расширения

Формат	Зона	Поз	Обозначение	Наименование	Кол	Приме-чание
				Документация		
А4			21-110304.53/09.215.04.04 СБ	Плата расширения		
				Сборочный чертеж		
				Стандартные изделия		
		1		ПЛАТА МАКЕТНАЯ 20мм х 80мм PCB (шаг 2,54мм)	1	
		2		Штыри 2х40 (шаг 2,54мм)	27	
				Резисторы		
		3		МЛТ-0,25-10 кОм±10%	7	
		4		МЛТ-0,25-4,7 кОм±10%	1	

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата	21-110304.53/09.215.04.04		
Разраб.		Платонов А.С.			Плата расширения	Лит.	Лист	Листов
Пров.		Кудинов А.К.						1
Н. контр.		Кудинов А.К.			Спецификация	ТГУ, ЭЛб-1702а		
Утв.		Шевцов А.А.						