Лабораторная работа №7

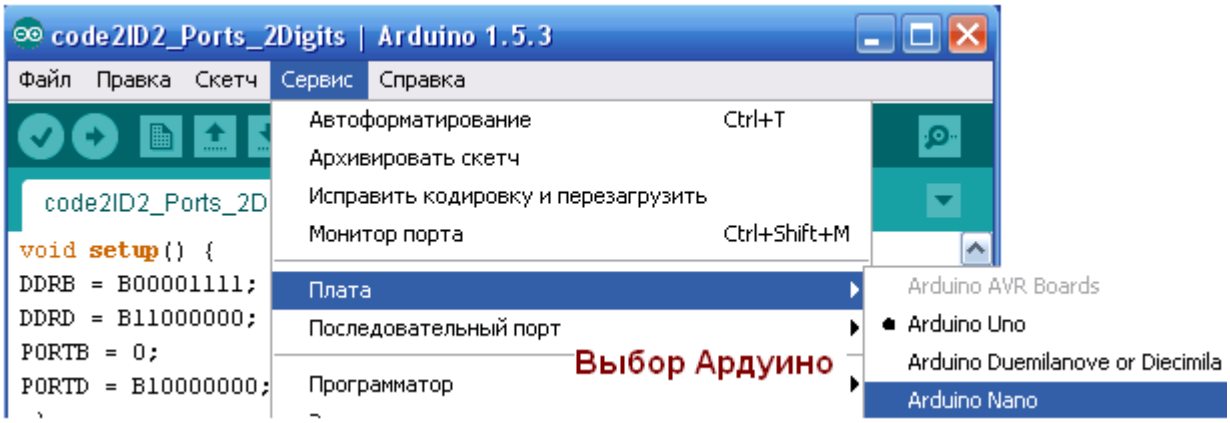
Тема: Программирование контроллера Ардуино. Схема «Бегущие огни».

Теоретическое введение: Описание Ардуино.

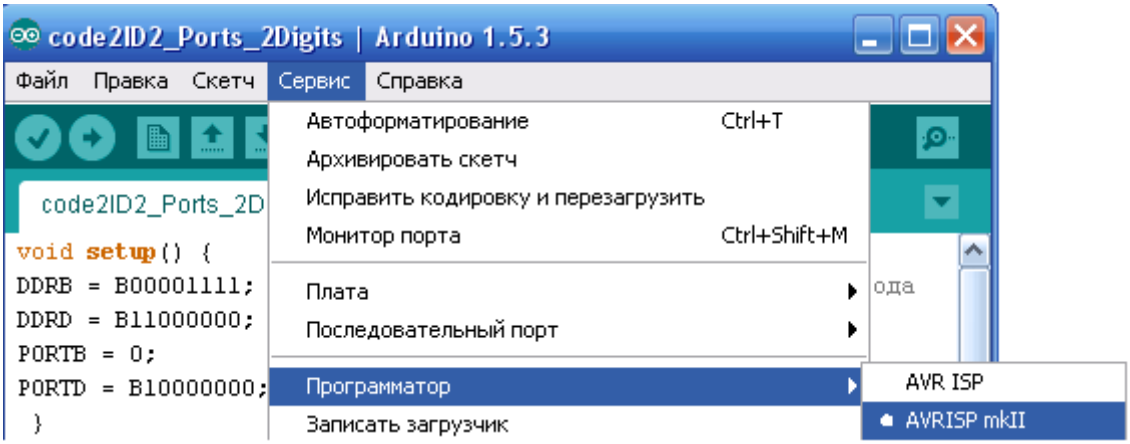
Справочник по программированию Ардуино.

Настройка среды Ардуино.

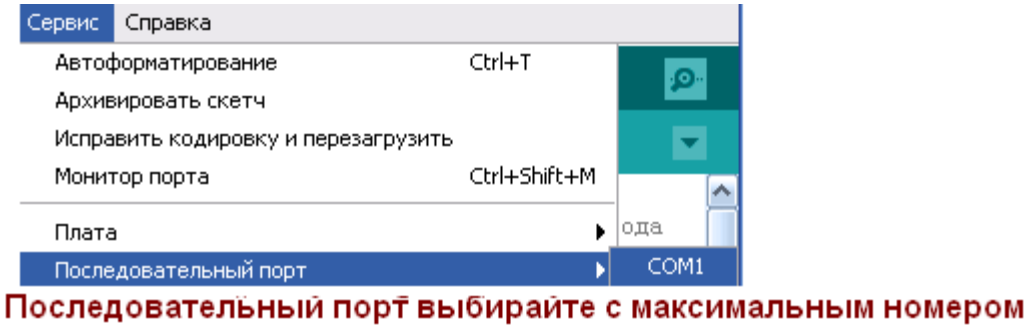
Выберите нужную плату (Uno или Nano):



При использовании USB-кабеля программатор оставьте дефолтный:



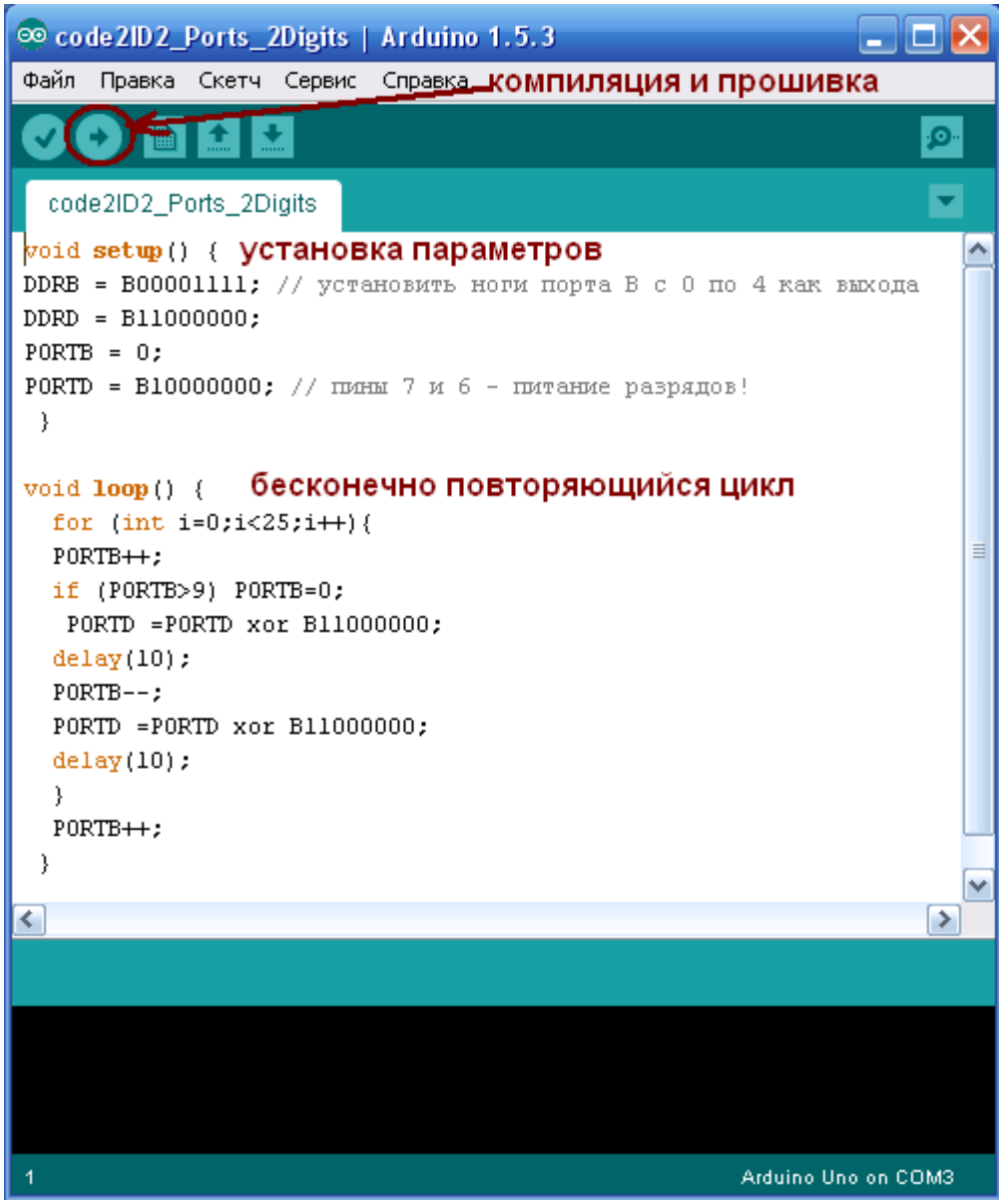
Выберите COM-порт:



Создание скетчей.

«Скетч» – программа для Ардуино – может начинаться с объявления глобальных переменных, и далее должен содержать (как минимум) процедуры Setup и Loop.

После ввода программного кода его можно проверить/компилировать кнопкой с галочкой или сразу компилировать/загрузить на флеш-память программ («прошить») микроконтроллера («сердцем» Uno и Nano является МК ATMega328):



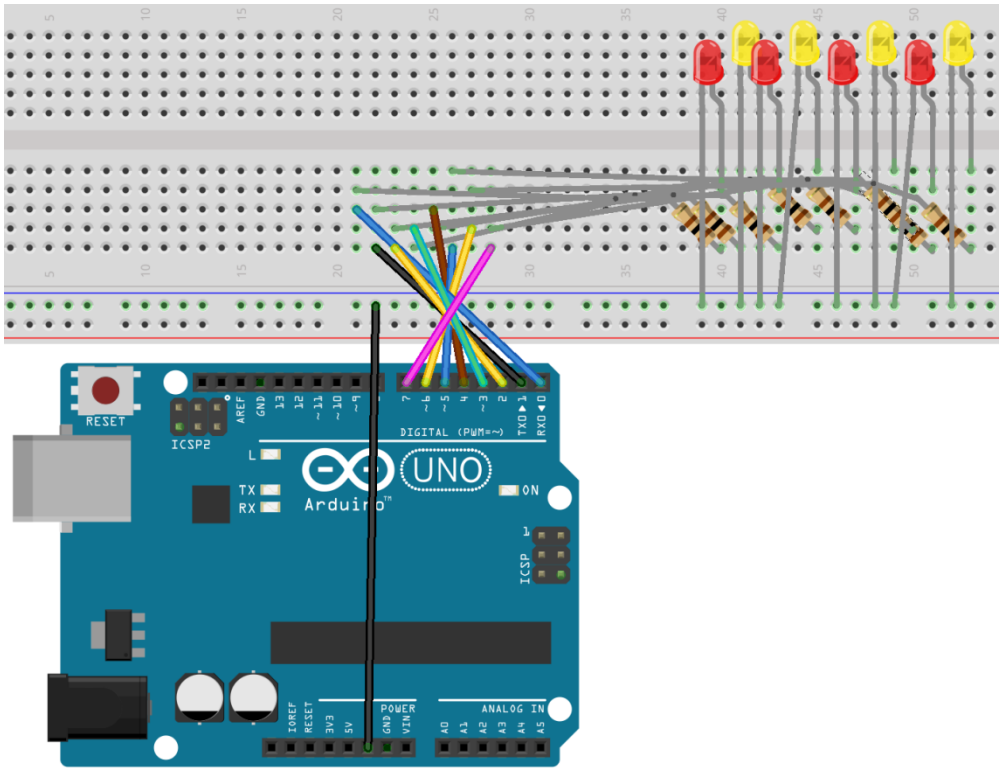
Задание 1. Бегущие огни (без дешифраторов)

Ознакомьтесь с программными основами работы с пинами (выводами) микроконтроллера как портами: «Побитовый сдвиг влево. Побитовая инверсия»

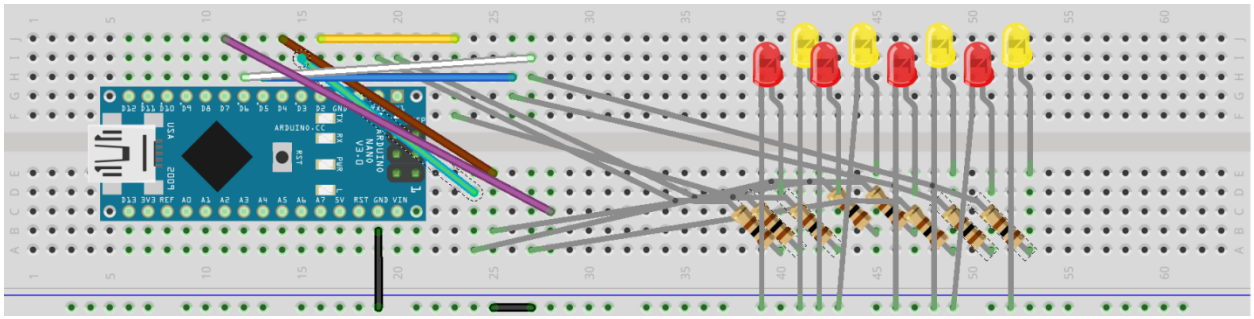
Порт D Ардуино – это 8 пинов с 0-го по 7-й, порт B – 6 пинов с 8-го по 13-й.

Для непосредственного «зажигания» светодиодов с выводов Ардуино пины от 0 до 7-го подключаем через резисторы к светодиодам (к их плюсам (анодам) – длинным ножкам, минусы (катоды) – на землю) (можно резисторы поставить между светодиодами и землёй, это не играет роли).

Возможное подключение см. ниже



В Ардуино Нано немного другая распиновка выводов – крайний справа пин (TX) – это D1, а второй справа (RX) – D0, третий и четвёртый выводы справа – Reset и Ground, и начиная с пятого идут от D2 до B5 (это пин D12, т.е. Digital12) , а пин B6=D13 – крайне левый внизу:



Пишем первую программу:

```
//программа 1 «Бегущие огни»
int led = 1; // глобальная переменная, хранящая номер «зажигаемого» вывода

void setup() {
  DDRD = 255; // все выводы порта D настраиваем как выходы
  }

void loop() {
    PORTD = led; // «зажигаются» соответствующие выводы порта D
    delay(300); // задержка в 0,3 сек.
    led = led<<1; // сдвиг влево – если горел 1-й светодиод, то следующим зажжётся второй
    if (led > 128) // 128 = B10000000
      led = 1; // после 8-го светодиода возвращаемся к первому
}
```

Задание 2. Бегущие огни с дешифратором 74HC42N

Как видим, при непосредственном подключении выводов Ардуино к светодиодам затрачивается неоправданно много пинов, так, если светодиодов 16, то выводов просто не хватит.

Для экономии выводов можно использовать дешифратор.

Воспользуемся уже знакомым двоично-десятичным (4 x 10) дешифратором 74HC42N:

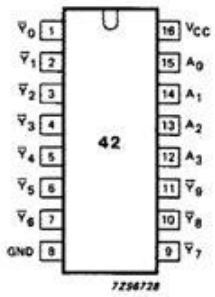


Fig.1 Pin configuration.

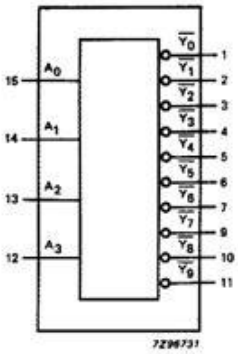
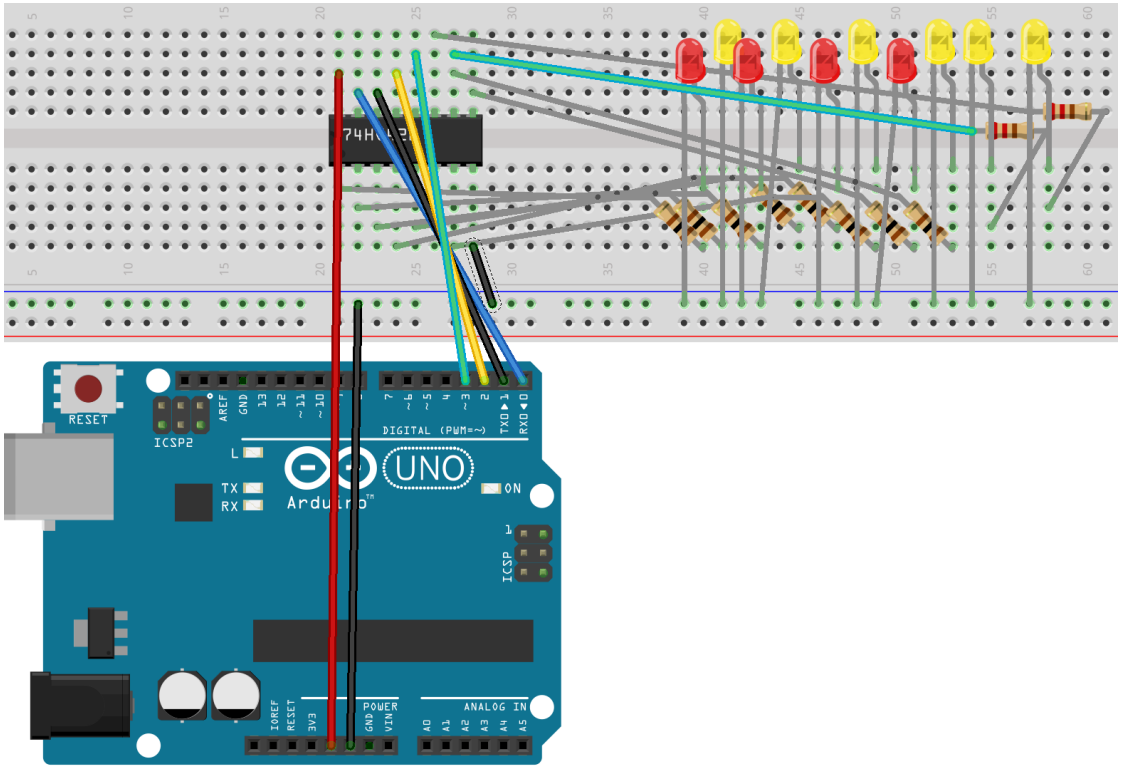


Fig.4 Functional diagram.

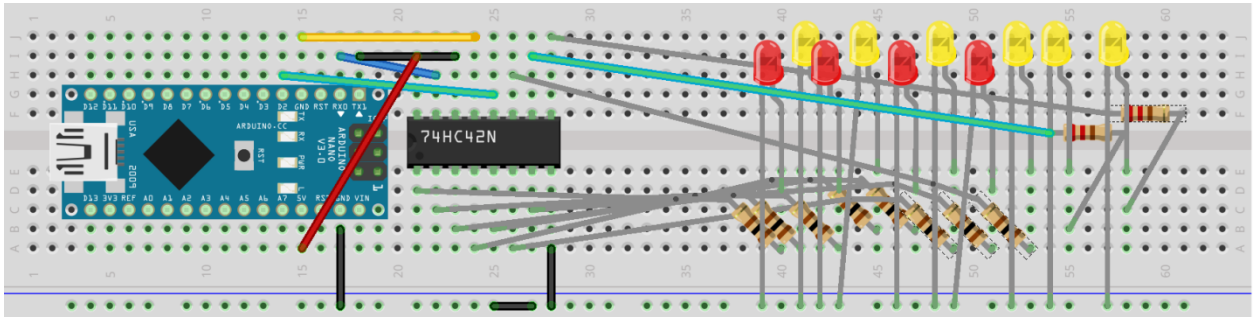
FUNCTION TABLE

INPUTS				OUTPUTS									
A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
L	L	L	L	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	H	H	L	H	H	H	H	H	H	H
L	L	H	H	H	H	H	L	H	H	H	H	H	H
L	H	L	L	H	H	H	H	L	H	H	H	H	H
L	H	L	H	H	H	H	H	H	L	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H
L	H	H	H	H	H	H	H	H	H	H	L	H	H
H	L	L	L	H	H	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L
H	L	H	L	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H
H	H	L	L	H	H	H	H	H	H	H	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H
H	H	H	L	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H

Соберите схему прмерно следующего вида (светодиоды, разумеется, можно сдвинуть поближе к дешифратору):



Нано-версия:



Теперь нам нужно не сдвигать активный выход, а +плюсовать значение led:

```
//программа 2 бегущего светодиода дешифратором 74HC42N
int led = 0;
void setup() {
  DDRD = 255;
  }

void loop() {
    PORTD = led;
    delay(300);
    led++;
    if (led > 9)
      led = 0;
}
```

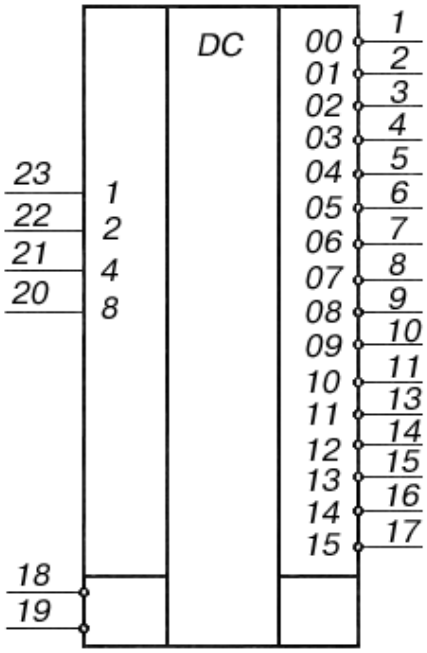
Задание 3. Бегущие огни с дешифратором ИД3

Теперь, для зажигания 16-ти светодиодов используем 4х16 дешифратор-демультиплексор К155ИД3.

К155ИД3 – дешифратор, позволяющий преобразовать четырехразрядный код, поступающий на входы A0-A3 в напряжение низкого логического уровня, появляющееся на одном из шестнадцати выходов 0-15. Дешифратор имеет два входа разрешения дешифрации E0 и E1. Эти входы можно использовать как логические, когда дешифратор ИД3 служит демультиплексором данных. Тогда входы A0-A3, используются как адресные, чтобы направить поток данных, принимаемых входами E0 или E1, на один из выходов 0-15. На второй, не используемый в этом включении вход E, следует подать напряжение низкого уровня.

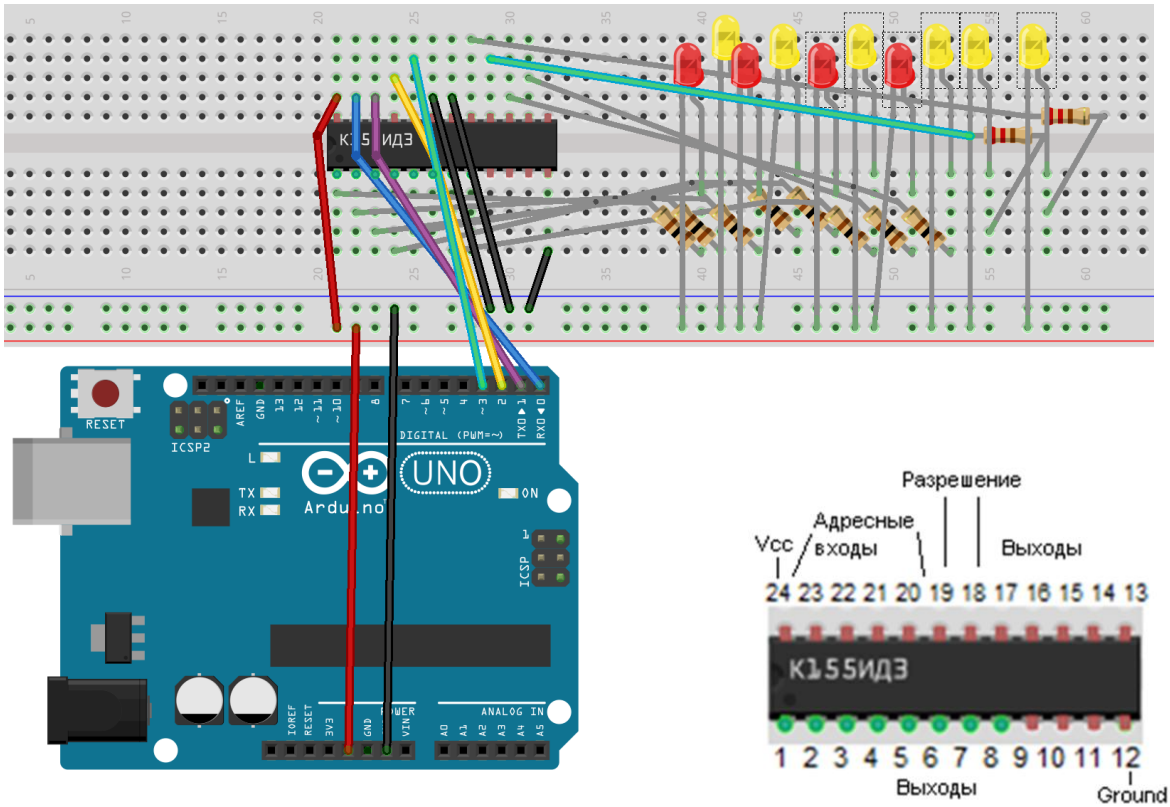
№ выв.	Назначение	№ выв.	Назначение
1	Выход 0	13	Выход 11
2	Выход 1	14	Выход 12
3	Выход 2	15	Выход 13
4	Выход 3	16	Выход 14
5	Выход 4	17	Выход 15
6	Выход 5	18	Вход стробирующий
7	Выход 6	19	Вход стробирующий
8	Выход 7	20	Вход информационный
9	Выход 8	21	Вход информационный
10	Выход 9	22	Вход информационный
11	Выход 10	23	Вход информационный
12	Общий	Ucc	24

По входам E0 и E1 даются сигналы разрешения выходов, чтобы устранять текущие выбросы, которыми сопровождается дешифрация кодов, появляющихся не строго синхронно (например, поступающих от счетчика пульсаций). Чтобы разрешить прохождение данных на выходы, на входы E0 и E1 следует дать напряжение низкого уровня. Эти входы необходимы также при наращивании числа разрядов дешифрируемого кода. Когда на входах E0 и E1 присутствуют напряжения высокого уровня, на выходах 0-15 появляются высокие уровни.



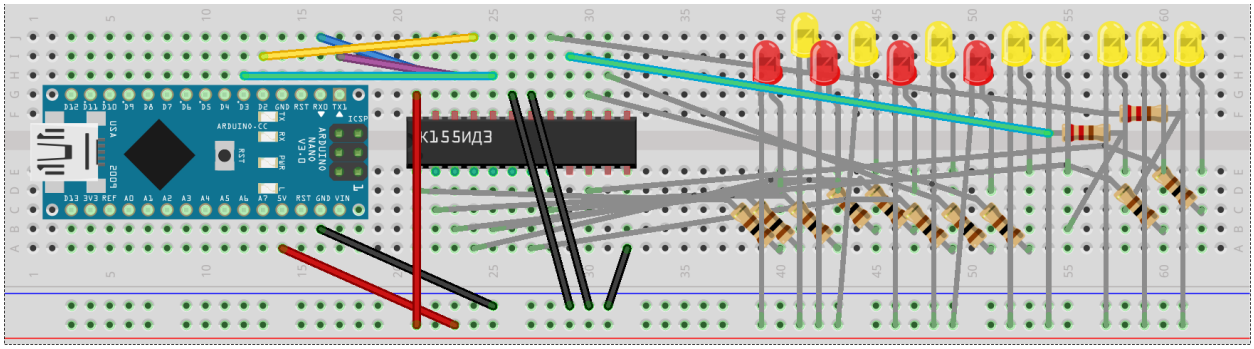
Вход						Выход													
$\overline{E0}$	$\overline{E1}$	A3	A2	A1	A0	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$	$\overline{8}$	$\overline{9}$	$\overline{10}$	$\overline{11}$	$\overline{12}$	$\overline{13}$	$\overline{14}$	$\overline{15}$
H	H	H	H	H	H	H	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
H	H	H	H	H	B	B	H	B	B	B	B	B	B	B	B	B	B	B	B	B	B
H	H	H	H	B	H	B	B	H	B	B	B	B	B	B	B	B	B	B	B	B	B
H	H	H	H	B	B	B	B	B	H	B	B	B	B	B	B	B	B	B	B	B	B
H	H	H	B	H	H	B	B	B	B	H	B	B	B	B	B	B	B	B	B	B	B
H	H	H	B	H	B	B	B	B	B	B	H	B	B	B	B	B	B	B	B	B	B
H	H	H	B	B	H	B	B	B	B	B	B	H	B	B	B	B	B	B	B	B	B
H	H	H	B	B	B	B	B	B	B	B	B	B	H	B	B	B	B	B	B	B	B
H	H	B	H	H	H	B	B	B	B	B	B	B	B	H	B	B	B	B	B	B	B
H	H	B	H	H	B	B	B	B	B	B	B	B	B	B	H	B	B	B	B	B	B
H	H	B	H	B	H	B	B	B	B	B	B	B	B	B	B	H	B	B	B	B	B
H	H	B	H	B	B	B	B	B	B	B	B	B	B	B	B	B	H	B	B	B	B
H	H	B	B	H	H	B	B	B	B	B	B	B	B	B	B	B	B	H	B	B	B
H	H	B	B	H	B	B	B	B	B	B	B	B	B	B	B	B	B	B	H	B	B
H	H	B	B	B	H	B	B	B	B	B	B	B	B	B	B	B	B	B	B	H	B
H	H	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	H
H	B	X	X	X	X	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
B	H	X	X	X	X	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
B	B	X	X	X	X	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B

Подсоедините пины 0–3 Ардуино ко входам 23–20 ИД3. Выходы ИД3 1–17 подсоедините к катодам (коротким ножкам) светодиодов, аноды – к линии питания (на картинке она первая снизу), входы 18 и 19 заземлите:



Скетч напишите по образцу программы задания 2. В проверке `if (led > x)` используйте имеющееся у вас количество светодиодов.

Нано-версия:

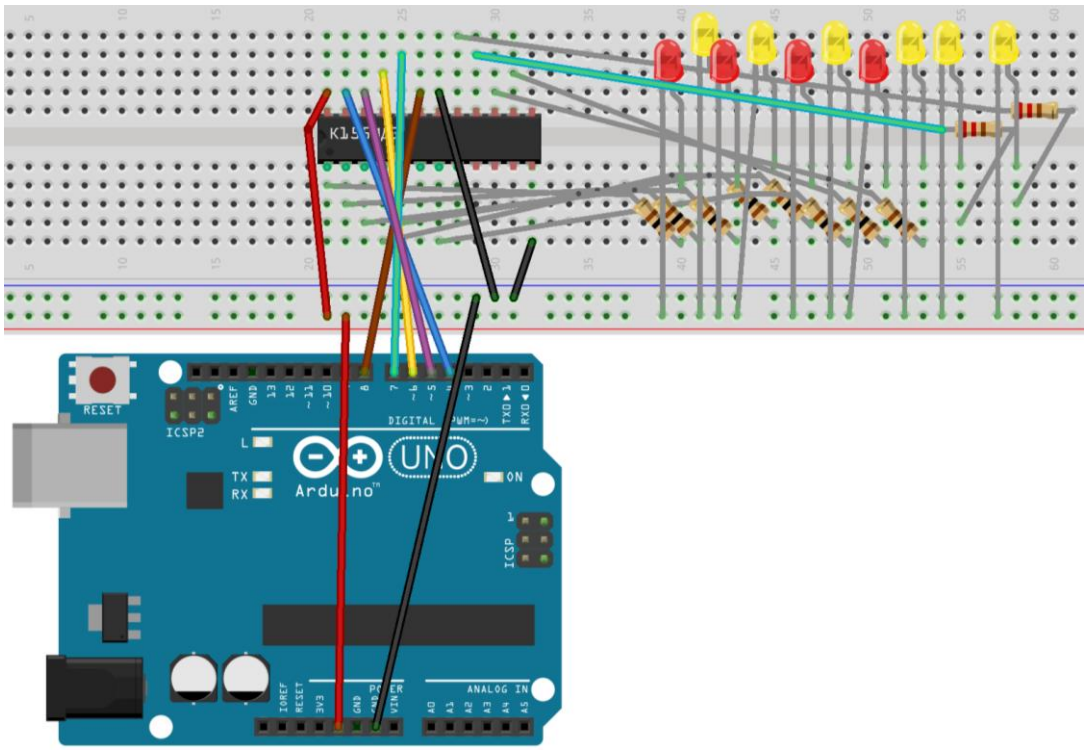


Задание 4. Демультиплексирование сигнала

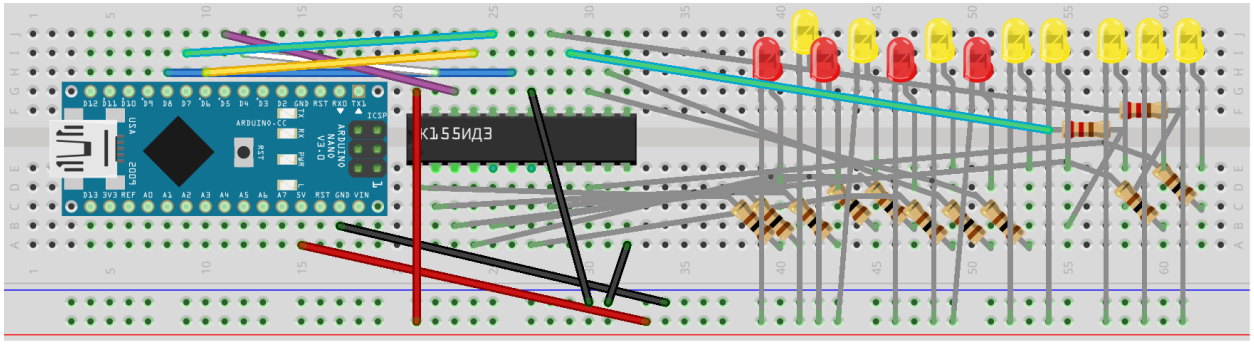
Если на одном из входов E0 или E1 (ножки 18, 19 ИД3) оставить напряжение низкого уровня, а на другой подавать некоторый информационный сигнал, то он будет передаваться на один из выходов 1–11, 13–17, номер выхода будет задаваться на адресных входах 23–20. Таким образом, входной код определяет, куда будет проходить сигнал из одного из стробирующих входов демультиплексора.

Подключим 19-й вывод ИД3 к 8-му выводу Ардуино. Пусть по нём будет идти прямоугольный сигнал. Поставим задачу определять номер светодиода, на который он будет подаваться и заставлять его мигать, во время выполнения программы, т.е. в процессе работы Ардуино. Для передачи номера по кабелю USB используем терминал COM-порта (кнопка «Монитор порта» в среде Ардуино находится справа вверху).

Если мы оставим адресные линии на тех же выводах Ардуино, где они находятся сейчас (0–3), то в силу того, что выводы 0 и 1 Ардуино используются для приёма-передачи данных по UART-интерфейсу, как раз задействованному при данном способе контакта Ардуино с компьютером и, соответственно, корректно работающей программы в этом случае не получится. Поэтому сдвинем все провода на 4 пина влево:



Нано-версия:



Для создания управляющей программы ознакомьтесь с набором функций Serial.

Прежде всего нас сейчас интересует команда Serial.read(), которая считывает очередной доступный байт из буфера последовательного соединения (см. пример программы).

Код программы будет примерно следующий:

```
int led;
int sdvig = 4; // на сколько пинов влево сдвинули провода
void setup() {
 DDRD = 255;
 DDRB = 1;
 Serial.begin(9600);  // подключаем монитор COM-порта
 }

void loop() {
   PORTB = 1;  // B1 – это пин 8, с которого идёт строб на 19 вывод ИД3
   delay(300);
   PORTB = 0;
   delay(300);
 if (Serial.available() > 0) {         //если есть доступные данные
     led = Serial.read();     // считываем байт
   if ((led>47)&&(led<58)) {       // по коду символа убеждаемся, что введена цифра
     led = led-48;// получаем саму цифру из её ASCII-кода
     Serial.print("led = ");
     Serial.println(led, DEC);        // выводим цифру на монитор порта
     PORTD = led<<sdvig;              // выводим сдвинутый на sdvig пин влево код в порт D
   }
 }
}
```