**ФАЙЛ ПРОЕКТОВ И ПОЯСНЕНИЙ**

**К проекту 1.4 Управляемый ночник на RGB-светодиоде**

При использовании эмулятора Arduino проблема дребезга контактов не является актуальной, поэтому имеет смысл исключить часть программы проекта, предназначенную для устранения последствий дребезга контактов кнопки управления. Упрощенная программа проекта будет иметь следующий вид:

// Управляемый ночник на RGB-светодиоде

const int BLED=9;

const int GLED=10;

const int RLED=11;

const int BUTTON=2;

int ledMode=0; // исходный режим светодиода

void setup()

{

pinMode(BLED,OUTPUT);

pinMode(GLED,OUTPUT);

pinMode(RLED,OUTPUT);

pinMode(BUTTON,INPUT);

}

/\*

\* Выбор режима светодиода

\* Передача номера режима и установка заданного режима светодиода

\*/

void setMode(int mode)

{

if(mode==1) // красный

{

digitalWrite(RLED,HIGH);

digitalWrite(GLED,LOW);

digitalWrite(BLED,LOW);

}

if(mode==2) // зеленый

{

digitalWrite(GLED,HIGH);

digitalWrite(RLED,LOW);

digitalWrite(BLED,LOW);

}

if(mode==3) // синий

{

digitalWrite(BLED,HIGH);

digitalWrite(RLED,LOW);

digitalWrite(GLED,LOW);

}

if(mode==4) // пурпурный (красный+синий)

{

analogWrite(RLED,127);

analogWrite(GLED,0);

analogWrite(BLED,127);

}

if(mode==5) // бирюзовый (синий+зеленый)

{

analogWrite(RLED,0);

analogWrite(GLED,127);

analogWrite(BLED,127);

}

if(mode==6) // оранжевый (красный+зеленый)

{

analogWrite(RLED,127);

analogWrite(GLED,127);

analogWrite(BLED,0);

}

if(mode==7) // белый

{

analogWrite(RLED,85);

analogWrite(GLED,85);

analogWrite(BLED,85);

}

if(mode==0) // выключен

{

digitalWrite(BLED,LOW);

digitalWrite(RLED,LOW);

digitalWrite(GLED,LOW);

}

}

void loop()

{

while(digitalRead(BUTTON)==0); // цикл ожидания нажатия кнопки

ledMode++; // изменение режима светодиода

if(ledMode==8)ledMode=0; // режимы изменяются от 0 до 7

setMode(ledMode); // установка режима светодиода

while(digitalRead(BUTTON)==1); // цикл ожидания отпускания кнопки

}

**К проекту 4.1. Создание мелодии**

Подобно Лабораторной работе № 4 мелодия задается в виде двух массивов: массива нот и массива длительностей звучания. Количества элементов в этих массивах одинаковы, равные количеству звуков в мелодии.

Однако, в отличие от программирования микроконтроллера ADuC842, где нам приходилось вычислять требуемые настройки таймеров для каждой ноты, в Arduino формирование массивов проще. Каждый элемент массива нот – число, равное частоте звука в Герцах, а каждый элемент массива длительностей – число, равное продолжительности звучания в миллисекундах.

Воспроизведение одной из мелодий лабораторной работы может быть реализовано на Arduino следующей программой:

const int SPEAKER=9;

int notes[ ]={294,392,392,392,294,523,523,523,294,466,

466,466,440,392,440,466,0};

int times[ ]={125,125,125,125.125,125,125,125,125,125,

125,63,63,63,63,125,250};

void setup()

{

for(int i=0;i<17;i++)

{

tone(SPEAKER,notes[i],2\*times[i]);

delay(3\*times[i]);

}

}

void loop()

{

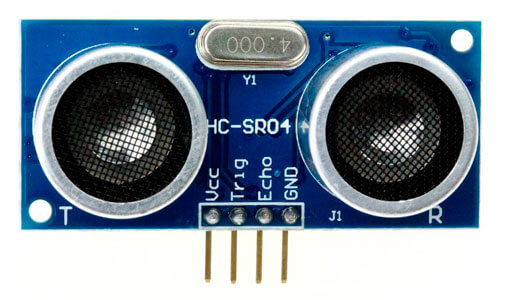
}

**Проект 4.2. Ультразвуковой дальномер HC-SR04**

Датчики расстояния являются неотъемлемой частью любого робота. Ультразвуковой дальномер модуль HC-SR04 для Arduino— это помещенные на одну плату приемник и передатчик ультразвукового сигнала. Принцип действия HC-SR04 основан на хорошо известном явлении эхолокации. Излучатель формирует акустический сигнал, который отразившись от преграды, возвращается к датчику и регистрируется приемником. Зная скорость распространения ультразвука в воздухе и время запаздывания между излученным и принятым сигналом, легко рассчитать расстояние до акустической преграды. В отличие от инфракрасных дальномеров на ультразвуковой датчик HC-SR04 не влияют источники света или цвет препятствия. Могут возникнуть затруднения при определении расстояния до пушистых или тонких объектов. Кроме приемника и передатчика на плате находится необходимая обвязка. Модуль имеет 4 вывода стандарта 2,54 мм:

VCC — питание +5 В; Trig (T) — вывод входного сигнала;

Echo (R) — вывод выходного сигнала; GND — земля.



Последовательность действий по измерению расстояния следующая:

* подаем импульс продолжительностью 10 мкс на вывод Trig;
* на плате модуля входной импульс преобразуется в 8 импульсов частотой 40 кГц и посылается через излучатель T;
* дойдя до препятствия, посланные импульсы отражаются и принимаются приемником R, в результате получаем выходной сигнал на выводе Echo.
* На стороне контроллера переводим полученный сигнал в расстояние по формуле:
* ширина импульса (мкс) / 58 = дистанция (см).

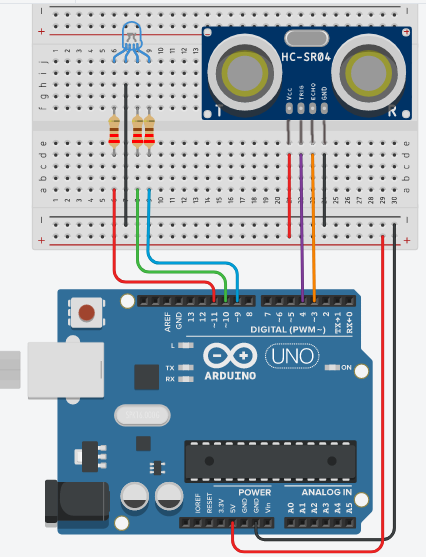
Рассмотрим простой пример работы с ультразвуковым дальномером.

Подключим вход запуска излучателя Trig к порту вывода 4, а выходной контакт дальномера Echo – к порту ввода 3. Результаты измерения расстояния **L\_sm** будем выводить в последовательный порт Arduino IDE - функция Serial.println( ) .

Кроме того, будем индицировать величину расстояния на RGB-светодиоде:

При L\_sm < 50 sm – цвет свечения – красный, при L\_sm > 200 sm – синий, а в промежутке от 50 до 200 sm – зеленый.

Монтажная схема проекта будет иметь следующий вид:



Программа проекта:

// Ультразвуковой дальномер

const int Trig=4; // Вход запуска дальномера подключить к порту 4

const int Echo=3; // выход Echo дальномера подключить к порту 3

const int RLED=11; // красный светодиод

const int GLED=9; // зеленый светодиод

const int BLED=10; // синий светодиод

long Duration; // переменная для измерения длительности (в мкс)

long L\_sm; // переменная для пересчета мкс в сантиметры расстояния

void setup()

{

pinMode(Trig,OUTPUT); // Trig для микроконтроллера – выходной сигнал

pinMode(Echo,INPUT); // Echo – для микроконтроллера – входной сигнал

pinMode(RLED,OUTPUT);

pinMode(GLED,OUTPUT);

pinMode(BLED,OUTPUT);

Serial.begin(9600); // включение последовательного порта

}

void loop() {

digitalWrite(Trig,LOW); // реализация плана действий с дальномером:

delayMicroseconds(2); // На входе Trig в течение 2 мкс держится 0

digitalWrite(Trig,HIGH); // затем на входе Trig формируется единичный импульс

delayMicroseconds(10); // продолжительностью 10 мкс; идет излучение ультразвука

digitalWrite(Trig,LOW); // а затем сигнал Trig сбрасывается

Duration=pulseIn(Echo,HIGH); // функция pulsing( ) определяет продолжительность

// интервала (в микросекундах) от начала посылки ультразвука до прихода отраженного

// сигнала

L\_sm=Duration/58; // пересчет микросекунд в сантиметры расстояния с учетом

//скорости распространения звука и пути до предмета и обратно

if (L\_sm<50) // выбор цвета свечения RGB-светодиода

{ // в зависимости от расстояния до предмета

digitalWrite(RLED,HIGH);

digitalWrite(GLED,LOW);

digitalWrite(BLED,LOW);

}

else

if (L\_sm>200)

{

digitalWrite(RLED,LOW);

digitalWrite(BLED,HIGH);

digitalWrite(GLED,LOW);

}

else

{

digitalWrite(RLED,LOW);

digitalWrite(BLED,LOW);

digitalWrite(GLED,HIGH);

}

Serial.print(L\_sm); // вывод величины расстояния на монитор последовательного порта

Serial.println(" sm"); // с указанием единиц измерения расстояния

delay(500); // задержка 0,5 сек между измерениями

}

**К проекту 5.3 Отправка последовательности цифр для управления RGB-светодиодом**

В программе, приводимой в листинге 6.5 книги Д. Блума, команду конца передачи

**if (Serial.read( ) == ‘\n’)**

следует исключить из программы, так как этот символ уже признан концом передачи третьего числа.

**Проект 7.1. Регулятор температуры**

Типовые элементы микропроцессорных систем, изученные ранее, позволяют создавать системы автоматического управления, которые содержат чувствительные элементы – аналоговые и цифровые датчики и разнообразные исполнительные элементы, и все они имеют единое управление от микроконтроллера, работающего по программе.

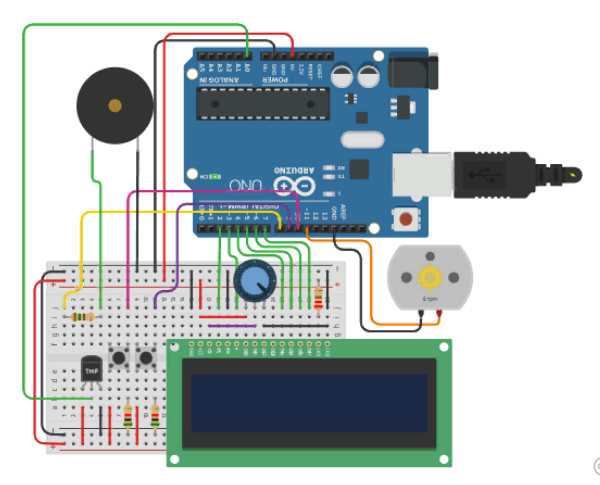
В качестве примера подобной системы рассмотрим в настоящем проекте Регулятор температуры, работающий по следующему алгоритму:

1. Регулятор содержит аналоговый датчик температуры (TMP36, как в проекте 2.2.), который служит для измерения текущего значения температуры воздуха.
2. В регуляторе температуры имеются также элементы, с помощью которых задается значение пороговой температуры. Это кнопки UP и DOWN, при удержании которых будет происходить увеличение (или, соответственно, уменьшение) значения пороговой температуры в ритме около 1 градуса за 0,2 сек удержания.
3. Принцип регулирования температуры окружающего около воздуха заключается в том, что если при постоянном сравнении текущего и порогового значений температуры обнаруживается, что текущее значение выше порогового, то включается вентилятор. При этом появляется кратковременный звуковой сигнал.
4. Состояние элементов системы должно непрерывно отображаться на экране ЖК-дисплея: значения текущей и пороговой температуры, включенное/ отключенное состояние вентилятора.

Таким образом, в состав Регулятора температуры будут входить следующие компоненты:

1. Аналоговый датчик температуры TMP36 (подключаем к портуА0);
2. Кнопка DOWN \_B (подключаем к порту 10 с R=5кОм, подтягивающим к нулю);
3. Кнопка UP \_B (подключаем к порту 9 с R=5кОм, подтягивающим к нулю);
4. Двигатель постоянного тока (вентилятор) (подключаем к порту 11);
5. Пьезоэлемент (подключаем через ограничивающее R=150 Ом к порту 8)
6. ЖК дисплей на 16 столбцов и 2 строки (подключаем стандартным способом, используя порт 2 как сигнал RS, порт 3 как сигнал Е и 4 линии для передачи байтов, подключенные к портам 4,5,6,7; подключаем также светодиод подсветки ЖКИ через ограничивающее R=220 Ом), а также используем потенциометр Rp=1кОм для регулировки контрастности экрана ЖКИ).

Монтажная схема регулятора температуры будет выглядеть следующим образом:



Программа с комментариями приведена ниже:

#include <LiquidCrystal.h>

LiquidCrystal lcd(2,3,4,5,6,7); // инициализация экземпляра библиотеки

byte degree[8]={ // пользовательский символ градуса

B00110,

B01001,

B01001,

B00110,

B00000,

B00000,

B00000,

B00000,

};

byte fan\_on[8]={ // пользовательский символ включения вентилятора

B00100,

B10101,

B01110,

B11111,

B01110,

B10101,

B00100,

B00000,

};

byte fan\_off[8]={ // пользовательский символ отключения вентилятора

B00100,

B00100,

B00100,

B11111,

B00100,

B00100,

B00100,

B00000,

};

const int SPEAKER=8;

const int DOWN\_B=10; // кнопка понижения пороговой температуры

const int UP\_B=9; // кнопка повышения пороговой температуры

const int FAN=11; // вентилятор

int set\_temp=23; // исходное значение пороговой температуры=23 град.

boolean one\_time=false;

int val; // текущее значение температуры

void setup()

{

pinMode(FAN,OUTPUT);

pinMode

pinMode(UP\_B,INPUT);

pinMode(SPEAKER,OUTPUT);

lcd.begin(16,2); // настройка экземпляра дисплея - число столбцов и //строк

lcd.createChar(0,degree); // определение кодов пользовательских //символов:

lcd.createChar(1,fan\_off); // 0 - градус; 1-вент. откл.; 2- вент. вкл.

lcd.createChar(2,fan\_on);

lcd.setCursor(0,0); // вывод на экран ЖКИ постоянных надписей

lcd.print("Current:");

lcd.setCursor(10,0);

lcd.write((byte)0);

lcd.setCursor(11,0);

lcd.print("C");

lcd.setCursor(0,1);

lcd.print("Set:");

lcd.setCursor(10,1);

lcd.write((byte)0);

lcd.setCursor(11,1);

lcd.print("C");

lcd.setCursor(15,1);

lcd.write(1); // вывод на экран символа "вент. откл"

}

void loop()

{

val=analogRead(0); // чтение датчика температуры - см. проект 2.2

val=map(val,102,1023,0,450); // преобразование показаний датчика из //милливольт в градусы

lcd.setCursor(8,0);

lcd.print(val); // печать текущего значения температуры

if(digitalRead(DOWN\_B)==HIGH) // если нажата кнопка DOWN, то

{

set\_temp--; // значение пороговой температуры будет //уменьшаться

delay(200); // по одному градусу за 0,2 сек

}

if(digitalRead(UP\_B)==HIGH) // если нажата кнопка UP, то

{

set\_temp++; // значение пороговой температуры будет //увеличиваться

delay(200); // по одному градусу за 0,2 сек

}

lcd.setCursor(8,1);

lcd.print(set\_temp);

if(val>set\_temp) // если текущее значение температуры выше //порогового,

{

if(!one\_time)

{

tone(SPEAKER,400); // то однократно включить звуковой //сигнал 400 Гц на 0,5 сек

delay(500);

one\_time=true;

}

else

{

noTone(SPEAKER);

}

digitalWrite(FAN,HIGH); // включить вентилятор

lcd.setCursor(15,1);

lcd.write(2); // и поменять на экране символ состояния //вентилятора

} // на "вент.вкл"

else

{

noTone(SPEAKER); // иначе, отключить звук,

one\_time=false;

digitalWrite(FAN,LOW); // отключить вентилятор

lcd.setCursor(15,1);

lcd.write(1); // и поменять на экране символ состояния //вентилятора

} // на "вент. откл"

}

На самом деле построенную систему нельзя назвать автоматической потому, что виртуальное включение вентилятора не будет приводить к понижению температуры окружающего воздуха, и этот процесс надо будет имитировать вручную, изменяя посредством ползункового регулятора датчика текущую температуру, а с помощью кнопок UP и DOWN – значение пороговой температуры.

Отладка программы должна включать в себя проверку работы регулятора в широком диапазоне температур, где будет наступать превышение текущей температуры над пороговой. И это исследование следует произвести, по крайней мере, в диапазоне от +5 до +50 град.С, и отметить работоспособность программы или отклонения от нормальной работы или значений параметров.

**Проект 7.2 Бойлер**

В этом проекте предстоит моделирование полностью автоматической системы управления нагреванием воды. Система будет включать в себя сервопривод для моделирования уровня воды в баке (от 0 – пустой бак, до 180 град. – полный бак) и RB- светодиод для моделирования состояния (наличия и температуры) воды в баке.

Система «Бойлер» должна осуществлять автоматическую работу в следующих режимах:

1. Набор холодной воды (с температурой 10 градусов).
2. Нагрев воды – от 10 градусов до температуры, задаваемой на потенциометре.
3. Слив полного бака горячей воды.

Далее режимы повторяются.

В режиме набора воды синий светодиод плавно усиливает свечение; в режиме нагрева свечение синего светодиода плавно уменьшается, при этом усиливается свечение красного; в режиме слива плавно уменьшается свечение красного светодиода.

При смене режима на монитор последовательного порта отправляется сообщение, содержащее название начавшегося режима, и выдается кратковременный звуковой сигнал. В ходе режима нагрева на монитор периодически выводится достигнутое значение температуры воды вплоть до перехода в режим слива при достижении заданной температуры нагрева.

В процессе работы системы возможно регулирование температуры нагрева воды в диапазоне от 40 до 90 градусов с помощью потенциометра.

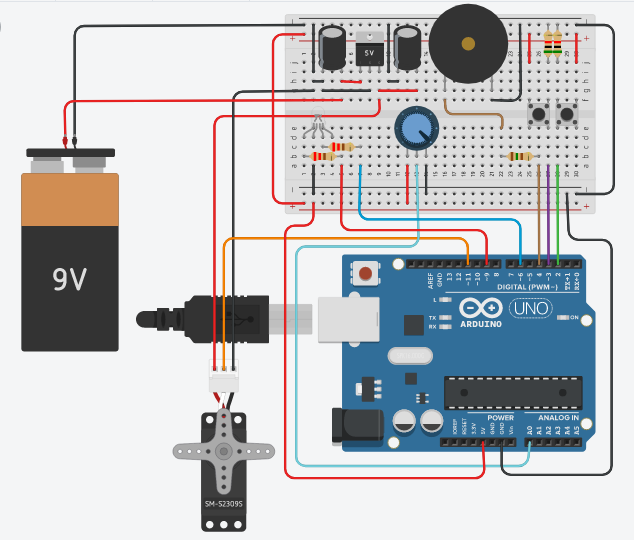
Для начального запуска автоматической системы и для остановки ее работы должны бить предусмотрены кнопки ПУСК и СТОП.

Таким образом, в макет системы «Бойлер» должны входить следующие компоненты:

1. Сервопривод с автономной системой питания (батарейка 9v, стабилизатор напряжения на 5v, два полярных конденсатора по 10мкФ), подключаемый к порту вывода с ШИМ – порту 11.
2. RGB-светодиод, у которого используются только 2 канала – R и B, а также 2 ограничивающих резистора по 220 Ом. Для подключения светодиода к микроконтроллеру также используются выходы ШИМ – порты 6 (канал В) и 9 ( канал R).
3. Потенциометр для регулировки температуры нагрева номиналом 1кОм, подключаемый к порту A0.
4. Пьезоэлемент для формирования звуковых сигналов при сменах режимов работы с ограничивающим резистором 150 Ом, подключаемый к порту 4
5. Кнопки ПУСК и СТОП, подключаемые соответственно к портам 2 и 3

с резисторами 5 кОм, подтягивающими к нулю.

Монтажная схема автоматической системы «Бойлер» будет выглядеть следующим образом:



Программа проекта с комментариями приведена ниже:

// BOYLER

#include <Servo.h> // подключение библиотеки

const int BLUE=6; // для RGB- светодиода два порта с ШИМ

const int RED=9;

const int SERVO=11; // для сервопривода – свой порт с ШИМ

const int PUSK=2; // кнопка PUSK будет на схеме справа

const int STOP=3; // кнопка STOP - слева

const int ZVUK=4; // пьезоэлемент к порту 4

const int T\_MAX=A0; // потенциометр для задания T\_MAX –

// к 0 аналоговому входу

Servo myServo; // создается объект с именем myServo

int val=0;

int svB=0; // переменная сигнала ШИМ для синего LED

int svR=0; // переменная сигнала ШИМ для красного LED

int i;

int tempV=10; // текущее значение температуры воды

int tempC=50; // заданная температура нагрева воды

boolean Work=false; // логическая переменная, отражающая

// состояние системы

void setup()

{

Serial.begin(9600); //

myServo.attach(SERVO);

myServo.write(0); // установка сервопривода в исходное

pinMode(RED,OUTPUT);

pinMode(BLUE, OUTPUT);

pinMode(PUSK, INPUT);

pinMode(STOP, INPUT);

pinMode(ZVUK, OUTPUT);

}

void loop()

{

if(digitalRead(STOP)==HIGH) // команды управления

Work=false; // пуском-остановом системы

if(Work==true)

{

Serial.println(" nabor vody"); // название этапа набора

tone (ZVUK, 500, 100); // звуковой сигнал

for (val=0; val<180; val=val+3) // цикл набора воды с поворотом

{ // ротора серводвигателя на 180 поЧС

analogWrite(RED, svR); // через 3 градуса

myServo.write(val);

delay(150);

analogWrite(BLUE,svB);

svB=svB+4; // повышение интенсивности синего

} // по ходу набора воды

tempV=10; // перед нагревом Т=10 град

svB=255;

svR=0;

Serial.println(" nagrev vody");

tone (ZVUK, 400, 100);

tempC=analogRead(T\_MAX); // чтение с потенциометра

tempC=map(tempC,0,1023,50,90); // перевод макс Т в градусы

for(i=255; i>=0; i=i-12)

{

analogWrite(BLUE,svB); // изменение засветки синего и красного

analogWrite(RED,svR); // LED в процессе нагрева воды

svR=svR+12;

svB=svB-12;

delay(200);

tempV=map(svR, 0, 255, 10,tempC); // перевод текущей температуры

Serial.print(tempV); // в градусы и печать значения

Serial.print(" "); // через последовательный порт

}

Serial.println(tempC);

svB=0;

svR=255;

analogWrite(BLUE, svB);

Serial.println(" sliv vody");

tone (ZVUK, 300, 100);

for(val=180; val>=0; val=val-3) // в режиме слова цикл поворота

{ // ротора серводвигателя

myServo.write(val); // шагами по 3 градуса противЧС

delay(150);

analogWrite(RED,svR); // с затуханием красного LED

svR=svR-4;

}

svR=0;

tempV=10;

}

else

if(digitalRead(PUSK)==HIGH)

Work=true;

}

Не забывайте, что после запуска моделирования программы сначала следует открыть и очистить монитор последовательного порта, а затем произвести запуск системы: для запуска надо нажать кнопку PUSK (это правая кнопка), придерживая ее до появления звукового сигнала, и отпустить ее. В процесс работы наблюдать за серводвигателем, RGB- светодиодом и за информацией, выводимой на монитор. Не останавливая работы, изменять температуру нагрева воды с помощью потенциометра. Подробно описать работу системы.

**Проект 7.3 Учебный стенд – тренажер**

В этом проекте создан стенд – тренажер, который, в принципе, является универсальным, способным контролировать знания и навыки в различных областях техники и по множеству тестов. В конкретной реализации стенд обеспечивает прохождение одного варианта теста из 15 заданий по тематике арифметических и логических операций с двоичными кодами. В каждом задании на экран ЖКИ выводятся два 4-разрядных двоичных кода, а также вопрос, требующий оценить результат заданной операции, на который требуется ответ «Да» или «Нет» в течение 10 секунд.

Ответ вводится нажатием одной из двух кнопок, оценивается стендом. Правильные и ошибочные ответы подсчитываются, их количества выводятся на экран. Просроченные ответы не подсчитываются, но их число в конце теста легко определяется, так как общее число заданий теста – 15.

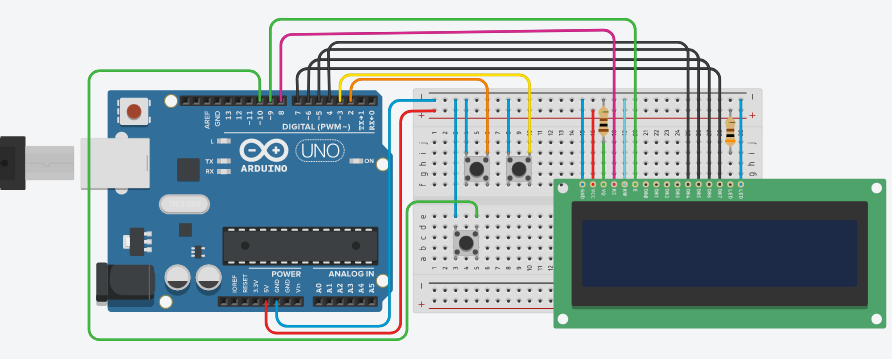
Аппаратный состав стенда-тренажера минимален – Микроконтроллер, ЖКИ, кнопки «Да» и «Нет», кнопка «PUSK».

Особенности программного обеспечения стенда связаны, во-первых, с тем, что текст заданий теста не помещается в оперативной памяти, а должен быть размещен во флэш-памяти данных, и для операций с флэш-памятью необходимо использовать библиотеку EEPROM.h.

Во-вторых, для фиксации нажатия кнопок «Да» и «Нет» используется система внешних прерываний, в которой используются порты 2 и 3.

В третьих, для этих кнопок, а также для кнопки «PUSK» используется подтягивание к напряжению питания +5v через внутренние подтягивающие резисторы.

Монтажная схема стенда-тренажера выглядит следующим образом:



Программа этой микропроцессорной системы приведена ниже:

#include <LiquidCrystal.h> // подключение библиотеки ЖКИ

#include <EEPROM.h> // подключение библиотеки флэш-памяти данных

// далее коды массива теста

unsigned char AB[17]={0x79, 0xa6, 0x59, 0xc7, 0xe9, 0xbe,

0x3c, 0xd2, 0x4e, 0x27, 0xfa, 0x6f, 0xed,0x16,0xb7, 0x5e,0x6e};

unsigned char USL [15][6] = {"A-B=2", "B%3=0", "B-A=4", "A=0xC",

"B=0x9", "B-A=2", "B-A=9", "A>13", "B%4=0", "A+B=9", "B=0xA",

"B-A=9", "A=0xD", "B-A=5", "A%3=2"};

LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // подключение ЖКИ

int address=0;

int PUSK=10; // порт 10 для кнопки PUSK

unsigned char A;

int schetPRAV=0; // счетчик правильных ответов

int schetOSHIB=0; // счетчик ошибочных ответов

int timer=10;

boolean otvet;

boolean Work=false;

int zadanie;

void setup()

{

pinMode(2, INPUT); // порты 2 и 3 будут использоваться

digitalWrite(2, HIGH); // для прерываний от кнопок ДА и НЕТ

pinMode(3, INPUT); // при этом кнопки подтянуты к +5v

digitalWrite(3, HIGH); // внутренними резисторами МК

pinMode(PUSK, INPUT);

digitalWrite(PUSK, HIGH);

lcd.begin(16,2); // размещение массивов задания

for (int i=0; i<17; i++) // во флэш-памяти

{

EEPROM.update(address, AB[i]);

address++;

}

for (int i=0; i<15; i++)

{

for (int j=0; j<6; j++)

{

EEPROM.update(address, USL[i][j]);

address++;

}

}

attachInterrupt(0, da, FALLING); // объявление программы обработки прерывания

// от кнопки ДА

attachInterrupt(1, net, FALLING); // и от кнопки НЕТ

}

void da() // сама программа обработки прерывания от кнопки ДА

{

if (zadanie>7)

{

otvet=bitRead(AB[16], zadanie-8);

}

else otvet=bitRead(AB[15], zadanie); // читает бит правильного ответа

// (1-да, 0-нет)

if(otvet==1) // если правильным ответом был ДА,

{

schetPRAV++; // то +1 в счетчик правильных ответов

}

else

{

schetOSHIB++; // если НЕТ, то +1 в счетчик ошибок

}

timer=0;

}

void net() // программа обработки прерываний от кнопки НЕТ

{

if (zadanie>7)

{

otvet=bitRead(AB[16], zadanie-8);

}

else otvet=bitRead(AB[15], zadanie);

if(otvet==0) // а здесь правильность ответа обратная

{

schetPRAV++;

}

else

{

schetOSHIB++;

}

timer=0;

}

void loop() цикл решения всего теста

{

if(digitalRead(PUSK)==LOW) // начало теста после нажатия кнопки PUSK

Work=true;

if(Work==true)

{

schetPRAV=0;

schetOSHIB=0;

for (zadanie=0; zadanie<15; zadanie++)

{

timer=10; // настройка таймера на 10 сек

address=zadanie; // вывод на экран данных условия

A=EEPROM.read(address);

lcd.setCursor(0,0);

lcd.print("A=");

lcd.setCursor(2,0);

lcd.print(A/0x10, BIN);

lcd.setCursor(0,1);

lcd.print("B=");

lcd.setCursor(2,1);

lcd.print(A&0x0f, BIN);

lcd.setCursor(7,0); // и вывод вопроса

address=zadanie\*6+17;

for(int j=0; j<5; j++)

{

lcd.print(char(EEPROM.read(address)));

address++;

}

lcd.setCursor(8,1); // вывод текущего состояния счетчиков правильных

lcd.print(schetPRAV); // и ошибочных ответов

lcd.print("-");

lcd.print(schetOSHIB);

lcd.print(" ");

lcd.setCursor(14,1); // вывод показаний таймера

lcd.print(timer);

while (timer>0) // обратный отсчет таймера

{

delay(1000);

if(timer==0)break;

timer--;

lcd.setCursor(14,1);

lcd.print(timer);

lcd.setCursor(15,1);

lcd.print(" ");

}

lcd.clear();

}

Work=false; // после окончания всего теста

lcd.setCursor(8,1); // на экране показания счетчиков

lcd.print(schetPRAV); // правильных и ошибочных ответов

lcd.print("-");

lcd.print(schetOSHIB);

}

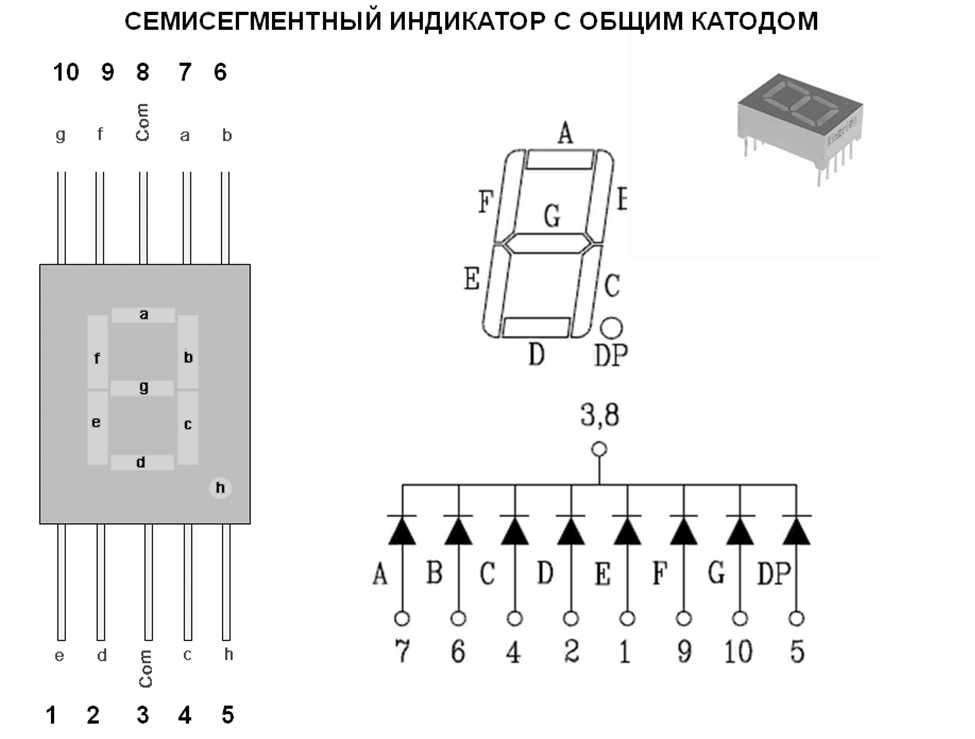
}

При работе с этой программой после начала моделирования надо не забыть нажать кнопку «PUSK», после чего запустится процесс тестирования. В процессе теста следует для ответа нажимать либо кнопку «Да» (левая) или кнопку «Нет» (правая), а после ответа на все 15 заданий выполнение теста завершится, а на экране останутся количества правильных и ошибочных ответов. Повторный запуск теста можно произвести кнопкой «PUSK».

**Проекты раздела 8. Динамический принцип индикации**

**Проект 8.1 Секундомер на 7-сегментных индикаторах**

Семисегментные индикаторы широко распространены среди цифровых индикаторов так как позволяют получить достаточно удобочитаемые изображения десятичных цифр в виде комбинации из семи подсвечиваемых сегментов. В микроконтроллерной технике широко используются светодиодные 7-сегментные индикаторы, построенные по схеме либо с общим катодом, либо с общим анодом. В элементной базе эмулятора Arduino Tinkercad используются индикаторы с общим катодом. Их принципиальная схема, обозначение сегментов и расположение выводов показаны на рисунке:



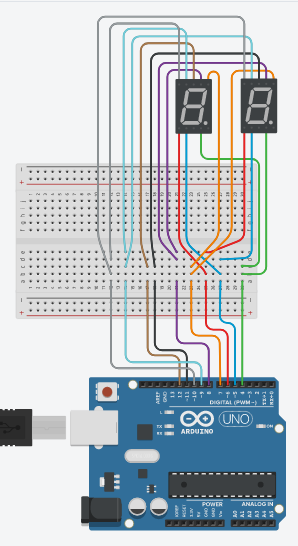
Для засветки изображения какой-либо цифры необходимо на аноды сегментов A … G подать соответствующую комбинацию нулей и единиц (1-горит, 0- не горит), а на общий катод подать уровень логического нуля.

Подключение такого индикатора к микроконтроллеру обычно не требует токоограничивающих резисторов и производится просто. Для непосредственного подключения приходится использовать 8 или 9 портов вывода (7 сегментов (или 8, если используется еще точка) и один – общий).

Если же 7-сегментные индикаторы использовать для индикации многоразрядных десятичных чисел, то требуемое количество используемых портов может оказаться непомерно большим.

Решение этой проблемы было найдено с помощью принципа динамической индикации. Идея заключается в том, что одноименные сегменты индикаторов всех разрядов соединяются между собой, и поэтому код изображения цифры будет одновременно поступать на индикаторы всех разрядов. Однако для подсветки изображения цифры необходимо еще подать логический ноль на общий катод индикатора, а ноль подается только на один из общих катодов и поэтому изображение цифры появится только на одном из индикаторов. Для изображения многоразрядного числа цифры из его разрядов подаются на индикатор поочередно, и при этом поочередно подается логический ноль и на общие катоды соответствующих разрядов. Получается, что цифры числа засвечиваются в своих разрядах, но по очереди.

Если увеличить частоту этого процесса, то инерционность нашего зрения позволит воспринимать свечение цифр всех разрядов как одновременное и непрерывное.



На этом принципе мы построим схему секундомера, который будет отсчитывать секунды от 00 до 59 на двух 7-сегментных индикаторах.

К портам 4,5,6,7,8,9,10 подключены одноименные сегменты обоих индикаторов c,d,e,b,a,f,g соответственно. К порту 11 подключен общий младщего разряда, к порту 12 – общий старшего разряда.

Листинг программы приведен ниже:

// Секундомер

const int seg\_a=8;

const int seg\_b=7;

const int seg\_c=4;

const int seg\_d=5;

const int seg\_e=6;

const int seg\_f=9;

const int seg\_g=10;

const int L\_com=11; // общий младшего разряда

const int H\_com=12; // общий старшего разряда

unsigned char secund=0;

unsigned char ds, es, Z, N;

unsigned char MC=0; // подсчет количества циклов частоты 50Гц

unsigned char cifry[10]= {B11111100, //abcdefg. 0

B01100000, // 1

B11011010, // 2

B11110010, // 3

B01100110, // 4

B10110110, // 5

B10111110, // 6

B11100000, // 7

B11111110, // 8

B11110110}; // 9

void setup() {

pinMode(seg\_a,OUTPUT);

pinMode(seg\_b,OUTPUT);

pinMode(seg\_c,OUTPUT);

pinMode(seg\_d,OUTPUT);

pinMode(seg\_e,OUTPUT);

pinMode(seg\_f,OUTPUT);

pinMode(seg\_g,OUTPUT);

pinMode(L\_com,OUTPUT);

pinMode(H\_com,OUTPUT);

}

void OTOBR(unsigned char Z)

{

N=cifry[Z];

digitalWrite(seg\_a,N&0x80); //a- выделение разрядов из кода N

digitalWrite(seg\_b,N&0x40); //b

digitalWrite(seg\_c,N&0x20); //c

digitalWrite(seg\_d,N&0x10); //d

digitalWrite(seg\_e,N&0x08); //e

digitalWrite(seg\_f,N&0x04); //f

digitalWrite(seg\_g,N&0x02); //g

}

void loop()

{

digitalWrite(L\_com,LOW); //0-активный-единицы секунд

digitalWrite(H\_com,HIGH); //1-пассивная-десятки секунд

Z=secund%10; // младшая цифра

OTOBR(Z);

delay(10); // первая половина периода

digitalWrite(L\_com,HIGH); //а теперь наоборот

digitalWrite(H\_com,LOW);

Z=secund/10; // старшая цифра

OTOBR(Z);

MC++;

delay(10); // вторая половина периода

if(MC==50)

{MC=0;

if(secund==59)secund=0;

else secund++;}

}

**Проект 8.2 Шестнадцатеричный секундомер**

В этом проекте Вам предстоит создать устройство, которое тоже является секундомером, но ведет отсчет секунд в шестнадцатеричном коде. Оказывается, что на 7-сегментном индикаторе можно формировать и изображения символов от A до F, ну и кроме того Вам известно, что

59 = 0x3b.

**Раздел 10. Работа с библиотеками Arduino**

Даже небольшой опыт работы с Arduino обязательно связан с понятием библиотек. И это не случайно. Очень часто одни и те же части кода кочуют из программы в программу. Например, код для работы с датчиком. Чтобы не писать этот код каждый раз заново, его выносят в отдельные файлы — библиотеки. Огромное количество готового кода уже написано другими людьми, и с помощью библиотек его можно легко использовать в своих программах.

В процессе изучения микропроцессорных систем на базе микроконтроллеров семейства MCS-51 при знакомстве с жидкокристаллическим дисплеем мы с вами создали свою микробиблиотеку из подпрограмм Delay, LCD\_SEND, LCD\_INIT, которые обязательно подключали к любой программе, в которой используется ЖКИ. Изучая Arduino, вы познакомились с библиотекой LiquidCrystal.h, позволяющей легко программировать множество функций ЖКИ: установку курсора в заданную позицию, вывод теста на экран (даже не надо этот текст кодировать самим), вывод на экран многоразрядных чисел (а мы писали специальную подпрограмму N\_to\_LCD) и т.п.

Каждая библиотека, в конечном счете, - это набор подпрограмм. Библиотеки Arduino – это наборы подпрограмм, позволяющих даже начинающему пользователю реализовывать все необходимые функции подключаемого устройства и без глубокого понимания того, как это делается. Такие библиотеки создают комфортную среду разработки. Библиотека последовательного порта Serial.h оказалась настолько удобной (в том числе и при отладке программ), что ее сделали внутренней библиотекой (то есть она подключена всегда по умолчанию), другие библиотеки подключаются пользователем по мере необходимости командами #include.

Необходимость в библиотеке чаще всего возникает при подключении в микропроцессорную систему нового устройства. В проекте 10.1 мы будем подключать а Arduino модули NeoPixel, состоящие из адресных светодиодов.

Адресный светодиод представляет собой RGB-светодиод, изготовленный вместе со схемой управления (драйвером). Каждый адресный светодиод имеет 4 провода для подключения **: gnd, vcc, in, out.** В модуле светодиоды соединены в ленту, причем к линиям питания и земли – параллельно, а по линиям **in** и **out** последовательно. В результате и модуль имеет одну входную линию **in** и одну выходную линию **out**. В модуле каждый светодиод имеет свой адрес по положению в цепочке от входа модуля.

Структура модуля **NeoPixel** показана на рисунке.



Поток информации для управления всеми светодиодами в последовательном коде посылается по единственному проводу. Каждый драйвер, зная свой адрес, выхватывает из этого потока только 3 байта, относящиеся к его светодиоду (R,G,B), а остальную информацию просто пропускает дальше. В результате каждый адресный светодиод может управляться независимо от других. Вы можете к выходу **out** одного модуля подключить вход **in** другого модуля – тогда у них будет сквозная адресация всех светодиодов.

Очевидно, что управление такими модулями требует необычных программ, а, следовательно, требуется библиотека. В эмуляторе Tinkercad используется библиотека **Adafruit Neo Pixel**.

Мы рассмотрим вопрос поиска нужной библиотеки, ее установки, использования удобного сервиса в виде примеров применения библиотеки, знакомства с командами библиотеки и их модификации для своего конкретного применения в том виде, в каком он обычно встречается на практике.

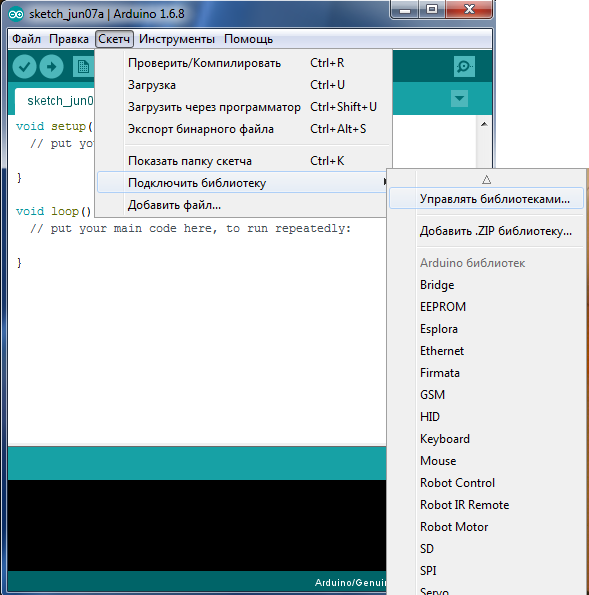
Для этого , в первую очередь, потребуется на своем компьютере установить среду Arduino IDE, если она у Вас еще не установлена. Существует много вариантов такой установки, например, на сайте **all-arduino.ru,** используя ссылку https://all-arduino.ru/arduino-ide/.

**Проект 10.1. Контроллер светодиодных адресных лент**

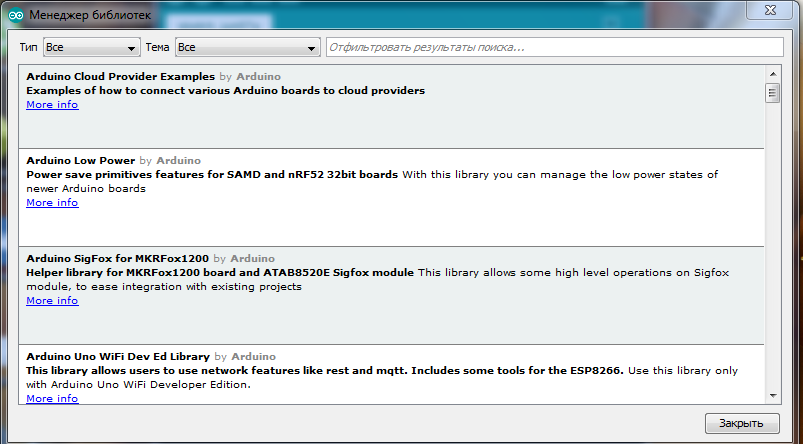
Действия по реализации проекта начнем с установки библиотеки и выбора нужной программы примера. Для этого потребуется:

1. В окне скетча Arduino выбрать в меню:

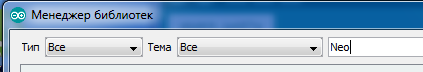
**Скетч / Подключить библиотеку / Управлять библиотеками**



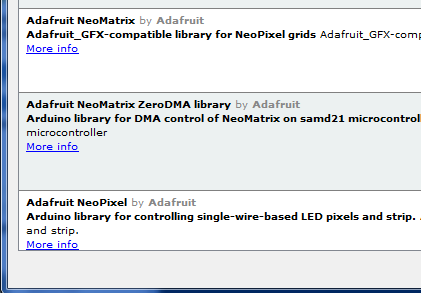
1. Через несколько секунд появится окно Менеджера библиотек:



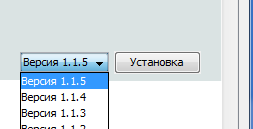
1. В окне «Отфильтровать результаты поиска» ввести ключевое слово для поиска нашей библиотеки **Neo**. Само это слово при наборе не сразу появится в окне, а вместе с результатами поиска:



1. Среди найденных библиотек найти **Adafruit Neo Pixel** и один раз щелкнуть на блоке этой библиотеки:



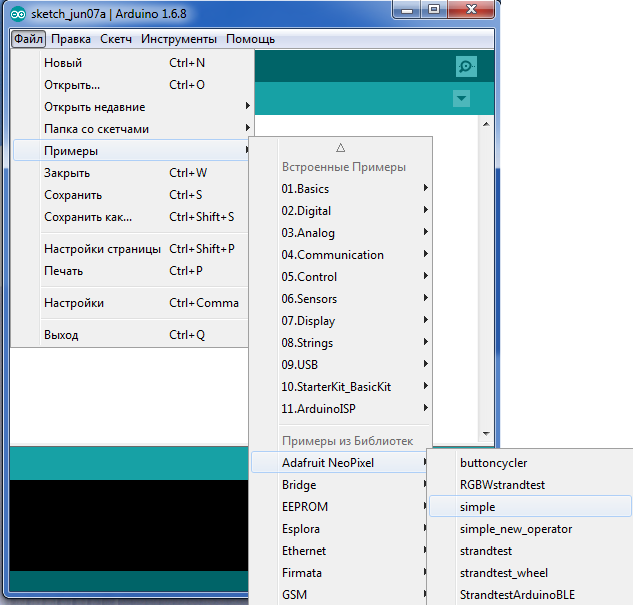
1. Появятся кнопки, которые позволят выбрать версию библиотеки (выбирайте последнюю) и произвести установку библиотеки. Сделайте это.



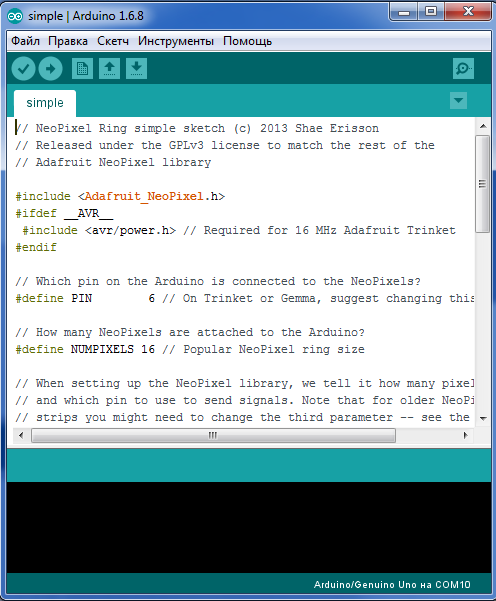
1. Теперь, закрыв окно менеджера библиотек, можно начать работу с самой библиотекой. И лучше всего начать со встроенных примеров:

Из окна скетч Arduino выбрать в меню:

**Файл/ Примеры /**Примеры из библиотек**:Adafruit NeoPixel/ simple**



1. Программа примера **simple** будет открыта в новом окне скетч Arduino:



1. Далее начинается работа по адаптации примера к конкретному заданию проекта в следующем порядке:
   1. С программой примера необходимо разобраться – понять, какие функции библиотеки используются, какие параметры придется задавать.
   2. Произвести настройку программы на следующие условия задания:

- используется модуль **NeoPixel Ring 12**.

**-** подключение сигнального провода модуля производить к порту 7;

**-** засвечивать все светодиоды синим цветом.

8.3. Все изменения программы производить в окне скетча Arduino.

8.4. Произвести проверку программы: **Скетч / Проверить и компилировать**

8.5. Если в программе ошибок нет, то она готова к переносу в симулятор Tinkercad методом копирования и вставки

9. Собрать схему проекта в соответствии с условиями п. 8.2.

10. Скопировать программ в поле Код проекта.

11. Произвести моделирование.

12. Модифицировать программу таким образом, чтобы цвета всех светодиодов чередовались: синие с зелеными.

**Проект 10.2 NeoPixels. Полет фантазии.**

В этом проекте создается динамичная разноцветная система на Ваше усмотрение из любых модулей при соблюдении следующих условий:

-В проекте должно быть использовано не менее 3 модулей NeoPixel.

-Два модуля должны быть соединены в одну цепочку.

-Для управления модулями должно быть использовано не менее двух портов микроконтроллера.