МИНОБРНАУКИ РОССИИ

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

Кафедра механики, мехатроники и робототехники

УТВЕРЖДАЮ
проректор по учебной работе
_____ О.Г. Локтионова
«__» января 2017 г.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО ЦИФРОВОМУ УПРАВЛЕНИЮ МЕХАТРОННЫМИ СИСТЕМАМИ

Методические указания к выполнению лабораторных работ
по дисциплине «Компьютерное управление мехатронными
системами» для студентов направления 15.03.06

Курск 2017

УДК 621

Составители: к.т.н. *Яцун А.С.*

Рецензент
Кандидат технических наук, доцент *Е.Н. Политов*

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО ЦИФРОВОМУ УПРАВЛЕНИЮ МЕХАТРОННЫМИ СИСТЕМАМИ: Методические указания к выполнению лабораторных работ по дисциплине «Компьютерное управление мехатронными системами» / Юго-Зап. гос. ун-т; сост. А.С. Яцун. Курск, 2017. 56 с.: ил. 45, табл. 6. Библиогр.: с._____.

Методические указания содержат лабораторные работы о методах проектирования и исследования цифровых систем управления, обработки сигналов, подключения исполнительных устройств и реализации цифровых регуляторов.
Методические указания соответствуют требованиям программы, утверждённой учебно-методическим объединением.
Предназначены для студентов направления подготовки 15.03.06 – Мехатроника и робототехника всех форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60x84 1/16 Усл.печ.л. ___. Уч.-изд.л. ___ Тираж 30 экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94

Оглавление

Общие требования к выполнению лабораторных работ

Изучение наиболее важных тем и разделов завершают лабораторные и практические занятия, которые обеспечивают закрепление учебного материала, развитие навыков работы с оборудованием, приобретение опыта устных выступлений, аргументации и защиты выдвигаемых положений и тезисов.

При выполнении работ у учащихся формируются следующие компетенции: ПК-1, ПК-2, ПК-5, ПК-6.

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. Лабораторная работа: дискретизация и квантование непрерывных сигналов

Цель работы: *изучение базовых принципов дискретизации и квантования сигналов, Теоремы Котельникова-Шеннона и эффекта поглощения частот.*
Объект исследования: *непрерывная и дискретная функции.*
Аппаратные средства: *математический пакет MathCAD или MATLAB\Octave.*

1. Краткие теоретические сведения

Входные и выходные сигналы непрерывных систем являются функциями непрерывного времени *t*. Если переменная времени принимает конечно множество значений, то сигнал называется дискретным. Дискретный сигнал называется также квантованным по времени. Цифровой сигнал - это дискретный сигнал, квантованный и по уровню и по времени.

Теорема Котельникова-Шеннона (англ. теорема Найквиста):
любую непрерывную функцию *F(t)*, состоящую из частот от 0 до *f*, можно непрерывно передавать с любой точностью при помощи чисел, следующих друг за другом через $1/(2f)$ секунд.
Такая трактовка рассматривает идеальный случай, когда сигнал начался бесконечно давно и никогда не закончится, а также не имеет во временной характеристике точек разрыва. Разумеется, реальные сигналы (например, информация на цифровом носителе) не обладают такими свойствами, так как они конечны по времени и обычно имеют разрывы во временной характеристике. Соответственно, ширина их спектра бесконечна. В таком случае полное восстановление сигнала невозможно, и из теоремы Котельникова вытекают следствия:

1. любой аналоговый сигнал может быть восстановлен с какой угодно точностью по своим дискретным отсчётам, взятым с частотой $f > 2f_c$, где f_c — максимальная частота, которая ограничена спектром реального сигнала;
2. если максимальная частота в сигнале равна или превышает половину частоты дискретизации (наложение спектра), то способа восстановить сигнал из дискретного в аналоговый без искажений не существует.

Преобразование Фурье заключается в представлении сигнала *Y(t)* в виде бесконечной суммы синусоид вида *F(v)*sin(v*x)*, где *v* - имеет смысл частоты, соответствующей гармоники сигнала; *F(v)* - преобразование Фурье или Фурье-спектр сигнала.

2. Выполнение работы

Рассмотрим вариант, где в качестве задания имеется непрерывный сигнал, заданный функцией вида:

$$y(t)=cos(0,05t)+sin(0,8t)$$

В программах математического моделирования построим график приведённой функции:

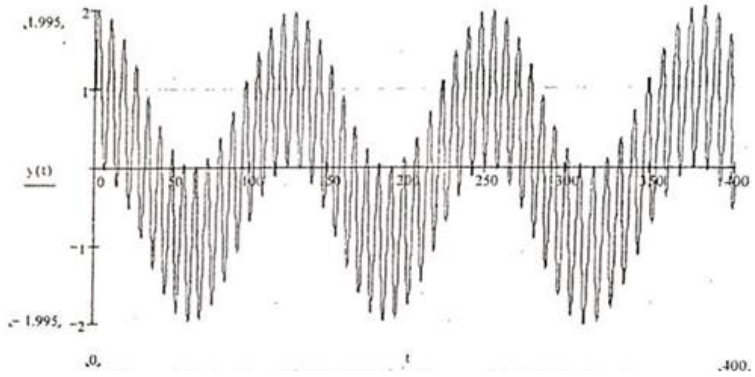

Рисунок 1.1 – График функции *y(t)=cos(0,05t)+sin(0,8t)*

Выберем время квантования сигнала из задания, в данном случае *T=5 сек*.

Полученные дискретные отсчеты соединим прямыми.

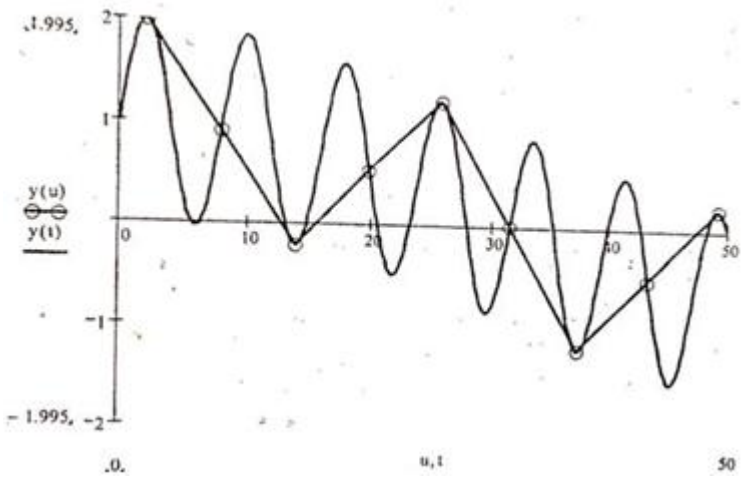

Рисунок 1.2 – Графики исходной функции и функции, полученной в результате квантования

Применение быстрого преобразования Фурье позволяет получить следующий спектр частот для заданной функции:

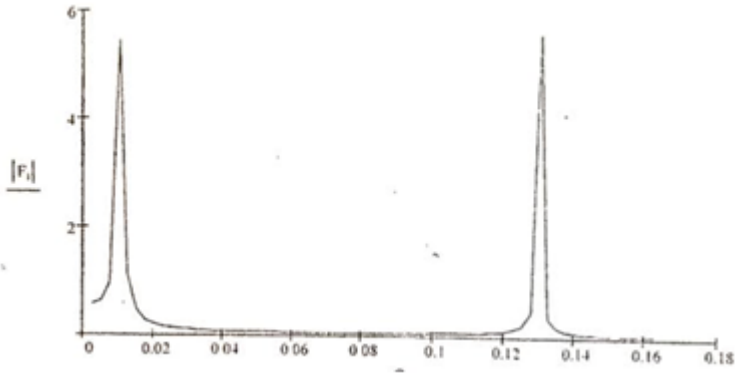

Рисунок 1.3 – Спектр функции *y(t)=cos(0,05t)+sin(0,8t)*

По следствию из теоремы Котельникова-Шеннона, для качественного отображения сигнала необходимо квантовать его с частотой, в 2 раза большей частоты спектра ω_o, в которой быстрое преобразование Фурье равно нулю.

Вычислим максимальную частоту, необходимую для корректной обработки сигнала. Т. к. в исходной функции *y(t)=cos(0,05t)+sin(0,8t)* наибольшую круговую частоту имеет слагаемое *sin(0,8t)*, то можно найти частоту f по формуле:

$$f = \frac{\omega}{2\pi}, f = \frac{0{,}8}{2\pi} = 0{,}12732$$

Как видно, максимальная частота совпадает с максимальной частотой спектра. Таким образом, по теореме Котельникова, частотой квантования необходимо взять частоту в 2 раза большую найденной.

Следовательно, период квантования будет равен:

$$T = \frac{1}{2f}, T = \frac{1}{2.0{,}12732} = 3{,}927\,c$$

Частота квантования этом будет равна:

$$f_s = \frac{1}{T} = \frac{1}{3{,}927} = 0{,}255\,Гц$$

Квантование должно начинаться не с нулевой секунды, а со времени, когда функция *sin(0,8t)* находится в максимуме, т.е. со времени *T/2*, или 1,963 с. Если же начать квантование с нуля, то в результате квантования мы получим нулевые отсчеты синусоиды и дискретный сигнал будет отличаться от исходного непрерывного.

В результате квантования получены следующие диаграммы:

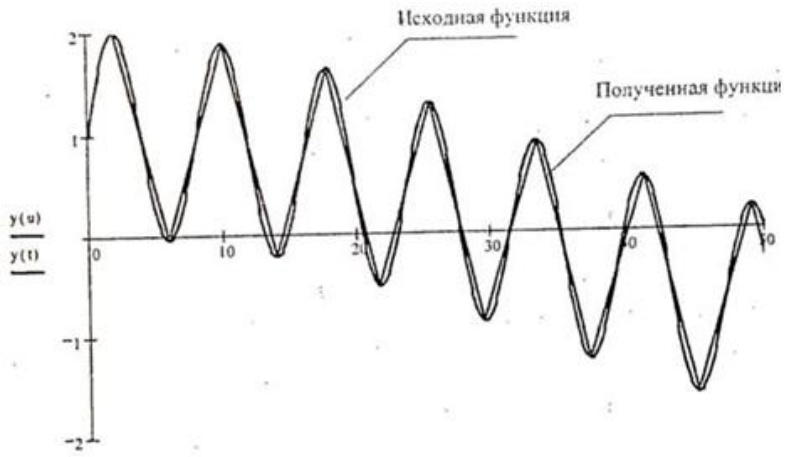

Рисунок 1.4 – Графики исходной функции и функции, полученной в результате квантования с частотой f_s

Если взять частоту квантования меньше, чем половина частоты спектра, то мы будем наблюдать так называемый эффект поглощения частот: частота квантования окажется недостаточной для качественной передачи сигнала

При увеличении частоты квантования качество получаемого: сигнала значительно увеличивается. Например, для частоты квантования $f = 4.f_s$ получим следующую диаграмму:

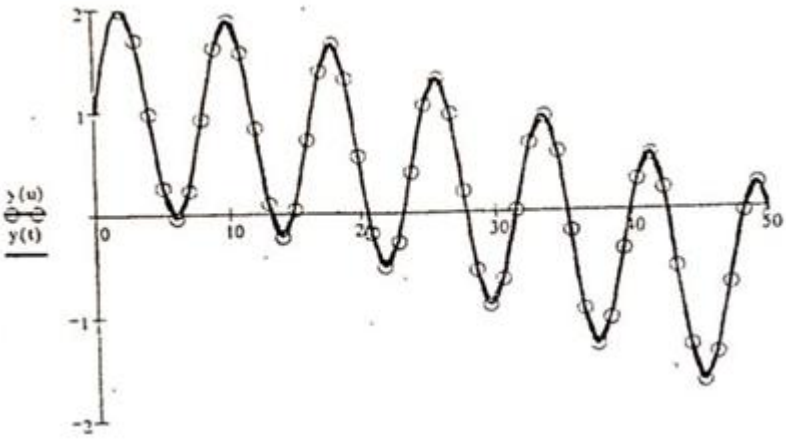

Рисунок 1.5 – Графики исходной функции и функции полученной в результате квантования с частотой $4f_s$

3. Задание на выполнение работы

Задание на выполнение работы состоит в написании программы в среде математического моделирования для нахождении частоты Найквиста для заданной функции, разложении в ряд Фурье, определении необходимого периода дискретизации.

Функции следует выбирать из таблицы 1 в соответствии с номером студента в списке группы.

Таблица 1 Задания на выполнение работы

№	Задание
1	*y(t)=cos(0,02t)+sin(0,4t), T=1 c*
2	*y(t)=sin(3t)-3*sin(0,45t), T=2 c*
3	*y(t)=sin(t)+cos(2t), T=0,5 c*
4	*y(t)=2*sin(0,1t)-cos(0,01t), T=5 c*
5	*y(t)=sin(0,45t)+2*cos(0,22t), T=0,33 c*
6	*y(t)=3*sin(2,3t)-cos(0,2t), T=0,5 c*
7	*y(t)=2*sin(5t)+sin(1,2t), T=2 c*
8	*y(t)=5*sin(0,03t)+cos(0,02t), T=5 c*
9	*y(t)=-sin(0,8t)+cos(0,25t), T=0,1 c*
10	*y(t)=sin(0,12t)-cos(0,43t), T=0,25 c*

2. Лабораторная работа: Программно-аппаратный комплекс для изучения цифровых систем управления Arduino IDE

***Цель работы**: Ознакомление с платой управления Arduino Uno и средой программирования Arduino IDE, построение принципиальные схемы на базе Arduino и программирование микроконтроллера.*
***Объект исследования:** плата Arduino Uno.*
***Аппаратные средства:** Персональный компьютер, Arduino IDE, набор Амперка Матрешка .*

1. Общие теоретические сведения

1.1 Плата Arduino Uno — это открытая платформа, которая позволяет собирать всевозможные электронные устройства. Устройства могут работать как автономно, так и в связке с компьютером. Платформа состоит из аппаратной и программной частей; обе чрезвычайно гибки и просты в использовании.

Рисунок 2.1 – – Общий вид Arduino Uno

Для программирования используется упрощённая версия C++, известная так же как Wiring. Разработку можно вести как с использованием бесплатной среды Arduino IDE, так и с помощью произвольного C/C++ инструментария. Для программирования и общения с компьютером понадобится USB-кабель. Для автономной работы потребуется блок питания на 7,5—12 В.

Arduino Uno - это устройство на основе микроконтроллера ATmega328. В его состав входит все необходимое для удобной работы с

микроконтроллером: 14 цифровых входов/выходов (из них 6 могут использоваться в качестве ШИМ-выходов), 6 аналоговых входов, кварцевый резонатор на 16 МГц, разъем USB, разъем питания, разъем для внутрисхемного программирования (ICSP) и кнопка сброса.

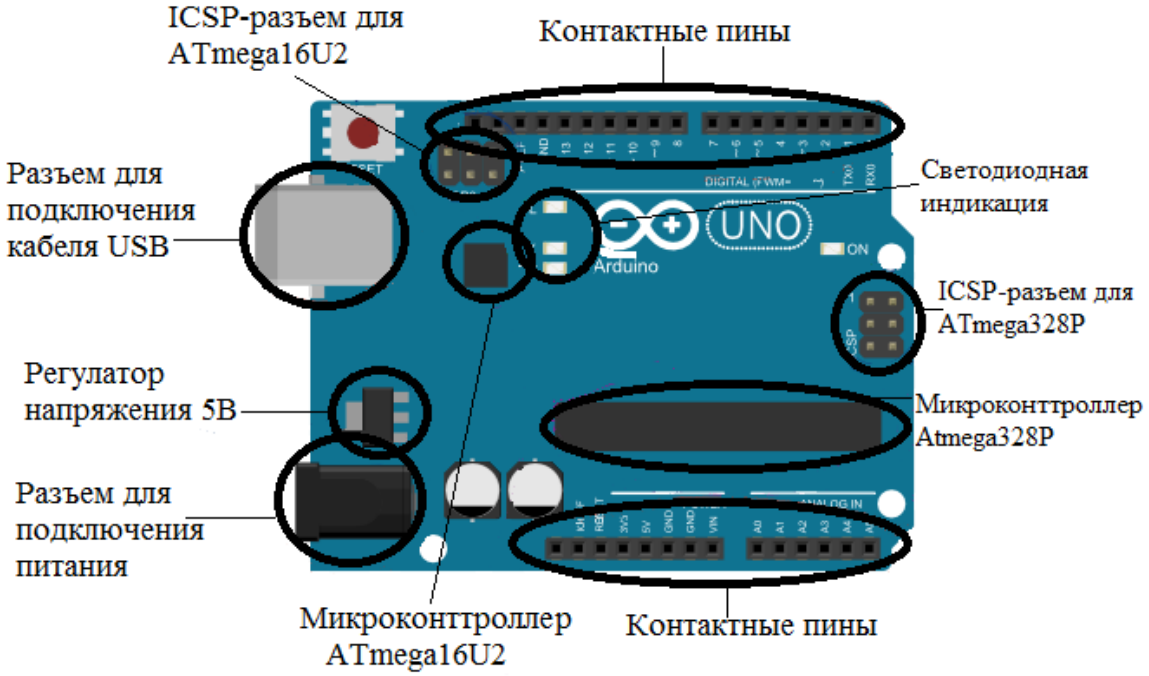

Рисунок 2.2 – – Основные компоненты Arduino Uno

Микроконтроллер ATmega328P: Основой платформы Arduino Uno является 8-битный микроконтроллер семейства AVR — ATmega328P.

Микроконтроллер ATmega16U2: Микроконтроллер ATmega16U2 обеспечивает связь микроконтроллера ATmega328P с USB-портом компьютера. При подключении к ПК Arduino Uno определяется как виртуальный COM-порт. Прошивка микросхемы 16U2 использует стандартные драйвера USB-COM, поэтому установка внешних драйверов не требуется.

Пины питания

- **VIN:** Напряжение от внешнего источника питания (не связано с 5 В от USB или другим стабилизированным напряжением). Через этот вывод можно как подавать внешнее питание, так и потреблять ток, если к устройству подключён внешний адаптер.
- **5V:** На вывод поступает напряжение 5 В от стабилизатора платы. Данный стабилизатор обеспечивает питание микроконтроллера ATmega328. Запитывать устройство через вывод 5V не рекомендуется — в этом случае не используется стабилизатор напряжения, что может привести к выходу платы из строя.
- **3.3V:** 3,3 В от стабилизатора платы. Максимальный ток вывода — 50 мА.

- **GND:** Выводы земли.
- **IOREF:** Вывод предоставляет платам расширения информацию о рабочем напряжении микроконтроллера. В зависимости от напряжения, плата расширения может переключиться на соответствующий источник питания либо задействовать преобразователи уровней, что позволит ей работать как с 5 В, так и с 3,3 В устройствами.

Порты ввода/вывода

- **Цифровые входы/выходы:** пины 0–13. Логический уровень единицы — 5 В, нуля — 0 В. Максимальный ток выхода — 40 мА. К контактам подключены подтягивающие резисторы, которые по умолчанию выключены, но могут быть включены программно.
- **ШИМ:** пины 3,5,6,9,10 и 11

Позволяют выводить 8-битные аналоговые значения в виде ШИМ-сигнала.

- **АЦП:** пины A0–A5

6 аналоговых входов, каждый из которых может представить аналоговое напряжение в виде 10-битного числа (1024 значений). Разрядность АЦП — 10 бит.

- **TWI/I²C:** пины SDA и SCL

Для общения с периферией по синхронному протоколу, через 2 провода. Для работы — используйте библиотеку Wire.

- **SPI:** пины 10(SS), 11(MOSI), 12(MISO), 13(SCK).

Через эти пины осуществляется связь по интерфейсу SPI. Для работы — используйте библиотеку SPI.

- **UART:** пины 0(RX) и 1(TX)

Эти выводы соединены с соответствующими выводами микроконтроллера ATmega16U2, выполняющей роль преобразователя USB-UART. Используется для коммуникации платы Arduino с компьютером или другими устройствами через класс Serial.

Разъём USB Type-B: Разъём USB Type-B предназначен для прошивки платформы Arduino Uno с помощью компьютера.

Разъём для внешнего питания: Разъём для подключения внешнего питания от 7 В до 12 В.

ICSP-разъём для ATmega328P: ICSP-разъём предназначен для внутрисхемного программирования микроконтроллера ATmega328P. С использованием библиотеки SPI данные выводы могут осуществлять связь с платами расширения по интерфейсу SPI. Линии SPI выведены на 6-контактный разъём, а также продублированы на цифровых пинах 10(SS), 11(MOSI), 12(MISO) и 13(SCK).

Характеристики:

- Тактовая частота: 16 МГц

- Напряжение логических уровней: 5 В
- Входное напряжение питания: 7–12 В
- Портов ввода-вывода общего назначения: 20
- Максимальный ток с пина ввода-вывода: 40 мА
- Максимальный выходной ток пина 3.3V: 50 мА
- Максимальный выходной ток пина 5V: 800 мА
- Портов с поддержкой ШИМ: 6
- Портов, подключённых к АЦП: 6
- Разрядность АЦП: 10 бит
- Flash-память: 32 КБ
- EEPROM-память: 1 КБ
- Оперативная память: 2 КБ
- Габариты: 69×53 мм

1.2. Редактор программ Arduino IDE

Редактор предназначен для написания, редактирования и загрузки прикладных программ в микроконтроллер. Редактор находится в свободном доступе на официальном сайте: http://arduino.cc/en/main/software

Меню редактора (1) включает в себя следующие главные элементы: файл, правка, скетч, сервис и справка. В пункте Файл можно найти команды, отвечающие за создание новой программы, чтение старой, сохранения её изменений, а также команды для загрузки программы на микроконтроллер.

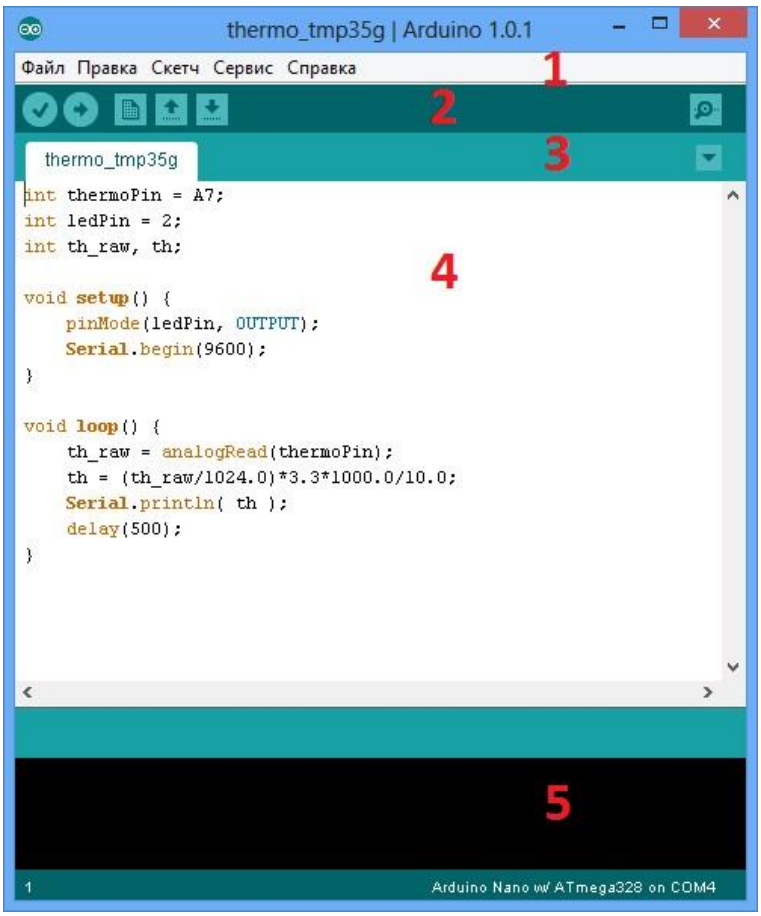

Рисунок 2.3 – – Окно редактора Arduino IDE

Пункт меню Правка содержит команды, связанные с редактирование текст программы, включая копирование, вставку, настройку отступов и поиск по ключевому слову. В разделе Скетч размещаются команды для управления компиляцией программы.

Каждая программа для Arduino может состоять из нескольких файлов. Для переключения между этими файлами служит система вкладок (3) в редакторе. Там же, можно создать новую вкладку, и ассоццировать с ней файл в папке с проектом.

Непосредственно текст программы создается и редактируется в главном окне редактора (4). По сути, окно редактора представляет собой типичный текстовый редактор, с подсветкой конструкций кода.

В самом низу редактора Arduino IDE имеется небольшое окно (5), служащее для вывода уведомлений об ошибках, возникающих в процессе компиляции программы, или во время загрузки программы в микроконтроллер.

1.3. Основы программирования Arduino

Программирование Arduino осуществляется на языке C++.

1. Каждое выражение заканчивается символом **;** точка с запятой. Например:

```
a = b+c;
```

2. Тело функций и составных операторов (if, else, for, while) обособляется фигурными скобками

3. Строки обособляются обычными двойными кавычками “.

```
println(“some text”);
```

4. Символы обособляются одинарными кавычками:

```
symbol = ‘a’;
```

5. Подключение библиотек осуществляется с помощью конструкции:

```
#include <math.h>
```

6. Комметарии в программе начинаются с символов // два прямых слеша.

```
// это моя программа
```

Объявление переменной в языке с++ осуществляется с помощью конструкции вида:

тип_переменной имя_переменной;

1. Целые числа

byte от 0 до 255

int от 32 768 до 32 767

word от 0 до 65535

long от 2 147 483 648 до 2 147 483 647

2. Дробные числа

float от 3.4028235E+38 до 3.4028235E+38

double эквивалетно float в текущей версии Arduino

3. Массивы

Массивы в с++ задаются конструкцией типа:

тип_элемента имя_массива[размер];

Пример:

```
int numbers[10]; // задает массив из десяти целых чисел
```

4. Строки и символы

char символ;

Строки в с++ представляют собой массивы с элементами типа **char**. Пример:

```
char my_str[10]; // строка из десяти символов
```

5. Прочие типы

void пустой тип;

boolean false либо true (ложь или истина)

Задание функции: *тип_функции имя_функции(аргументы) { команды, выполняемые в рамках функции return результат_функции;*
}

тип_функции тип возвращаемого значения. Например, стандартная функция **sin** имеет тип возвращаемого значения **float**.

имя_функции любая строка, начинающаяся с буквы, и содержащая только буквы и символы подчеркивания.

аргументы перечень аргументов, которые функция использует для своих действий.

результат_функции пременная или число, определяющее возвращаемое значение функции.

Минимальная программа, которая может запустить на Arduino состоит всего из двух функций:

```
void setup() {
}

void loop() {
}
```

Первая функция **setup** вызывается только один раз, после запуска Arduino. Обычно она используется для конфигурации портов микроконтроллера и других настроек – в ней инструкции для инициализации микроконтроллера.

После выполнения setup запускается процедура loop, которая выполняется в бесконечном цикле, она является основным программным циклом.

Процедуры setup и loop должны присутствовать в любой программе (скетче), даже если не нужно ничего выполнять в них — они будут пустые:

```
void setup()
```

2. Выполнение лабораторной работы

В работе необходимо подключить светодиод к выходу микроконтроллера и написать программу, которая будет включать и выключать светодиод на время согласно заданию. Для работы из набора Амперка Матрешка соберите следующую принципиальную схему с использованием беспаечной макетной платы. Пин для подключения светодиода и время включения необходимо выбрать из таблицы с заданием. При отсутствии физической платы допустимо использовать виртуальный комплекс моделирования с использованием ресурса www.tinkercad.com или аналога.

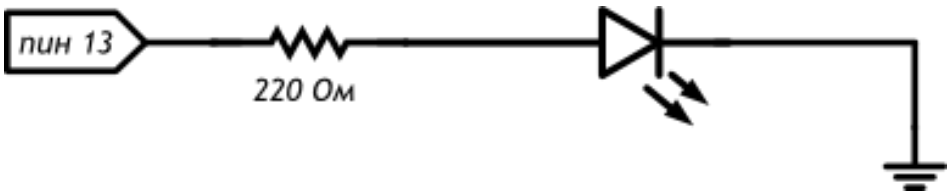

Рисунок 2.4 – Принципиальная схема

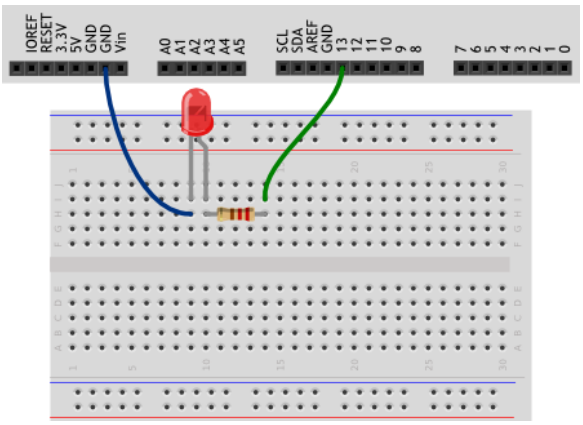

Рисунок 2.5 – Схема подключения

Далее напишите код программы с учетом следующей информации:
Функция digitalWrite(pin, value) не возвращает никакого значения и принимает два параметра:

- pin — номер цифрового порта, на который мы отправляем сигнал
- value — значение, которое мы отправляем на порт. Для цифровых портов значением может быть HIGH (высокое, единица) или LOW (низкое, ноль)

- Если в качестве второго параметра вы передадите функции digitalWrite значение, отличное от HIGH,LOW, 1 или 0, компилятор может не выдать ошибку, но считать, что передано HIGH.
- Обратите внимание, что использованные константы: INPUT, OUTPUT, LOW, HIGH, пишутся заглавными буквами, иначе компилятор их не распознает и выдаст ошибку. Когда ключевое слово распознано, оно подсвечивается синим цветом в Arduino IDE

Пример листинга программы с комментариями:

```
void setup()
{
  // настраиваем пин №13 в режим выхода,
  // т.е. в режим источника напряжения
  pinMode(13, OUTPUT);
}

void loop()
{
  // подаём на пин 13 «высокий сигнал» (англ. «high»), т.е.
```

```
  // выдаём 5 вольт. Через светодиод пойдет ток.
  digitalWrite(13, HIGH);

  // задерживаем (англ. «delay») микроконтроллер в этом
  // состоянии на 100 миллисекунд
  delay(100);

  // подаём на пин 13 «низкий сигнал» (англ. «low»), т.е.
  // выдаём 0 вольт или, точнее, приравниваем пин 13 к земле.
  // В результате светодиод погаснет
  digitalWrite(13, LOW);

  // замираем в этом состоянии на 900 миллисекунд
  delay(900);

  // после «размораживания» loop сразу же начнёт исполняться
  // вновь, и со стороны это будет выглядеть так, будто
  // светодиод мигает раз в 100 мс + 900 мс = 1000 мс = 1 сек
}
```

Далее необходимо загрузить данную программу в микроконтроллер и убедиться в правильности ее выполнения.

3. Задание на выполнение работы

Задание на выполнение работы состоит в реализации принципиальной схемы и загрузке программы управления. Необходимо продемонстрировать понимание аппаратно-программных принципов работы системы, показать, как осуществляется управление светодиодом. Параметры следует выбирать из таблицы 1 в соответствии с номером студента в списке группы.

Таблица 1 Задания на выполнение работы

№	Задание
1	*Пин 13, высокий уровень 100 мс, низкий уровень 900 мс*
2	*Пин 13, высокий уровень 500 мс, низкий уровень 500 мс*
3	*Пин 13, высокий уровень 800 мс, низкий уровень 200 мс*
4	*Пин 9, высокий уровень 2 с, низкий уровень 0,5 с*
5	*Пин 9, высокий уровень 1 с, низкий уровень 1,5 с*
6	*Пин 9, высокий уровень 400 мс, низкий уровень 1600 мс*
7	*Пин 3, высокий уровень 1500 мс, низкий уровень 500 мс*
8	*Пин 3, высокий уровень 200 мс, низкий уровень 800 мс*
9	*Пин 3, высокий уровень 100 мс, низкий уровень 100 мс*
10	*Пин 5, высокий уровень 300 мс, низкий уровень 200 мс*

3. Лабораторная работа: Исследование принципов аппаратной широтно-импульсной модуляции на основе микроконтроллера

***Цель работы**: Ознакомление с возможностями платы управления Arduino Uno и средой программирования Arduino IDE, реализация аналогового сигнала на базе ШИМ, подключение осциллографа.*

***Объект исследования:** плата Arduino Uno.*

***Аппаратные средства:** Персональный компьютер, Arduino IDE, набор Амперка Матрешка, цифровой осциллограф R&S RTE 1034 или аналог.*

1. Общие теоретические сведения

Обычно микроконтроллеры не могут формировать произвольный уровень напряжение на своих выводах, они выдают цифровой сигнал либо «1» - напряжение питания (например, 5 В или 3.3 В), либо «0» - землю (0 В). Но для цифрового управления очень часто требуется управление уровнем напряжения: например, для изменения яркости светодиода или управления скоростью вращения мотора. Для реализации аналогового уровня напряжения используется Широтно-Импульсная Модуляция, или сокращенно ШИМ (англ. Pulse Width Modulation или PWM) - процесс управления мощностью, подводимой к нагрузке, путём изменения скважности импульсов, при постоянной частоте и амплитуде.

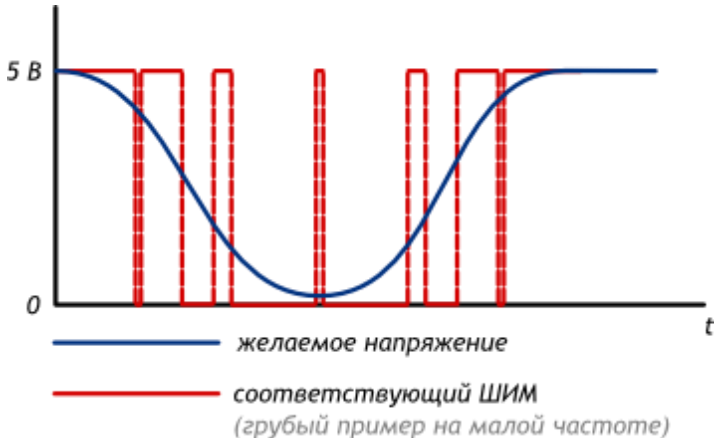

Рисунок 3.1 – График ШИМ-сигнала

При аппаратной реализации ШИМ-сигнала заданный выход микроконтроллера переключается между землёй и питанием тысячи раз в секунду. Глаз человека не замечает колебаний частотой более 70-80 Гц, поэтому кажется, что светодиод не мерцает, а горит в половину накала. Аналогично, разогнанный мотор не может остановить вал за миллисекунды, поэтому ШИМ-сигнал заставит вращаться его в неполную силу.

Отношение полного периода к времени включения называют скважностью ШИМ, обратная величина называется величиной (англ. duty cycle). Рассмотрим несколько сценариев при напряжении питания Vcc равным 5 вольтам.

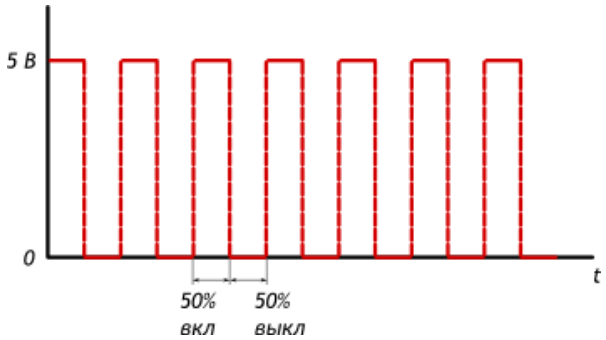

Рисунок 3.2 – Величина ШИМ 50%, эквивалент 2,5 В

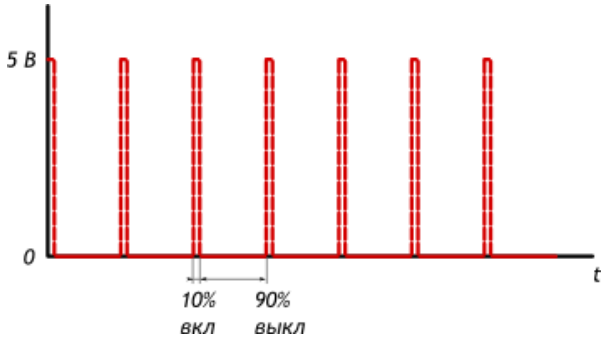

Рисунок 3.3 – Величина ШИМ 10%, эквивалент 0,5 В

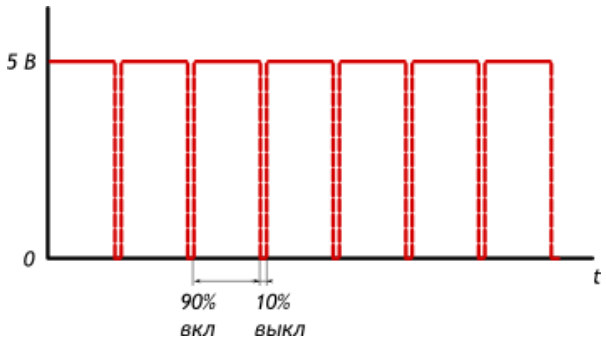

Рисунок 3.4 – Величина ШИМ 90%, эквивалент 4,5 В

Частота сигнала при аппаратном формировании ШИМ на платах Arduino составляет приблизительно 490 Гц. На Uno и подобных платах, пины 5 и 6 имеют частоту приблизительно 980 Hz. На большинстве плат с МК ATmega168 или ATmega328P функция ШИМ работает на пинах 3, 5, 6, 9, 10, и 11. Разрядность ШИМ на плате Arduino UNO равна 8 бит, то есть величина может принимать 256 значений.

Для выполнения работы необходимо изучить техническую документацию и функционал цифрового осциллографа ROHDE&SCHWARZ RTE 1034 (350 MHz, 5 GSa/s), представленного на рисунке.

Рисунок 3.5 – Осциллограф ROHDE&SCHWARZ RTE 1034

2. Выполнение лабораторной работы

В работе необходимо подключить светодиод к выходу ШИМ микроконтроллера (обозначен на плате тильдой ~) и написать программу, которая будет плавно изменять яркость. Для работы из набора Амперка Матрешка соберите следующую принципиальную схему с использованием беспаечной макетной платы. Режим включения необходимо выбрать из таблицы с заданием. При отсутствии физической платы допустимо использовать виртуальный комплекс моделирования с использованием ресурса www.tinkercad.com или аналога.

Соберем электрическую схему для проведения работы.

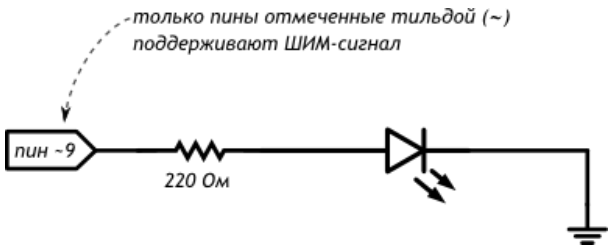

Рисунок 3.6 – Принципиальная схема

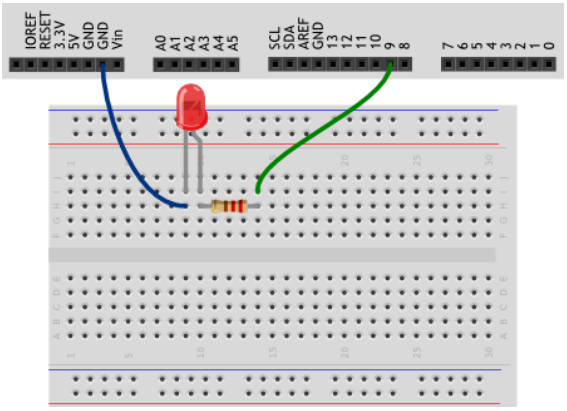

Рисунок 3.7 – Схема подключения

После подключения элементов цифровой системы управления напишите код программы, аналогично примеру. Обратите внимание, что Идентификаторы переменных, констант, функций (в этом примере идентификатор `LED_PIN`) являются одним словом (т.е. нельзя создать идентификатор `LED PIN`).

Идентификаторы могут состоять из латинских букв, цифр и символов подчеркивания `_`. При этом идентификатор не может начинаться с цифры.

Регистр букв в идентификаторе имеет значение. Т.е. `LED_PIN`, `LED_pin` и `led_pin` с точки зрения компилятора — различные идентификаторы

Директива `#define` говорит компилятору заменить все вхождения заданного идентификатора на значение, заданное после пробела (здесь `9`), эти директивы помещают в начало кода. В конце данной директивы точка с запятой `;` не допустима

Названия идентификаторов всегда нужно делать осмысленными, чтобы при возвращении к ранее написанному коду было понятно, зачем нужен каждый из них.

Функция `analogWrite(pin, value)` не возвращает никакого значения и принимает два параметра:

`pin` — номер порта, на который мы отправляем сигнал

`value` — значение величины ШИМ, которое мы отправляем на порт. Он может принимать целочисленное значение от 0 до 255, где 0 — это 0%, а 255 — это 100%

Рассмотрим пример листинга программы:

```
// даём имя для пина №9 со светодиодом
// (англ. Light Emitting Diode или просто «LED»)
#define LED_PIN 9
```

```
void setup()
{
  // настраиваем пин со светодиодом в режим выхода
pinMode(LED_PIN, OUTPUT);
}

void loop()
{
  // выдаём ШИМ-сигнал на светодиод
// Микроконтроллер переводит число от 0 до 255 к напряжению
  // от 0 до 5 В.
  analogWrite(LED_PIN, 85);
  // держим такую яркость 250 миллисекунд
  delay(250);

  // выдаём 170, т.е. 2/3 от 255, или 3,33 В.
  analogWrite(LED_PIN, 170);
  delay(250);

  // далее 5 В — полная мощность
  analogWrite(LED_PIN, 255);
  // ждём перед тем, как начать заново
  delay(250);
}
```

Далее необходимо убедиться в изменении яркости светодиода согласно заданию и далее подключить выход микроконтроллера к щупу осциллографа и получить зависимость выходного напряжения от времени.

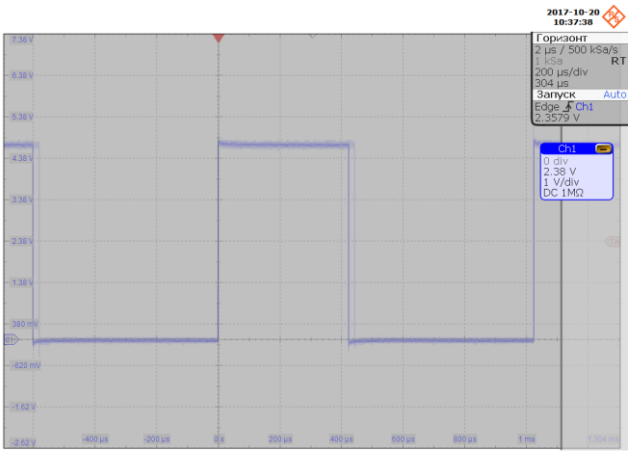

Рисунок 3.8 – ШИМ сигнал с выхода микроконтроллера на осциллографе

На основании полученной осциллограммы необходимо измерить амплитуду сигнала, рассчитать частоту и величину ШИМ и внести полученные результаты и графики в отчет.

3. Задание на выполнение работы

Задание на выполнение работы состоит в реализации принципиальной схемы и загрузке программы управления. Необходимо продемонстрировать понимание аппаратно-программных принципов работы системы, показать,

как реализовать различные режимы мощности сигнала (PWM) и времени действия (T). Параметры следует выбирать из таблицы 1 в соответствии с номером студента в списке группы.

Таблица 1 Задания на выполнение работы

№	Задание
1	*PWM1 = 20%, T1 = 100 мс, PWM2 = 80%, T1 = 500 мс,*
2	*PWM1 = 30%, T1 = 200 мс, PWM2 = 50%, T1 = 1000 мс,*
3	*PWM1 = 50%, T1 = 700 мс, PWM2 = 65%, T1 = 500 мс,*
4	*PWM1 = 50%, T1 = 600 мс, PWM2 = 20%, T1 = 400 мс,*
5	*PWM1 = 30%, T1 = 300 мс, PWM2 = 90%, T1 = 200 мс,*
6	*PWM1 = 10%, T1 = 200 мс, PWM2 = 25%, T1 = 400 мс,*
7	*PWM1 = 50%, T1 = 500 мс, PWM2 = 70%, T1 = 800 мс,*
8	*PWM1 = 70%, T1 = 200 мс, PWM2 = 90%, T1 = 800 мс,*
9	*PWM1 = 85%, T1 = 300 мс, PWM2 = 20%, T1 = 700 мс,*
10	*PWM1 = 100%, T1 = 100 мс, PWM2 = 40%, T1 = 400 мс,*

Далее отключите светодиод от 9-го порта и подключите к 11-му. Измените программу так, чтобы схема снова заработала

Измените код программы так, чтобы в течение секунды на светодиод последовательно подавалось усреднённое напряжение 0, 1, 2, 3, 4, 5 В.

Возьмите еще один светодиод, резистор на 220 Ом и соберите аналогичную схему на этой же плате, подключив светодиод к пину номер 3 и другому входу GND, измените программу так, чтобы светодиоды мигали в противофазе: первый выключен, второй горит максимально ярко и до противоположного состояния

4. Контрольные вопросы

1. Почему мы не сможем регулировать яркость светодиода, подключенного к порту 7?
2. Какое усреднённое напряжение мы получим на пине 6, если вызовем функцию `analogWrite(6, 153)`?
3. Какое значение параметра `value` нужно передать функции `analogWrite`, чтобы получить усреднённое напряжение 2 В?

4. Лабораторная работа: Исследование работы аналого-цифрового преобразователя на базе микроконтроллере ATMega

Цель работы*: Ознакомление с возможностями АЦП платы управления Arduino Uno и средой программирования Arduino IDE, обработка аналогового сигнала на базе Arduino и программирование микроконтроллера.*
Объект исследования: *плата Arduino Uno.*
Аппаратные средства: *Персональный компьютер, Arduino IDE, набор Амперка Матрешка .*

1. Общие теоретические сведения

При использовании аналогового (непрерывного) сигнала, показания датчика передаются в виде переменного напряжения на сигнальном (информационном) проводе. Сигнальное напряжение может принимать значение от 0 В до напряжения питания. Хотя обычно «рабочий диапазон» напряжений более узкий.

На плате Arduino Uno имеется 6 аналоговых входов с помощью которых можно считывать переменное напряжение, и исходя из его значения получать значения с датчика. Эти входы объединены на плате в группу «Analog In» и пронумерованы от A0 до A5 (на других платах может быть другое число входов АЦП).

Между измеряемой величиной и выдаваемым напряжением установлена определённая зависимость, в зависимости от типа датчика, например: пропорциональная или логарифмическая. Иногда зависимость более сложная: напряжение растёт до определённого значения, затем падает пропорционально ему.

Так например, инфракрасный дальномер типа Sharp измеряет расстояние до объекта перед ним: чем меньше расстояние, тем больше напряжение. Если объект находится на расстоянии 20 см, сенсор выдаёт ~2.5 В на сигнальном проводе; на расстоянии 60 см ~ 1 В; на расстоянии 150 см ~ 0.4 В. Точная диаграмма зависимости напряжения от расстояния для инфракрасного дальномера приводится в технических характеристиках. Для других типов сенсоров диаграммы можно так же найти в документации или получить экспериментально.

Для аппаратной обработки аналогового сигала применятся встроенный в МК 10-разрядный АЦП. Для того, чтобы получить данные с него в Arduino существует стандартная функция analogRead (). Так, например, если вы подключили сенсор к контакту A5, чтобы получить показания сенсора в переменную value достаточно указать:

```
int value = analogRead(A5);
```

Диапазон входного напряжения от 0 до 5 В в программе проецируется на диапазон целочисленных значений от 0 до 1023. Перевести полученное значение в необходимые физические единицы, такие как, например, расстояние, поможет функция map. Работа функции чтения АЦП занимает около 100 микросекунд, поэтому максимальная скорость чтения сигналов составит около 10.000 отсчетов в секунду.

Преимуществом сенсоров с аналоговым сигналом является простота их подключения к платам Arduino. Кроме того, поскольку показания датчика можно считывать всего одной командой, лишние килобайты памяти на микроконтроллере не расходуются на хранение алгоритма расшифровки протокола, присущего цифровым сенсорам.

Главным недостатком аналогового сигнала является неустойчивость к внешним шумам. Если провод от сенсора до микроконтроллера будет достаточно длинным, он начнёт работать как антенна и улавливать внешние электромагнитные поля: провод сам будет влиять на выходное напряжение и тем самым искажать показания. Поэтому разумный предел длины провода для аналогового сенсора — не более 50 см.

Чтобы уменьшить влияние помех на полезный сигнал можно воспользоваться усреднением. Так как помехи носят случайный характер, они будут влиять на полезный сигнал тем меньше, чем больше выборок используется для усреднения. Один из простейших вариантов цифровой фильтрации сигнала на основании усреднения показан ниже:

```
#define SENSOR_PIN A5

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

void loop()
{
    delay(1000);
    long val = 0;
    for (int i=0; i<100; ++i) {
        val = val + analogRead(SENSOR_PIN);
    }
    val = val/100;
    Serial.println(val);
```

```
}
```

Аналоговый сигнал при чтении на Arduino может иметь максимум 1024 градации. С учётом того, что рабочий диапазон почти всегда меньше допустимого, полезных значений ещё меньше, что может оказаться недостаточно для высокоточных измерений.
На ATmega328p, установленном на Arduino, как и на большинстве других микроконтроллеров аналоговых входов не много. Поэтому количество одновременно контролируемых аналоговых сенсоров ограничено. У Arduino Uno — их 6, у Arduino Mega 2560 — 16.

2. Выполнение лабораторной работы

В качестве аналогового датчика используется переменный резистор (потенциометр) на основе углеродистого проводника. Поворотом движка изменяет сопротивление от нуля до номинального сопротивления в 10 кОм.

Характеристики:
- Мощность: 0.25 Вт
- Точность: ±20 %
- Максимальное рабочее напряжение: 250 В
- Угол поворота движка: 250°
- Характеристика изменения сопротивления: линейная

В работе необходимо подключить светодиод к выходу микроконтроллера и написать программу, которая будет включать и выключать светодиод согласно повороту потенциометра. Для работы из набора Амперка Матрешка соберите следующую принципиальную схему с использованием беспаечной макетной платы. Зависимость яркости от угла поворота необходимо выбрать из таблицы с заданием. При отсутствии физической платы допустимо использовать виртуальный комплекс моделирования с использованием ресурса www.tinkercad.com или аналог.

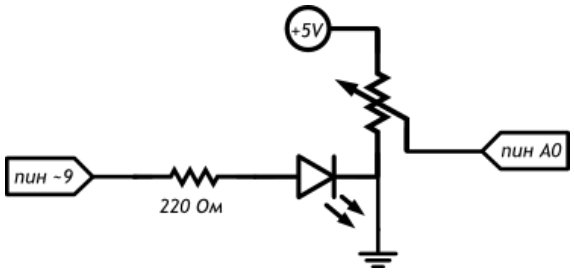

Рисунок 4.1 – Принципиальная схема

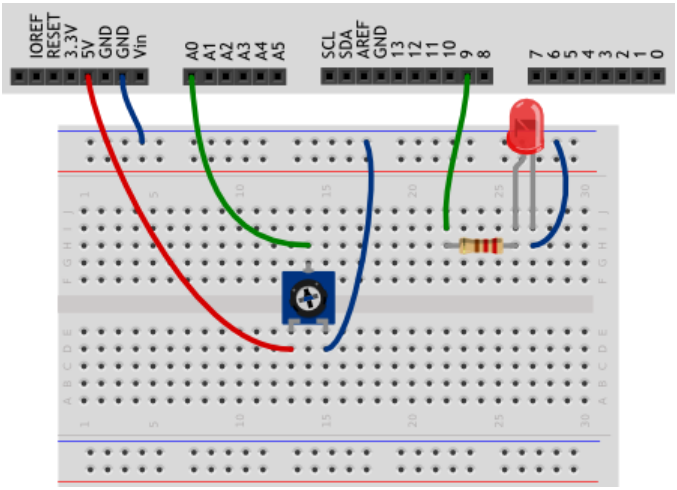

Рисунок 4.2 – Схема подключения

После подключения элементов цифровой системы управления напишите код программы, аналогично примеру. Обратите внимание, что с помощью директивы #define мы говорим компилятору заменять идентификатор POT_PIN на A0 — номер аналогового входа. Можно встретить код, где обращение к аналоговому порту будет по номеру без индекса A. Такой код будет работать, но во избежание ошибок с цифровыми портами используйте индекс. Переменным принято давать названия, начинающиеся со строчной буквы.

Чтобы использовать переменную, необходимо ее объявить, что мы и делаем инструкцией:

```
int rotation, brightness;
```

Для объявления переменной необходимо указать ее тип, здесь — int (от англ. integer) — целочисленное значение в диапазоне от -32 768 до 32 767.

Функция `analogRead(pinA)` возвращает целочисленное значение в диапазоне от 0 до 1023, пропорциональное напряжению, поданному на аналоговый вход, номер которого мы передаем функции в качестве параметра pinA

Обратите внимание, как мы получили значение, возвращенное функцией `analogRead()`:поместили его в переменную rotation с помощью оператора присваивания =, который записывает то, что находится справа от него в ту переменную, которая стоит слева.

Пример программы:

```
// даём имена для пинов со светодиодом
// и потенциометром (англ potentiometer или «pot»)
#define LED_PIN     9
#define POT_PIN     A0

void setup()
{
```

```
  // пин со светодиодом — выход
  pinMode(LED_PIN, OUTPUT);

  // пин с потенциометром должен быть входом
  // (англ. «input»): мы хотим считывать напряжение,
  // выдаваемое им
  pinMode(POT_PIN, INPUT);
}

void loop()
{
  // объявляем, что далее мы будем использовать 2 переменные с
  // именами rotation и brightness, и что хранить в них будем
  // целые числа (англ. «integer», сокращённо просто «int»)
  int rotation, brightness;

  // считываем в rotation напряжение с потенциометра:
  // микроконтроллер выдаст число от 0 до 1023
  // пропорциональное углу поворота ручки
  rotation = analogRead(POT_PIN);

  // в brightness записываем полученное ранее значение rotation
  // делённое на 4. Поскольку в переменных мы пожелали хранить
  // целые значения, дробная часть от деления будет отброшена.
  // В итоге мы получим целое число от 0 до 255
  brightness = rotation / 4;

  // выдаём результат на светодиод
  analogWrite(LED_PIN, brightness);
}
```

На основании полученных результатов необходимо измерить яркость сигнала, рассчитать зависимость ШИМ от угла поворота ручки и напряжения на входе АЦП и внести полученные результаты в таблицу. Обратите внимание на возможный выход значений АЦП и ШИМ за диапазон и оптимизируйте формулу преобразования.

3. Задание на выполнение работы

Задание на выполнение работы состоит в реализации принципиальной схемы и загрузке программы управления. Необходимо продемонстрировать понимание аппаратно-программных принципов работы системы, показать, как реализовать различные режимы мощности сигнала (PWM) в зависимости от значения аналогового сигнала на АЦП. Параметры следует выбирать из таблицы 1 в соответствии с номером студента в списке группы.

Таблица 1

№	Задание
1	*LED_PIN= 9, POT_PIN=A0, brightness = rotation / 4*

2	*LED_PIN= 5, POT_PIN=A2, brightness = rotation / 4*
3	*LED_PIN= 6, POT_PIN=A4, brightness = rotation / 3*
4	*LED_PIN= 9, POT_PIN=A1, brightness = rotation / 2*
5	*LED_PIN= 5, POT_PIN=A2, brightness = rotation / 2*
6	*LED_PIN= 6, POT_PIN=A3, brightness = rotation* 2*
7	*LED_PIN= 9, POT_PIN=A0, brightness = rotation* 4*
8	*LED_PIN= 5, POT_PIN=A3, brightness = rotation* 4*
9	*LED_PIN= 6, POT_PIN=A4, brightness = rotation / 3*
10	*LED_PIN= 9, POT_PIN=A1, brightness = rotation / 3*

4. Контрольные вопросы

1. Можем ли мы при сборке схемы подключить светодиод и потенциометр напрямую к разным входам GND микроконтроллера?
2. В какую сторону нужно крутить переменный резистор для увеличения яркости светодиода?
3. Что будет, если стереть из программы строчку `pinMode(LED_PIN, OUTPUT)`? строчку `pinMode(POT_PIN, INPUT)`?
4. Зачем мы делим значение, полученное с аналогового входа перед тем, как задать яркость светодиода? что будет, если этого не сделать?

5. Лабораторная работа: Организация светодиодной индикации с обратной связью на микроконтроллере

***Цель работы**: Ознакомление с реализацией цифрового управления с обратной связью на ядре Arduino Uno, обработка аналогового сигнала на базе Arduino и формирование цифрового управляющего воздействия.*
***Объект исследования:** плата Arduino Uno.*
***Аппаратные средства:** Персональный компьютер, Arduino IDE, набор Амперка Матрешка, цифровой осциллограф R&S RTO2034.*

1. Общие теоретические сведения

Фоторезистор VT90N2 — сенсор, меняющий сопротивление в зависимости от количества света падающего на него. В полной темноте он имеет максимальное сопротивление в сотни КОм, а по мере роста освещённости сопротивление уменьшается до десятков КОм.
На его основе очень просто создать схему, которая бы поставляла данные об уровне освещённости в виде аналогового сигнала на управляющую электронику. Для этого достаточно сделать элементарный делитель напряжения, где одним из резисторов будет фоторезистор, а вторым — резистор на 100 кОм.

Характеристики сенсора:

- Теневое сопротивление (около 0 люкс): 500 кОм
- Сопротивление при 10 люкс: 24 ± 12 кОм

Для удобства отладки и настройки Arduino имеет последовательный порт, реализованный на интерфейсе UART, через который можно передавать текущие значения переменных в монитор порта с помощью команды `Serial.print()`.
Для инициализации порта используется команда `Serial.begin(9600),` в которую передается необходима скорость обмена данных по порту.

2. Выполнение лабораторной работы

В работе необходимо подключить светодиод к выходу микроконтроллера и написать программу, которая будет плавно изменять яркость светодиода в пропорциональной зависимости (П-регулятор) от уровня внешней освещенности (обратной связи), имеющей разную величину разрядности из задания (от 7 бит до 9 бит). Уровень внешней освещенности необходимо

настроить экспериментально таким образом, чтобы минимальная освещенность соответствовала 0, а максимальная – максимальному значению для разрядности. Величина коэффициента пропорциональности определяется согласно повороту потенциометра, диапазон значений определяется также из задания. Для работы из набора Амперка Матрешка соберите схему с использованием макетной платы, фоторезистор и потенциометр подключаются к входам АЦП, светодиод – на выход с ШИМ. Постарайтесь разместить компоненты так, чтобы светодиод не засвечивал фоторезистор.
При отсутствии физической платы допустимо использовать виртуальный комплекс моделирования с использованием ресурса www.tinkercad.com или аналога.

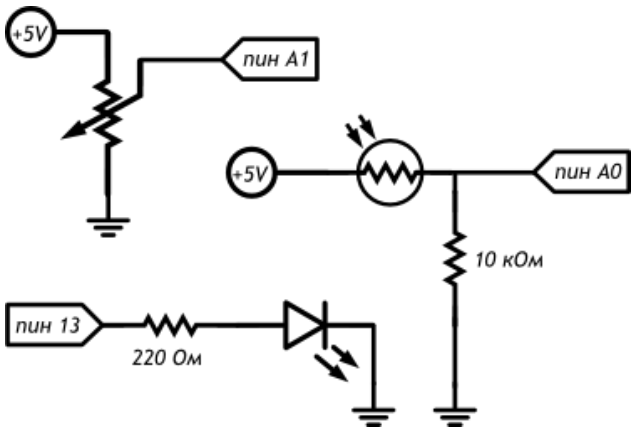

Рисунок 5.1 – Схема подключения элементов

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

```
#define LED_PIN  11
#define LDR_PIN  A0
#define POT_PIN  A1

void setup()
{
  pinMode(LED_PIN, OUTPUT); //инициализация пина светодиода на выход
  Serial.begin(9600); // инициализация последовательного порта на скорости
9600 бит\с
}

void loop()
{
  // считываем уровень освещённости с сенсора в 8-разрядном формате, чтобы
  // при максимальной освещенности было значение около 250,
  // а при минимальной - около нуля (экспериментально подобрано)
  int lightness = map (analogRead(LDR_PIN), 60, 1023, 0, 255);

  // считываем значение с потенциометра, которым мы регулируем
  // коэффициент пропорциональности регулятора в диапазоне от 0 до 10
  int koef_prop = map (analogRead(POT_PIN), 0, 1023, 0, 10);
```

```
  // рассчитываем управляющее значение, пропоциональное
  // уровню внешней освещенности
  int control = (koef_prop*lightness);
// формируем управляющее напряжение
    analogWrite(LED_PIN, control);
// посылаем строки с параметрами системы в монитор порта
    Serial.print("lightness="); // текущий уровень освещенности
    Serial.println(lightness);
    Serial.print("koef_prop=");  // текущий коэффициент пропорциональности
    Serial.println(koef_prop);
    Serial.print("control="); // текущее управляющее значение
    Serial.println(control);
    Serial.println();
}
```

Обратите внимание, что все переменные являются целочисленными. В результате выполнения программы яркость светодиода должна меняться из-за уровня освещенности, а так же настроек коэффициента регулятора. Проведите несколько различных экспериментов с разными параметрами, отслеживая текущие значения переменных с помощью монитора порта.

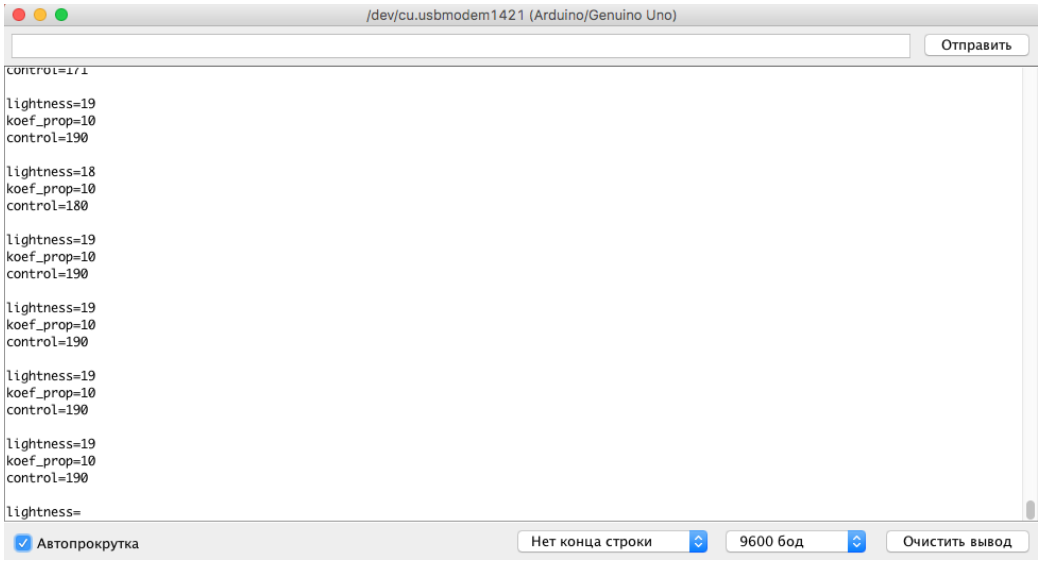

Рисунок 5.2 – Окно монитора порта

Далее подключите к ШИМ-выходу платы осциллограф и получите зависимости выходного сигнала от времени при различных уровнях освещенности.

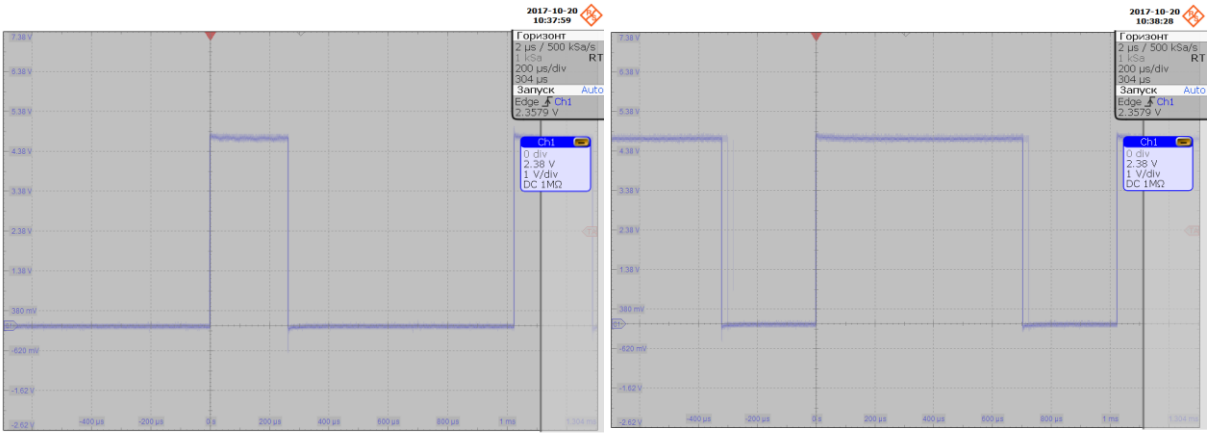

Рисунок 5.3 – ШИМ сигнал с выхода соответствующего яркости при низкой и высокой освещенности

Сопоставьте уровень освещенности и величину ШИМ-сигнала, запишите полученные значения. Далее подключите щуп осциллографа к входу АЦП платы и получите временные зависимости с фоторезистора при различных уровнях освещенности.

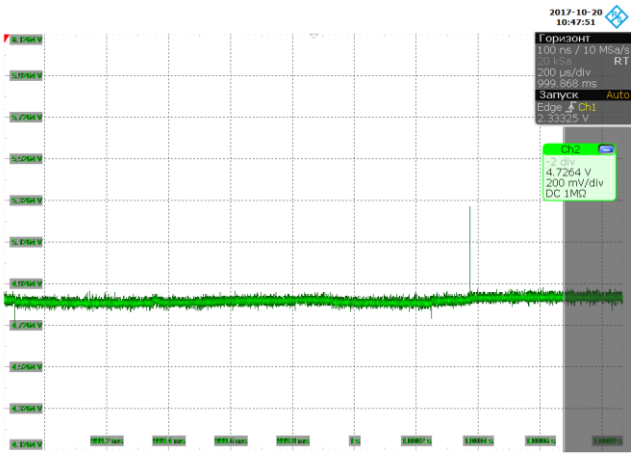

Рисунок 5.4 – Осциллограмма сигнала с фоторезистора

Так как сигнал является зашумленным примените фильтр низки х частот (ФНЧ) в блоке математической обработки MATH на осциллографе.

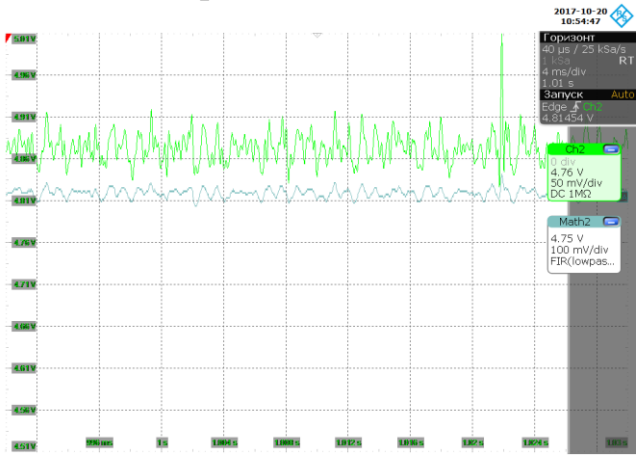

Рисунок 5.5 – Совмещенные осциллограммы сигнала и отфильтрованного сигнала с фоторезистора

На основании полученной осциллограммы оцените необходимую частоту среза для ФНЧ, внесите полученные результаты и графики в отчет.

3. Задание на выполнение работы

Задание на выполнение работы состоит в реализации цифрового П-регулятора с регулируемым коэффициентом на базе платы Arduino и загрузке программы управления. Необходимо продемонстрировать понимание

аппаратно-программных принципов работы системы, показать, как реализовать различный управляющий сигнал в зависимости от значения показаний датчиков. Параметры разрядности датчика освещенности и диапазона коэффициента пропорциональности следует выбирать из таблицы в соответствии с номером студента в списке группы.

№	Задание
1	*8-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…10)*
2	*8-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…5)*
3	*7-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…8)*
4	*7-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…15)*
5	*9-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…10)*
6	*9-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…5)*
7	*9-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…10)*
8	*7-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…20)*
9	*8-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…3)*
10	*8-разрядный* `lightness`*, коэффициент* `koef_prop` *в диапазоне (0…15)*

6. Лабораторная работа: Реализация цифрового ПИ-регулятора на базе микроконтроллера с помощью УЗ-дальномера

***Цель работы**: Ознакомление с реализацией цифрового управления с обратной связью по расстоянию на ядре Arduino Uno, подключение ультразвукового дальномера и изучение принципа работы.*
***Объект исследования:** плата Arduino Uno.*
***Аппаратные средства:** Персональный компьютер, Arduino IDE, набор Амперка Матрешка.*

1. Краткие теоретические сведения

Дальномер может служить датчиком для робота, благодаря которому он сможет определять расстояния до объектов, объезжать препятствия, или строить карту помещения. Его можно также использовать в качестве датчика для сигнализации, срабатывающего при приближении объектов.

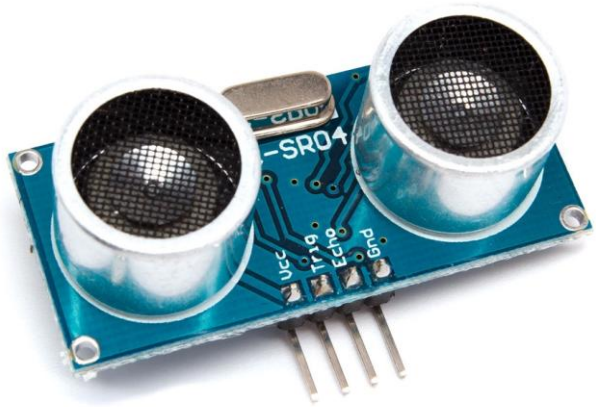

Рисунок 6.1 – Общий вид УЗ-дальномера

Принцип действия: ультразвуковой дальномер генерирует звуковые импульсы на частоте 40 кГц и принимает отраженный от препятствия сигнал. По времени распространения звуковой волны туда и обратно определяется расстояние до объекта.
В отличие от инфракрасных дальномеров, на показания ультразвукового дальномера не влияют засветки от солнца или цвет объекта. Даже прозрачная поверхность будет для него препятствием, но могут возникнуть трудности с определением расстояния до бесформенных или очень тонких предметов.
Vcc — положительный контакт питания.

Trig — цифровой вход. Для запуска измерения необходимо подать на этот вход логическую единицу на 10 мкс. Следующее измерение рекомендуется выполнять не ранее чем через 50 мс.

Echo — цифровой выход. После завершения измерения, на этот выход будет подана логическая единица на время, пропорциональное расстоянию до объекта.

GND — отрицательный контакт питания.

Характеристики:

- Напряжение питания: 5 В
- Потребление в режиме тишины: 2 мА
- Потребление при работе: 15 мА
- Диапазон расстояний: 2–400 см
- Эффективный угол наблюдения: 15°
- Рабочий угол наблюдения: 30°

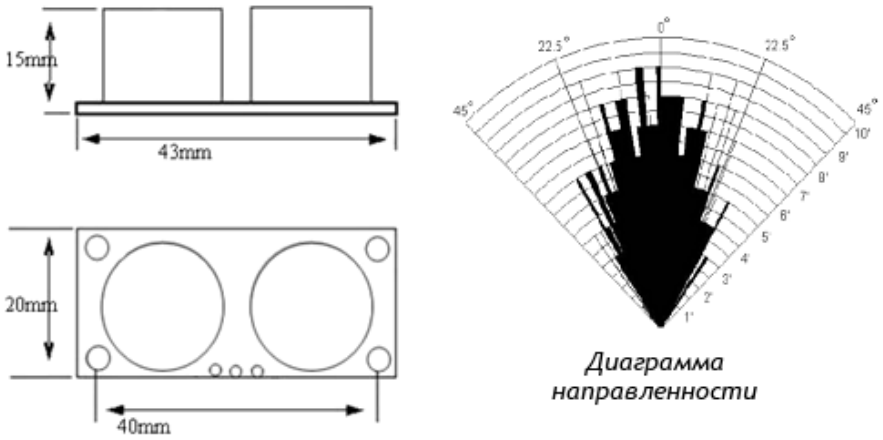

Рис. 6.2 – Параметры дальномера

2. Выполнение работы

Подключите ультразвуковой дальномер к Troyka Shield с помощью четырёх проводов в следующем порядке: пины VCC и GND модуля соедините с аналогичным пинам на Troyka Shield, пин Trig подключите к пину 8, а пин Echo к пину 9.

Подключите 10 светодиодов к цифровым пинам платы, они будут отражать уровень расстояния до препятствия.

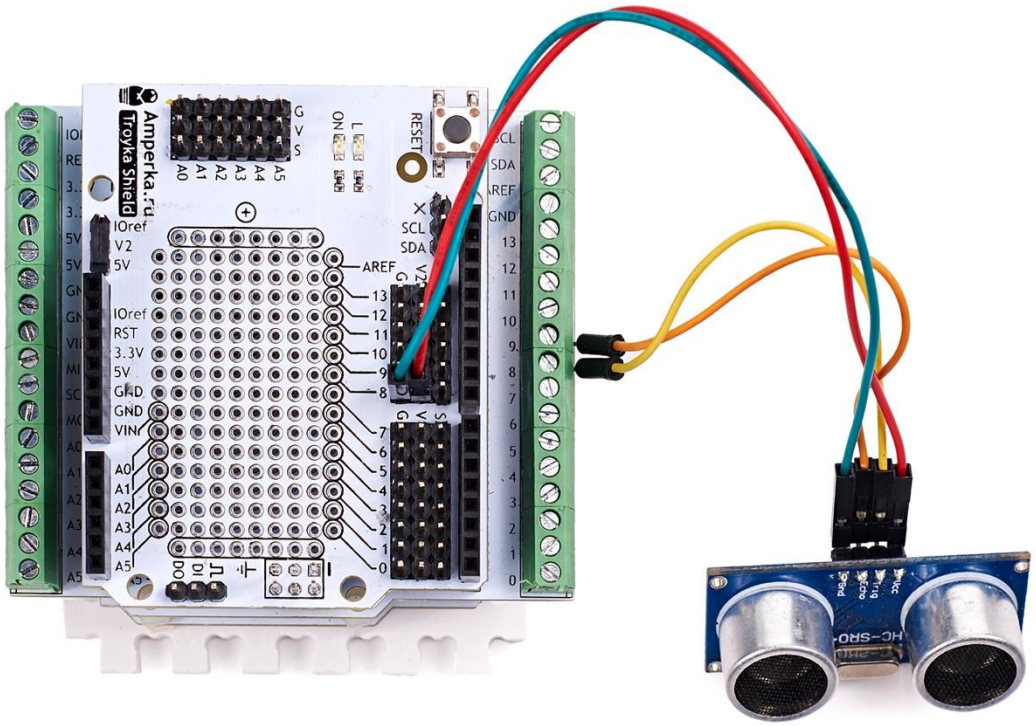

Рисунок 6.3 – Общий вид цифровой системы.

Реализуйте следующий алгоритм работы цифровой системы управления:
1. Измеряем расстояние ультразвуковым датчиком: если расстояние больше 60-70 см, то звуковой и световой индикации нет.
2. С уменьшением расстояния постепенно включаем большее количество светодиодов согласно задающему воздействию пропорционально-интегрального регулятора с постоянными коэффициентами, подобранными экспериментально.
3. Если расстояние становится меньше 5 см, то включаем все светодиоды.

3. Задание на выполнение работы

Задание на выполнение работы состоит в самостоятельной реализации программы цифрового ПИ-регулятора на базе платы Arduino, загрузке программы управления, оптимизации коэффициентов регулятора. Необходимо продемонстрировать понимание аппаратно-программных принципов работы системы, показать, как реализовать различный управляющий сигнал в зависимости от значения показаний датчика.

7. Лабораторная работа: 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 7.1 – 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:

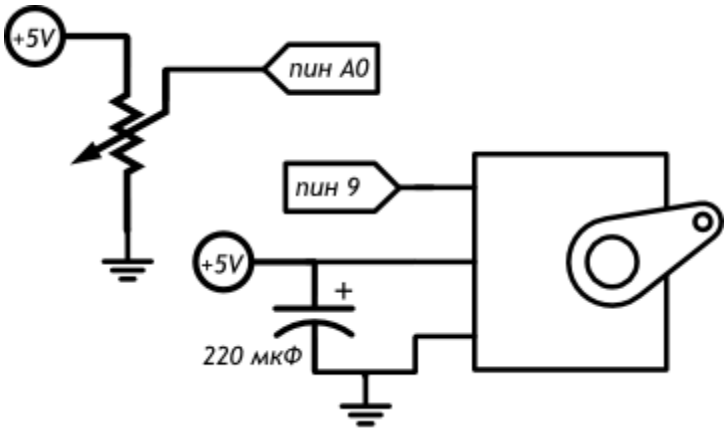

Рисунок 7.2 – Принципиальная схема подключения

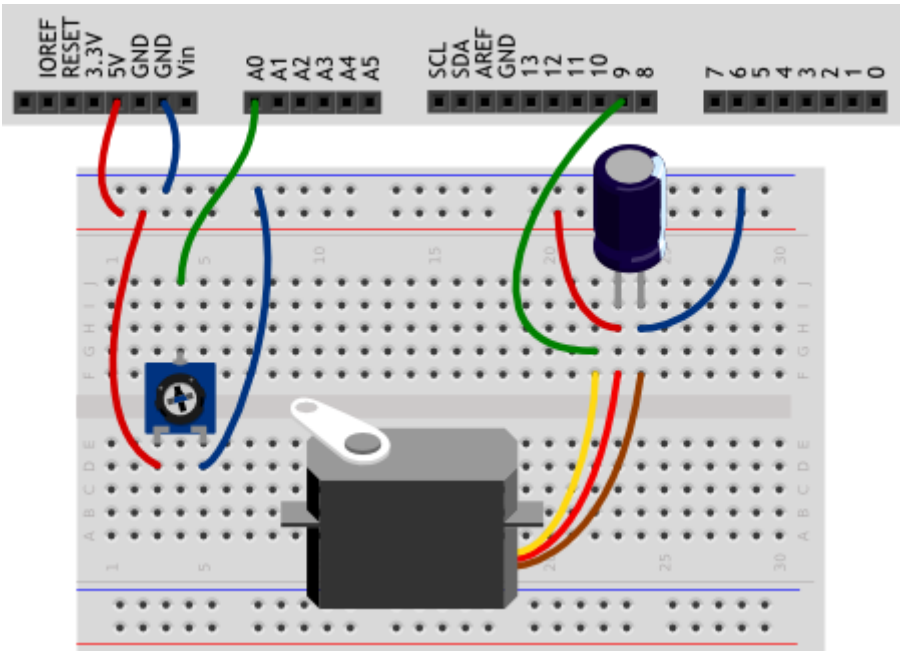

Рисунок 7.3 – Принципиальная схема подключения

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 серве 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 объекты `leftGrip` 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?

8. Лабораторная работа: Изучение системы управления SMART PAD манипулятора KUKA Agilus KR10

Цель работы: *изучение элементов системы управления манипулятором и принципов движения.*
Объект исследования: *манипулятор KUKA Agilus KR10.*
Аппаратные средства: *манипулятор KUKA Agilus KR10.*

1. Краткие теоретические сведения

Робототехническая система **KUKA Agilus KR10** (Рис. 1.1.) включает в себя все узлы промышленного робота, такие как манипулятор (механика робота с электропроводкой), шкаф управления, соединительные кабели, инструмент и детали оснастки.

Промышленный робот (Рис.8.1.) состоит из следующих компонентов:

- Манипулятор(1)
- Система управления роботом(4)
- Ручной программатор smartPAD(2)
- Соединительные кабели(3,5,6)
- Программное обеспечение

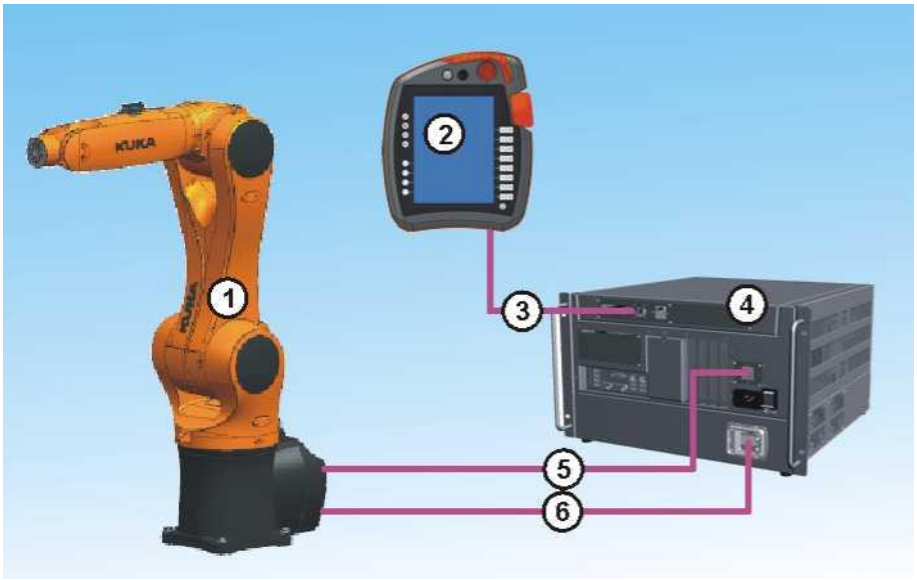

Рисунок 8.1 – Структура промышленного робота KUKA Agilus KR10

Манипулятор — это сочлененная конструкция из металла легкого сплава в исполнении с 6 осями. Каждая ось оснащена тормозом. Все приводные узлы и токоведущие кабели расположены под привинченными крышками, тем самым они защищены от попадания грязи и воды.

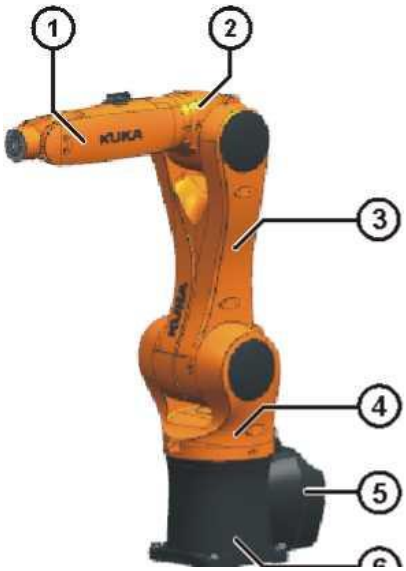

Рис. 8.2 – Устройство манипулятора

Робот оснащен 3-осной центральной рукой. В центральной руке установлены три 5/2-ходовых магнитных клапана и кабели обмена данными CAT5, которые можно использовать для задействования инструментов. На центральной руке имеются также 10-полюсный круглый штекер кабеля ввода/вывода и интерфейс A4 для системы энергоснабжения.

Манипулятор является связующим звеном между центральной рукой и балансиром. Манипулятор приводится в движение двигателем оси 3.

Балансир — это узел, который находится между каруселью и манипулятором. На нем крепится двигатель и редуктор оси 2. В балансире проложены проводка системы энергоснабжения и комплект кабелей для осей 2-6.

На карусели крепятся двигатели осей 1 и 2. Вращение оси 1 выполняется посредством карусели. Карусель привинчена к станине над редуктором оси 1 и приводится в действие размещенным в ней двигателем. В карусели расположен также балансир.

Станина — это основание робота. На задней стороне станины расположен интерфейс A1. К этому интерфейсу подключаются соединительные кабели между механикой робота, системой управления и энергоснабжения.

Электропроводка включает в себя все кабели и линии управления для двигателей осей 1-6. Все соединения выполнены в виде разъемов. К электропроводке относится также блок RDC, интегрированный в роботе. Разъемы двигателя и кабеля обмена данными прикреплены к станине робота. Здесь посредством штекера подключаются соединительные линии от

системы управления роботом. Электропроводка включает в себя также систему заземляющих проводов.

Система управления роботом состоит из следующих компонентов:

1. управляющий компьютер;
2. энергоблок;
3. логика защиты;
4. переносное программирующее устройство smardPAD;
5. панель присоединения.

Коробка управления имеет следующий вид:

Рисунок 8.4 – Общий вид коробки управления

Шкаф управления для малых роботов (CCU_SR) – это центральный интерфейс для распределения энергии и связи между всеми компонентами системы управления роботом. Шкаф CCU_SR состоит из интерфейсной платы шкафа управления для малых роботов (CIB_SR) и платы управления электропитанием для малых роботов (PMB_SR). Все данные передаются к системе управления по внутренним каналам связи и обрабатываются в ней. При сбое электропитания аккумуляторы снабжают током компоненты системы управления до тех пор, пока не будут сохранены данные положений и не будет выключена система управления. С помощью теста на нагрузку можно проверить степень зарядки и качество аккумуляторов. Шкаф CCU_SR также выполняет функции регистрации, управления и переключения.

Приводная коробка (Рис.8.5) состоит из следующих компонентов:

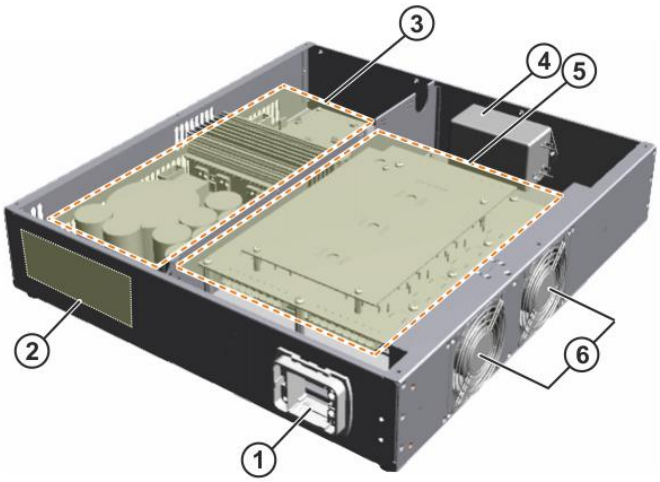

Рисунок 8.5 – Обзор приводной коробки

На рис.8.5 цифрами обозначены:

1 Штекер двигателя X20

2 Тормозной резистор

3 KUKA Servo-Pack для малых роботов (KPP_SR)

4 Сетевой фильтр

5 KUKA Servo-Pack для малых роботов (KSP_SR)

6 Вентиляторы

Приводная коробка выполняет следующие функции: создание напряжения промежуточного контура; управление двигателями; управление тормозами; проверка напряжения в промежуточном контуре в тормозном режиме.

Выполнение лабораторной работы

Выполнение различных траекторий движения происходит на пульте SmartPAD.

Рисунок 8.6 - Пульт SmartPAD

Изучите основные принципы программирования движения: KUKA Робот может двигаться из точки А в точку Б по трем основным направлениям.

1. PTP - точка-точка - движение по кратчайшему пути к конечной точке (Рис. 8.7).

Робот направляет исполнительный орган вдоль наиболее быстрого пути к конечной точке. Быстрый путь, как правило, не кратчайший путь, и поэтому не прямая линия. Это связанно с тем, что движение робота происходит по осям вращения и изогнутые контуры могут быть выполнены быстрее, чем прямые пути. Точный путь движения не может быть предсказан.

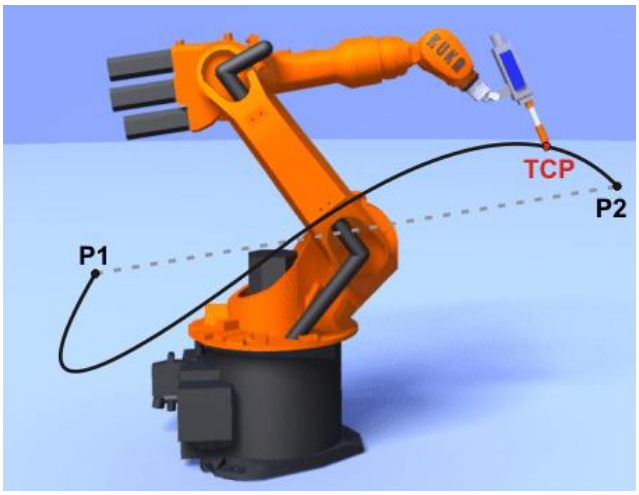

Рис. 8.7 PTP движение

2. LIN - Линейное Движение с определенной скоростью и ускорением по прямой линии (Рис.8.8). Робот использует точку, определенную на предыдущем шаге, как стартовую точку и точку, определенную в текущей команде в качестве конечной точки и проводит интерполяцию прямой линии между этими двумя точками.

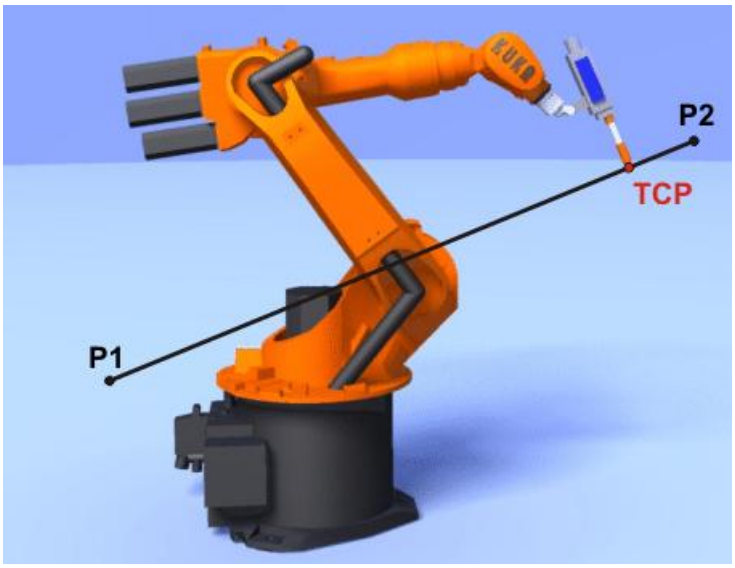

Рис. 8.8 LIN движение

3. CIRC - круговое движение с определенной скоростью и ускорением по круговой траектории или части круговой траектории (Рис.8.9). Используя начальную точку робота (определяется как конечная точка в предыдущей

команде движения) робот интерполирует по круговой траектории, проходящей через среднюю точку и конечную точку.

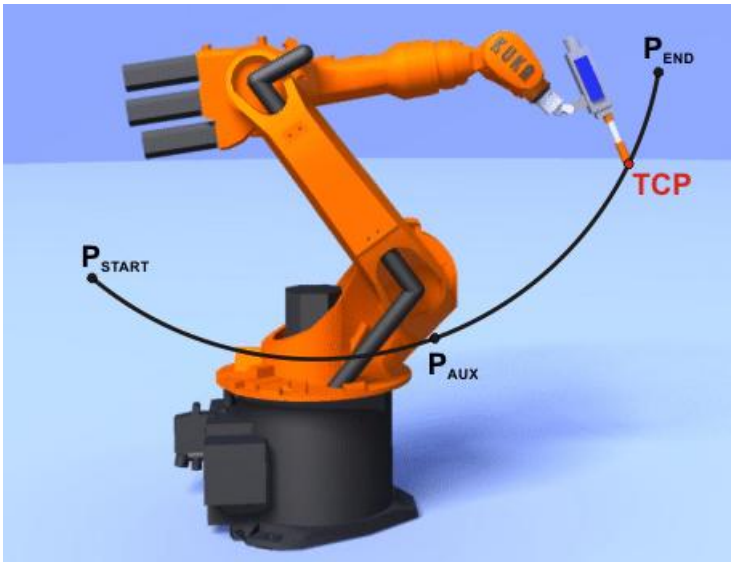

Рис.8.9 CIRC движение.

4. Тип движения "Сплайн"

"Сплайн" является декартовым типом движения, который подходит для особенно сложных криволинейных траекторий. Такие траектории могут быть получены с использованием приближенных LIN и CIRC движение, но Spline, тем не менее, имеет свои преимущества. По итогам проведенных работ подготовьте отчет о с программой и полученными траекториями.

9. Лабораторная работа: 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.

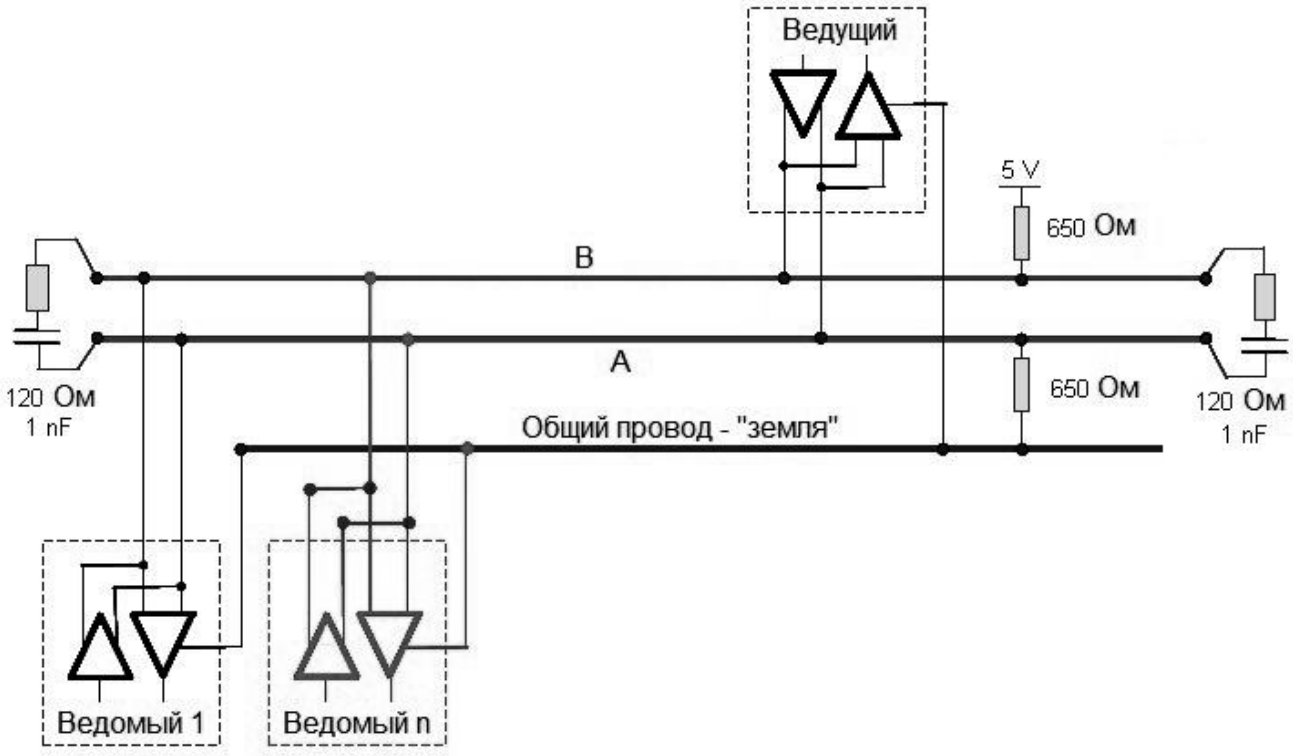

Рисунок 9.1 Схема подключения устройств к шине Modbus, используя интерфейс RS-485

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.

1. 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.

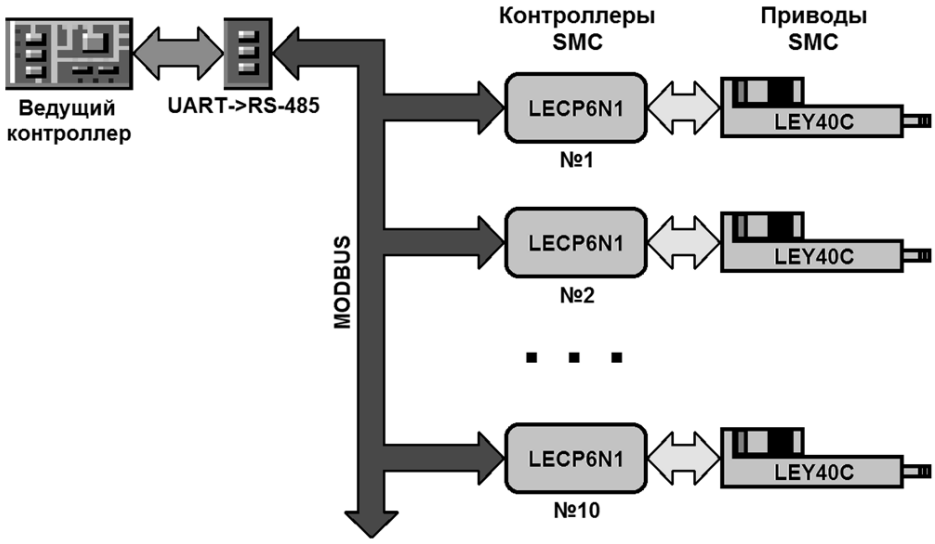

Рисунок 9.2 – Схема коммуникации контроллеров SMC LECP6N1 с ведущим контроллером

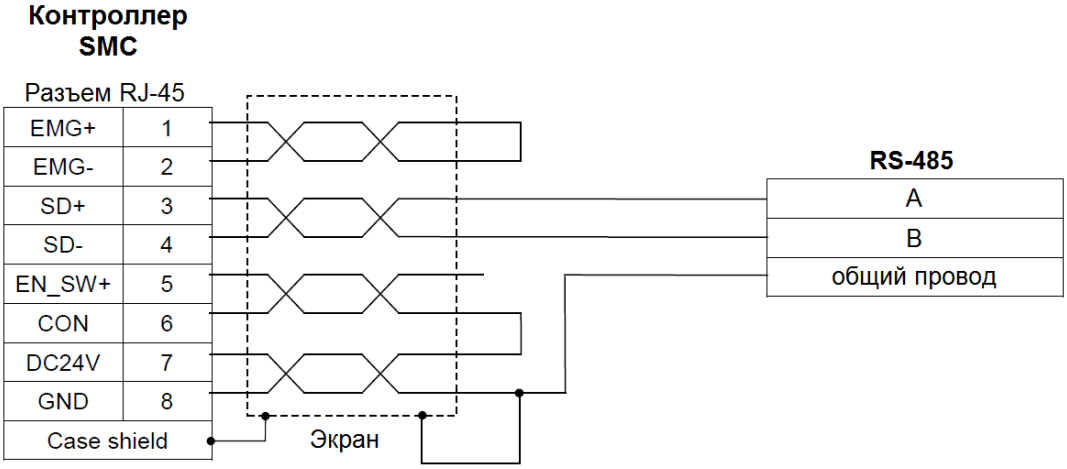

Рисунок 9.3 Подключение интерфейса RS-485 к разъему RJ-45 контроллера SMC LECP6N1

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.

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.

10. Лабораторная работа: 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*

1. Краткие теоретические сведения.

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.

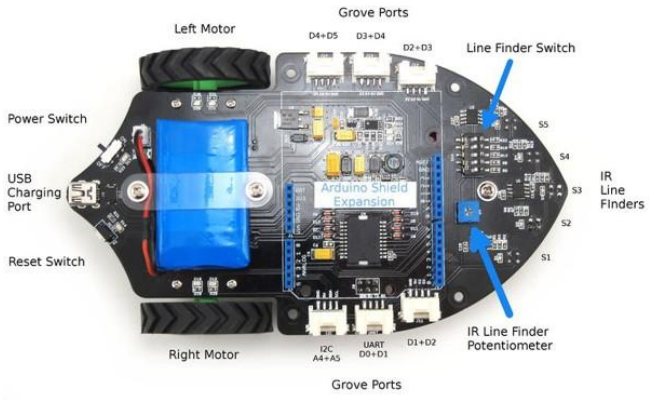

Рисунок 10.1 – Робототехническая платформа ShieldBot

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
5. 900 мАч литий-ионная аккумуляторная батарея
6. USB mini-B разъём для зарядки аккумулятора.

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

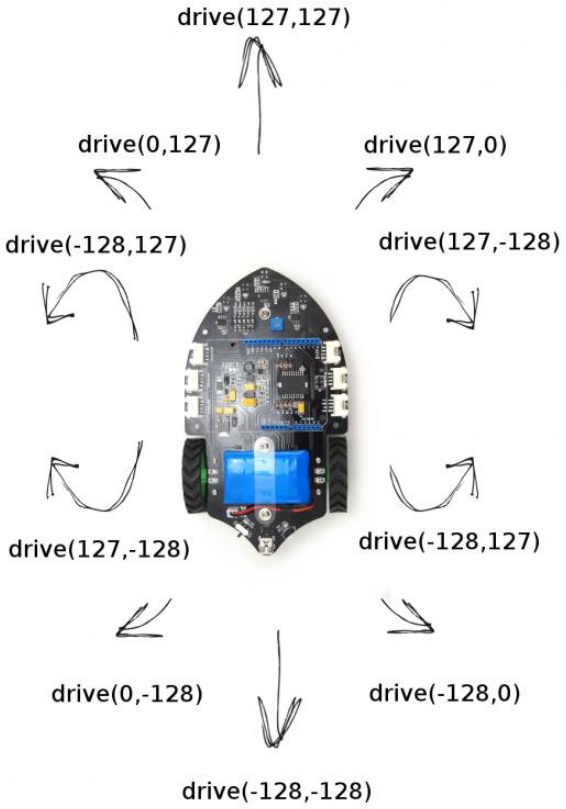

Рисунок 10.2 – Схема управления приводами

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*

Рекомендуемая литература

1. [Электронный ресурс]. - URL: http://wiki.amperka.ru (24.01.2016).
2. [Электронный ресурс]. - URL: http://smc.com (24.01.2016).
3. Цифровые системы автоматизации и управления / Олссон Г., Пиани Д., СПб.: Невский Диалект, 2001, 557с., ил.
4. Макаров, В.Г. Проектирование цифровой системы управления автоматической линии станков : учебное пособие / В.Г. Макаров ; Министерство образования и науки России, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Казанский национальный исследовательский технологический университет». - Казань : Издательство КНИТУ, 2014. - 240 с. : схем., ил. - Библиогр.: с. 237. - ISBN 978-5-7882-1641-6 ;
5. Подураев, Ю.В. Мехатроника: основы, методы, применение [Текст] // М.: Машиностроение, 2007. – 256 с.
6. Яцун С. Ф. Аналого-цифровые системы автоматического управления. :[Текст] : учебное пособие / С. Ф. Яцун, Т. В. Галицына. - Курск: КурскГТУ,. 2007
7. Бесекерский, В. А. Теория систем автоматического управления / В. А. Бесекерский, Е. П. Попов. — 4-е изд., перераб. и доп. — СПб. : Профессия, 2004. — 747 с.