МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра механики, мехатроники и робототехники

УТВЕРЖДАЮ:
проректор по учебной работе
_____Локтионова О.Г.
_____2015г.

МИКРО- И НАНОДВИЖИТЕЛИ

Методические указания по выполнению
лабораторных работ
для студентов направления
«Нанотехнологии и микросистемная техника»

Курск

УДК 537.9: 537.62

Составители: П.А. Безмен, А.С. Яцун, Е.Н. Политов

Рецензент
Кандидат технических наук, доцент *А.Е. Кузько*

Микро- и нанодвижители: методические указания по выполнению лабораторных работ для студентов направления «Нанотехнологии и микросистемная техника»/ Юго-Зап. гос. ун-т; сост.: П.А. Безмен, А.С. Яцун, Е.Н. Политов. Курск, 2015. 23с., табл. 5. Библиогр.: с. 23.

Изложен план проведения лабораторных работ по дисциплине «Микро- и нанодвижители», методика формирования в рамках данной дисциплины профессиональных компетенций, студентов, обучающихся по направлению «Нанотехнологии и микросистемная техника», а также вопросы для самостоятельного рассмотрения.

Методические указания соответствуют требованиям программы, утверждённой учебно-методическим объединением (УМО).

Предназначены для студентов направления «Нанотехнологии и микросистемная техника» всех форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60x84 1\16
Усл.печ.л. .Уч.изд.л. .Тираж 50 экз.Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, г.Курск, ул.50 лет Октября, 94.

Содержание

Планируемые результаты обучения по дисциплине «Микро- и нанодвижители»

Целью изучения дисциплины «Микро- и нанодвижители» является формирование у студентов базовых знаний об эффектах и процессах, лежащих в основе функционирования микромеханических и микроэлектромеханических систем в элементах с микронными и нанометровыми размерами, способами управления их параметрами, приемами эксплуатации, при создании элементной базы микро- и наносистем.

Лабораторные занятия включают в себя:

а) теоретическую подготовку студента к занятию, в ходе которой студент обязан осмыслить теоретический материал, выносимый на занятие, и заучить основные законы и формулы;

б) выполнение лабораторной работы на занятии;

в) написание отчета по выполненной лабораторной работе;

г) защита лабораторной работы.

Каждый отчет должен быть подготовлен самостоятельно и соответствовать требованиям:

- отчет содержит титульный лист, описание выполняемого задания, описание проделанной работы, анализ полученных результатов, выводы, список использованной литературы;
- отчет выполняется на листах формата А4, 14 кегль, одинарный межстрочный интервал;
- список литературы оформляется согласно ГОСТ 7.1-2003.

1.ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ СЕРВОПРИВОДОМ С ПОМОЩЬЮ МИКРОКОНТРОЛЛЕРА

***Цель работы**: Ознакомление с принципами управления сервоприводом с помощью платы Arduino Uno.*
***Объект исследования:** сервопривод, плата Arduino Uno.*
***Аппаратные средства:** Персональный компьютер**,** Arduino IDE, набор Амперка Матрешка.*

1. Краткие теоретические сведения

Сервопривод — это привод с управлением через отрицательную обратную связь, позволяющую точно управлять параметрами движения. Сервоприводом является любой тип механического привода, имеющий в составе датчик (положения, скорости, усилия и т.п.) и блок управления приводом, автоматически поддерживающий необходимые параметры на датчике и устройстве согласно заданному внешнему значению.

Привод может быть подключен непосредственно к микроконтроллеру, без силового драйвера. Для этого от него идёт шлейф из трёх проводов:

1. красный — питание
2. коричневый — земля
3. жёлтый — сигнал; подключается к цифровому выходу микроконтроллера

Для подключения к Arduino будет удобно воспользоваться платой расширения Troyka Shield. Для управления из программы следует воспользоваться стандартной библиотекой Servo.

Принцип работы сервпривода:

1. Сервопривод получает на вход значение управляющего параметра. Например, угол поворота
2. Блок управления сравнивает это значение со значением на своём датчике
3. На основе результата сравнения привод производит некоторое действие, например: поворот, ускорение или замедление так, чтобы значение с внутреннего датчика стало как можно ближе к значению внешнего управляющего параметра

Рисунок 1.1 – Общий вид сервпоривода (рулевой машинки)

Библиотека Servo позволяет осуществлять программное управление сервоприводами. Для этого заводится переменная типа Servo. Управление осуществляется следующими функциями:
attach() — присоединяет переменную к конкретному пину. Возможны два варианта синтаксиса для этой функции: servo.attach(pin) и servo.attach(pin, min, max). При этом pin — номер пина, к которому присоединяют сервопривод, min и max — длины импульсов в микросекундах, отвечающих за углы поворота 0° и 180°. По умолчанию выставляются равными 544 мкс и 2400 мкс соответственно.
write() — отдаёт команду сервоприводу принять некоторое значение параметра. Синтаксис следующий: servo.write(angle), где angle — угол, на который должен повернуться сервопривод.
writeMicroseconds() — отдаёт команду послать на сервоприводимульс определённой длины, является низкоуровневым аналогом предыдущей команды. Синтаксис следующий:servo.writeMicroseconds(uS), где uS — длина импульса в микросекундах.
read() — читает текущее значение угла, в котором находится сервопривод. Синтаксис следующий: servo.read(), возвращается целое значение от 0 до 180.
attached() — проверка, была ли присоединена переменная к конкретному пину. Синтаксис следующий: servo.attached(), возвращается логическая истина, если переменная была присоединена к какому-либо пину, или ложь в обратном случае.
detach() — производит действие, обратное действию attach(), то есть отсоединяет переменную от пина, к которому она была приписана. Синтаксис следующий: servo.detach().

2. Выполнение лабораторной работы

Соберите элементы подключения сервопривода согласно схеме:

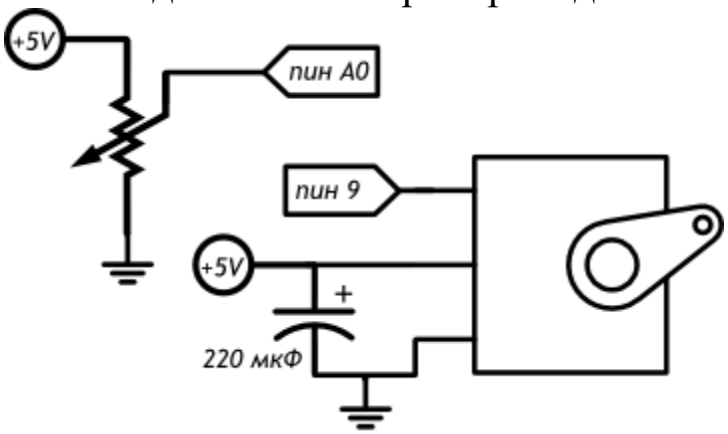

Рисунок 1.2 – Принципиальная схема подключения

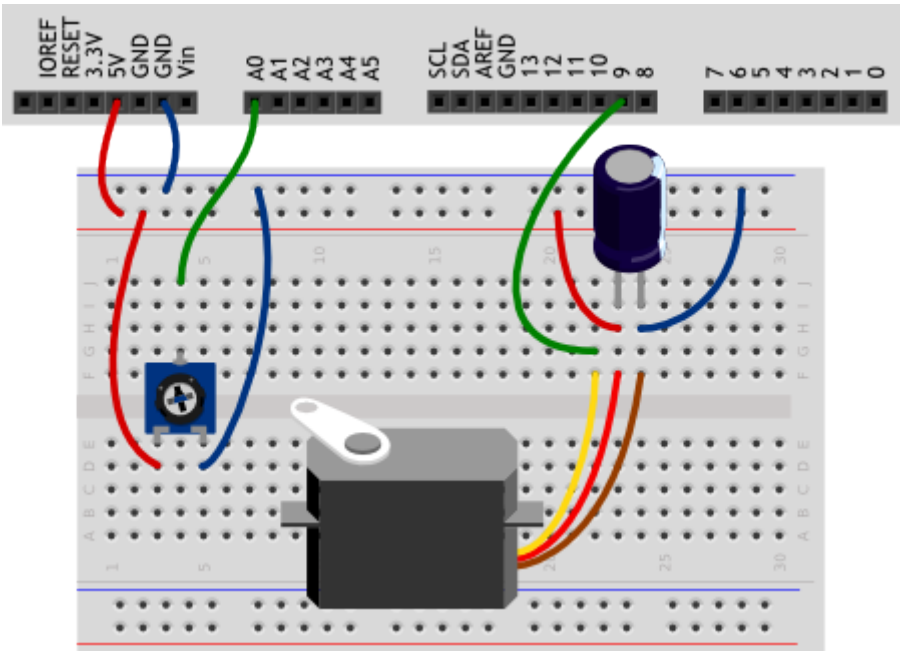

Рисунок 1.3 – Принципиальная схема подключения

После подключения элементов цифровой системы управления напишите код программы, согласно примеру.

```
// подключение стандартной библиотеки
#include <Servo.h>

#define POT_MAX_ANGLE 270.0 // макс. угол поворота потенциометра

// объявляем объект типа Servo с именем myServo. Ранее мы
```

```
// использовали int, boolean, float, а теперь точно также
// используем тип Servo, предоставляемый библиотекой. В случае
// Serial мы использовали объект сразу же: он уже был создан
// для нас, но в случае с Servo, мы должны сделать это явно.
// В проекте могут быть одновременно несколько
// приводов, и нам понадобится различать их по именам
Servo myServo;

void setup()
{
  // прикрепляем (англ. attach) нашу серву к 9-му пину. Явный
  // вызов pinMode не нужен: функция attach сделает всё за нас
  myServo.attach(9);
}

void loop()
{
  int val = analogRead(A0);
  // на основе сигнала понимаем реальный угол поворота движка.
  // Используем вещественные числа в расчётах, но полученный
  // результат округляем обратно до целого числа
  int angle = int(val / 1024.0 * POT_MAX_ANGLE);
  // обычная серва не сможет повторить угол потенциометра на
  // всём диапазоне углов. Она умеет вставать в углы от 0° до
  // 180°. Ограничиваем угол соответствующе
  angle = constrain(angle, 0, 180);
  // и, наконец, подаём сервер команду встать в указанный угол
  myServo.write(angle);
}
```

- Как отмечено в комментариях, в отличие от объекта `Serial`, объекты типа`Servo` нам нужно явно создать: `Servo myServo`, предварительно подключив библиотеку `<Servo.h>`.
- Далее мы используем два метода для работы с ним:
- `myServo.attach(pin)` — сначала «подключаем» сервопривод к порту, с которым физически соединен его сигнальный провод. `pinMode()` не нужна, метод `attach()` займется этим.
- `myServo.write(angle)` — задаем угол, т.е. позицию, которую должен принять вал сервопривода. Обычно это 0—180°.

- myServo здесь это имя объекта, идентификатор, который мы придумываем так же, как названия переменных. Например, если вы хотите управлять двумя захватами, у вас могут быть объектыleftGrip и rightGrip.
- Мы использовали функцию int() для явного преобразования числа с плавающей точкой в целочисленное значение. Она принимает в качестве параметра значение любого типа, а возвращает целое число. Когда в одном выражении мы имеем дело с различными типами данных, нужно позаботиться о том, чтобы не получить непредсказуемый ошибочный результат.

4. Контрольные вопросы

1. Зачем нужен конденсатор при включении в схему сервопривода?
2. Каким образом библиотека <Servo.h> позволяет нам работать с сервоприводом?
3. Зачем мы ограничиваем область допустимых значений для angle?
4. Как быть уверенным в том, что в переменную типа int после вычислений попадет корректное значение?

2. ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА УПРАВЛЕНИЯ СЕРВОПРИВОДОМ SMC

Цель работы: *изучение линейного электропривода SMC и реализация протокола Modbus, используя интерфейс RS-485*
Объект исследования: *набор электроприводов SMC, стандарт RS-485.*
Аппаратные средства: *электропривод и микроконтроллер SMC.*

1. Краткие теоретические сведения

При использовании RS-485 стандарт Modbus определяет правила подключения устройств по 2-х проводной и 4-х проводной схеме, а также правила совместимости 2-х проводных и 4-х проводных интерфейсов на единственной линии. Ниже рассмотрено 2-х проводное подключение, поддержка которого является обязательной.

По сути, 2-х проводное подключение на самом деле является 3-х проводным, так как кроме линий A и B используется также общий провод («земля») – рисунок 2.

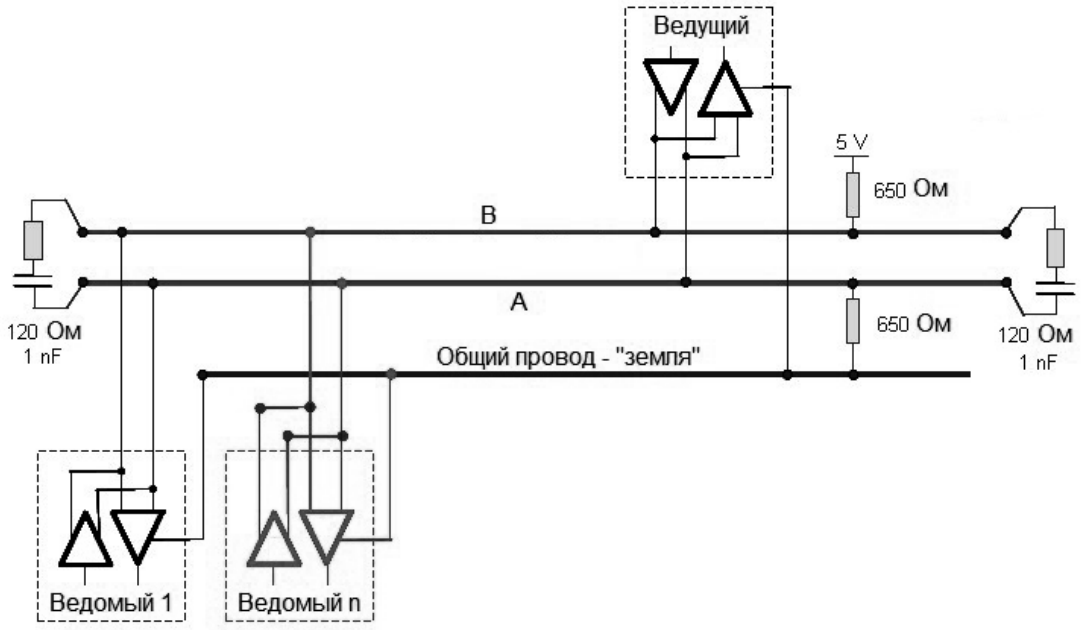

Рисунок 2.1 Схема подключения устройств к шине Modbus, используя интерфейс RS-485

Общее количество устройств ограничено: 32 устройства на одном сегменте RS-485 без репитеров (использование репитеров разрешается). Максимальная длина кабеля зависит от: скорости, типа кабеля, количества нагрузок и конфигурации сети (2-х

проводная или 4-х проводная). Для скорости 9600 бит/сек и кабеля AWG26 максимальная длина ограничена 1000 м. Кабель ответвления должен быть короче 20 м.

В стандарте Modbus определены правила реализации защитного смещения (поляризации), которые предусматривают подключение питания номиналом 5 В через резисторы для поддержания логической единицы на линии при отсутствии передачи (рисунок 2). Номинал резисторов выбирается от 450 Ом до 650 Ом в зависимости от количества устройств (650 Ом при большом количестве). Защитное смещение проводится только в одной точке линии, как правило, на стороне ведущего. Максимальное количество устройств с реализованной поляризацией уменьшается на 4 по сравнению с системой без поляризации. Поляризация является необязательной. Однако, коммуникации на устройствах могут давать сбой при отсутствии логического сигнала. Если это так, то поляризацию необходимо реализовывать самостоятельно, или использовать существующие схемы, если таковые предусмотрены устройствами.

На концах шины, между линиями A и B, выставляется терминаторы линии (LT). Терминаторы разрешается выставлять только на магистральном кабеле. В качестве терминаторов можно использовать:

- резистор номиналом 150 Ом и мощностью 0,5 Вт;
- последовательно соединенные конденсатор (1 нФ, 10 В минимум) и резистор номиналом 120 Ом (0,25 Вт) при использовании поляризации линии.

Выполнение лабораторной работы

Схема коммуникации контроллеров SMC LECP6N1 с ведущим контроллером, используя шину Modbus, приведена на рисунке 3. Линии интерфейса RS-485 шины подключаются к разъемам RJ-45 контроллеров SMC LECP6N1 (на корпусах контроллеров LECP6N1маркированы «CN4») как показано на рисунке 4.

Для управления каждым линейным приводом SMC LEY40C используются контроллеры SMC LECP6N1. Скорость передачи данных по шине Modbus – от 9600 бит/сек до 57600 бит/сек. В качестве ведущего контроллера может использоваться как микроконтроллер (например, Atmel Atmega 2560), так и

персональный компьютер, оснащенный портом с интерфейсом RS-232 (COM-порт) и конвертером интерфейсов RS-232-RS-485.

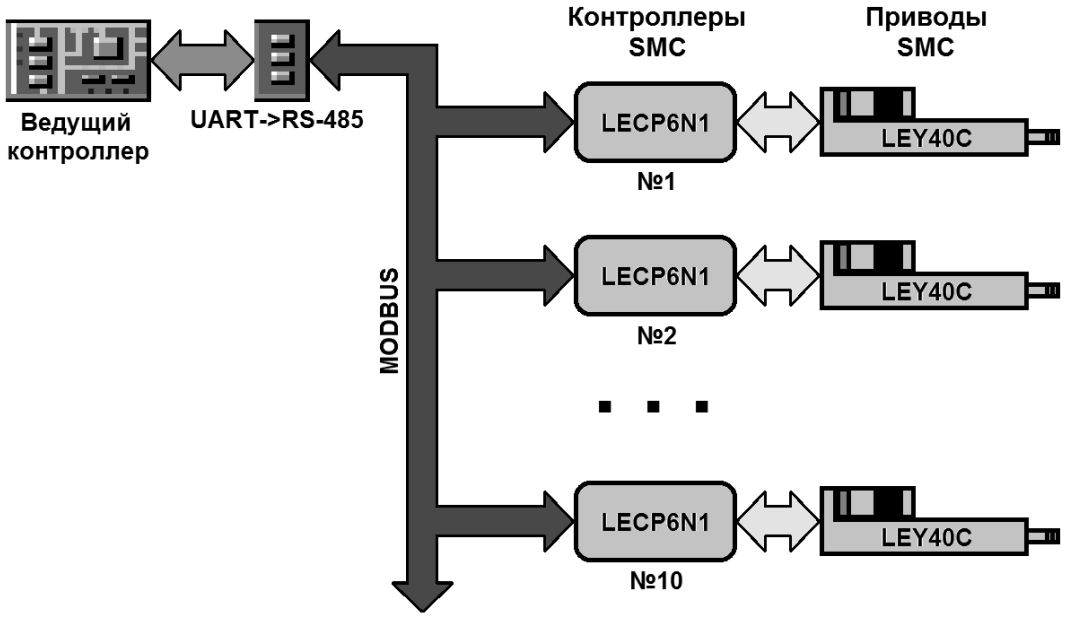

Рисунок 2.2 – Схема коммуникации контроллеров SMC LECP6N1 с ведущим контроллером

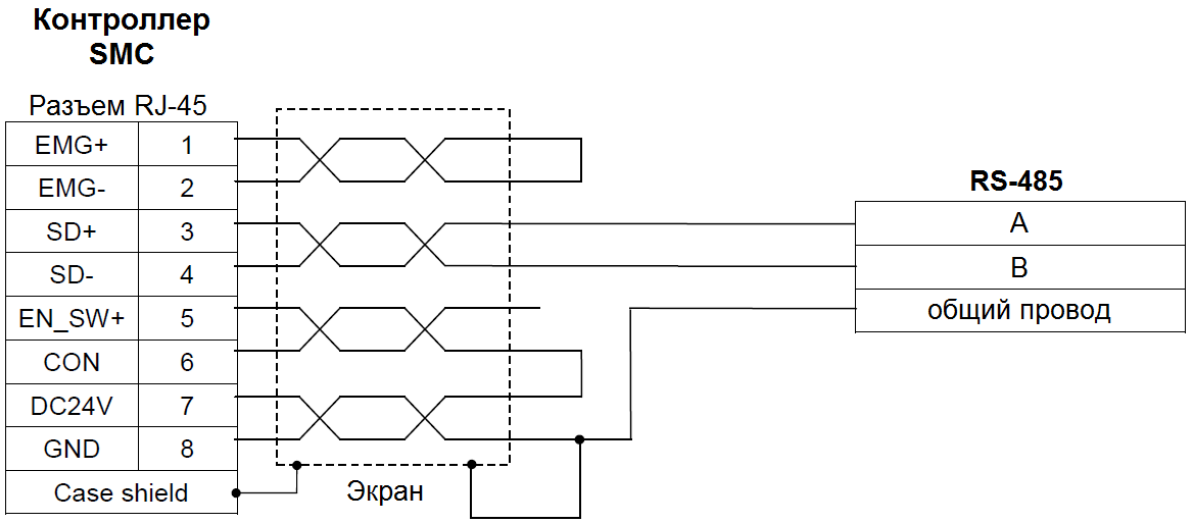

Рисунок 2.3 Подключение интерфейса RS-485 к разъему RJ-45 контроллера SMC LECP6N1

Работа приводов SMC обеспечивается отправкой управляющих команд от ведущего контроллера к ведомым устройствам - контроллерам SMC LECP6N1. Всего в сети Modbus присутствуют 10 контроллеров SMC – каждый контроллер имеет уникальный номер (ID): 1…10. Все контроллеры SMC предварительно настраиваются: всего для каждого контроллера могут быть установлены 64 позиции (точки) привода SMC LEY40C (от 0 до 63), при этом движение от позиции к позиции

регламентируется настройками требуемой абсолютной или относительной координаты, усилия, скорости, ускорения и т.д.

Для ведущего контроллера на языке программирования Си++ была разработана библиотека SMCLIB с набором функций, обеспечивающих:

- установку соединения с заданным контроллером SMC LECP6N1 (функция StartCommunication),
- включение заданного сервопривода (функция ServoON),
- выключение заданного сервопривода (функция ServoOFF),
- инициализацию (возврат штока) заданного сервопривода (функция ServoReturn),
- перемещение штока заданного сервопривода на заданную позицию (функция MoveToPoint),
- получение текущей координаты конца штока (в миллиметрах) заданного сервопривода (функция GetPosition),
- функция генерации контрольных сумм CRC для управляющих команд ведущего контроллера (функция CRC16).

В библиотеке SMCLIB приняты следующие обозначения аргументов функций:

- UARTNumber – номер порта UART: от 0 до 3,
- DeviceNumber – ID контроллера SMC: от 0 до 255,
- PointNumber – номер позиции привода SMC: от 0 до 63.

Последовательность вызова функций библиотеки SMCLIB:

```
StartCommunication(1, 2);          //установка соединения с
                                   //контроллером SMC с ID = 2
используя
                                   //порт UART №1
ServoON(1, 2);                     //включение сервопривода,
                                   //управляемого контроллером
SMC с ID = 2
ServoReturn(1, 2);                 //инициализация сервопривода,
                                   //управляемого контроллером
SMC с ID = 2
MoveToPoint(1, 2, 3);              //перемещение штока
сервопривода,
                                   //управляемого контроллером
SMC с ID = 2,
```

```
                                        //на позицию 3
double POS = GetPosition(1, 2); //получение текущей координаты
конца штока
                                        //сервопривода, управляемого
                                        //контроллером SMC с ID = 2
ServoOFF(1, 2);                 //выключение сервопривода,
                                        //управляемого контроллером
SMC с ID = 2
```

Подготовьте отчет о проведенной работе.

3 РЕАЛИЗАЦИЯ ОПТИМАЛЬНОЙ ТРАЕКТОРИИ ДВИЖЕНИЯ МОБИЛЬНЫМ КОЛЕСНЫМ РОБОТОМ

Цель работы: *изучить систему управления электроприводом постоянного тока на примере роботехнической платформы ShieldBot.*
Объект исследования: *роботехническая платформа ShieldBot*
Аппаратные средства: *Arduino IDE, набор Амперка Матрешка, .роботехническая платформа ShieldBot*

Робототехническая платформа ShieldBot, управляемая контроллером ARDUINO uno, имеет два коллекторных электропривода с редуктором, двухканальный драйвер двигателей, порты для подключения датчиков и микроконтроллера, аккумулятор. На платформе предусмотрены контакты в формфакторе Arduino для подключения стандартных плат.

На рисунке 1 представлена схема ShieldBot, содержащая основные элементы, такие как порты для подключения grove датчиков, инфракрасные датчики линии, моторы-редукторы, кнопки питания и перезагрузки, USB порт для зарядки батареи, IR Line Finder Sensor, Line Finder Switch.

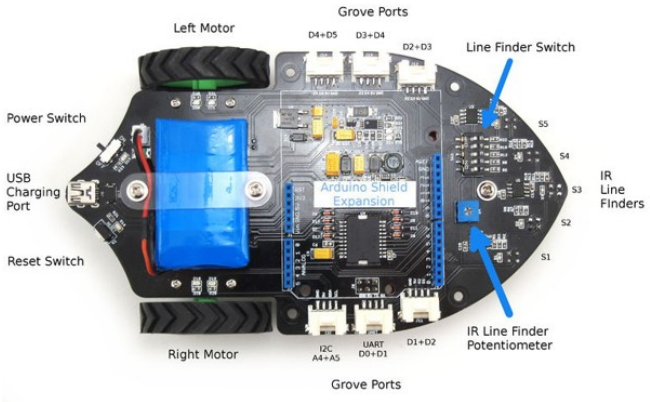

Рисунок 3.1 – Робототехническая платформа ShieldBot

Технические характеристики ShieldBot:
1. 6 разъёмов для датчиков Grove

2. 5 инфракрасных датчиков линии
3. 2 мотор-редуктора 160:1, шаровая опора
4. пластина из акрила для датчиков

6. USB mini-B разъём для зарядки аккумулятора.

На рисунке 2 представлена схема управления движением платформы с помощью встроенной функции.

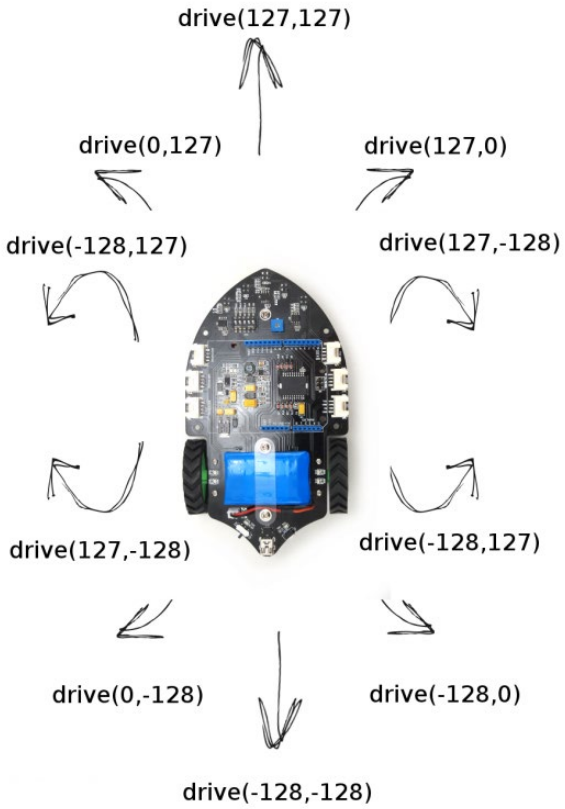

Рисунок 3.2 – Схема управления приводами

2. Выполнение работы.

Проверьте заряд аккумулятора, подготовьте алгоритм и программу управления согласно заданию, и загрузите ее в микроконтроллер управления роботом. Ниже представлен пример, реализующий движение по требуемой траектории:

```
#include <Shieldbot.h>
Shieldbot shieldbot = Shieldbot();
void setup() {
  shieldbot.setMaxSpeed(128);
}
void loop() {
  shieldbot.drive(50,60);
  delay(2000);
```

```
  shieldbot.drive(100,-120);
  delay(500);
  shieldbot.drive(50,60);
  delay(2000);
  shieldbot.drive(-100,120);
  delay(500);
}
```

3. Выполнение работы.

Задание на выполнение работы состоит в ознакомлении с конструкцией и системой управления робота, и реализации и загрузке программы управления. Необходимо продемонстрировать понимание аппаратно-программных принципов работы системы, показать, как осуществляется управление электроприводами при движении по траектории. Параметры следует выбирать из таблицы 1 в соответствии с рабочей группой.

№	Задание
Группа 1	*Реализовать программное движение по траектории типа "восьмерка"*
Группа 2	*Реализовать программное движение по траектории типа "квадрат"*
Группа 3	*Реализовать программное движение по траектории типа "кольцо"*
Группа 4	*Реализовать движение по линии контрастного цвета с помощью датчиков*