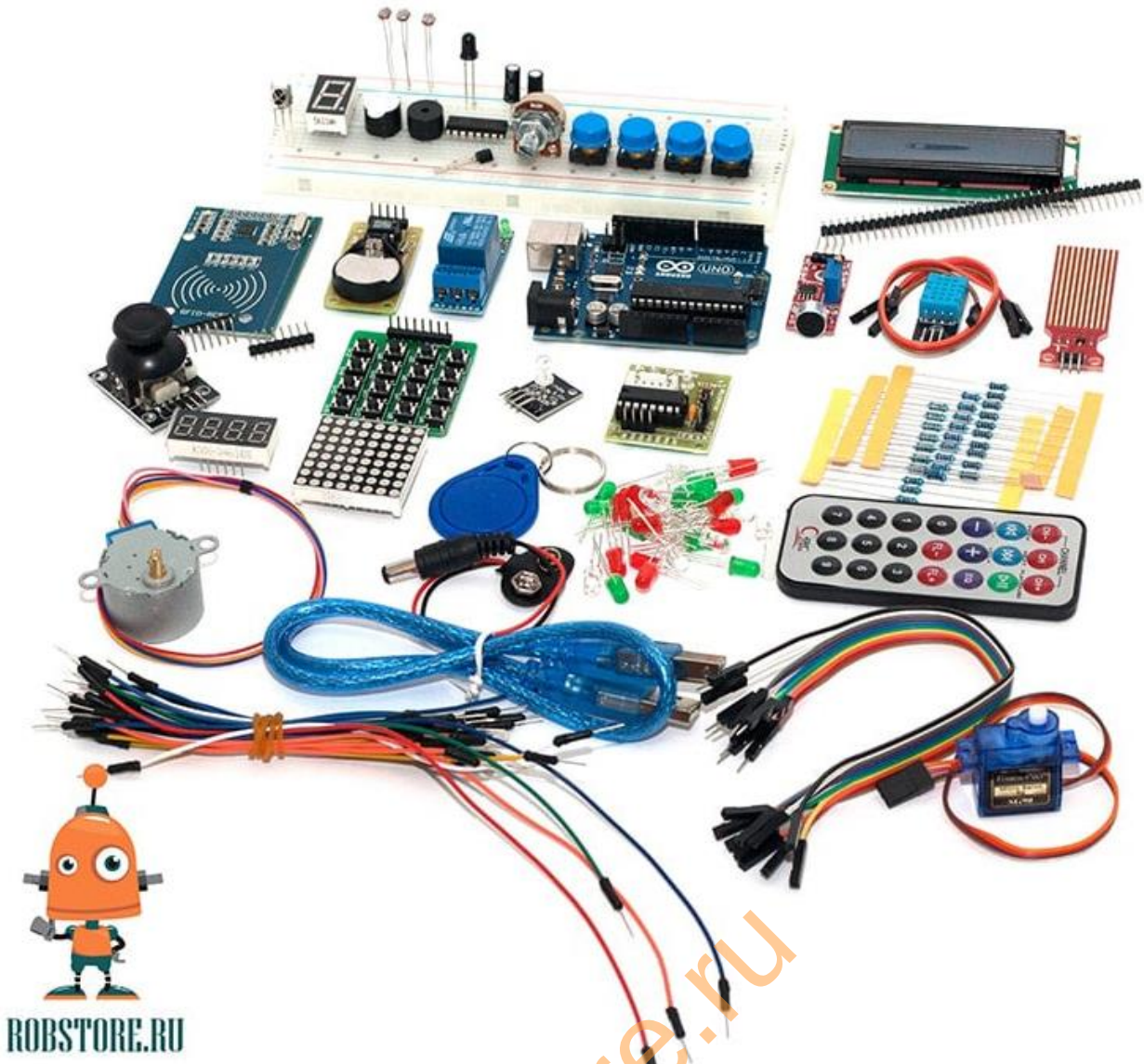
Уроки

Набор Ардуино Старт #1



Урок 1 - Скажи «Hello World», Arduino!

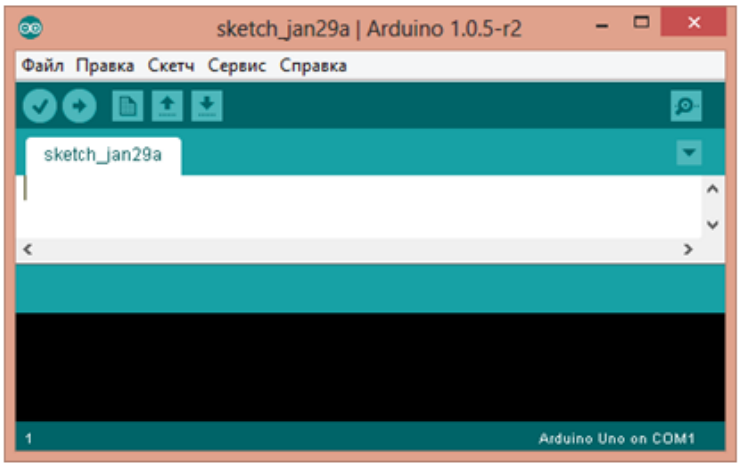
Для первого знакомства с платформой Arduino нам понадобится только сама плата Uno R3 и USB кабель для связи с компьютером. Наша плата скажет известную фразу «Hello World!», как только мы подключим её к компьютеру и запрограммируем.
Для эксперимента нам понадобятся:

Необходимые компоненты:

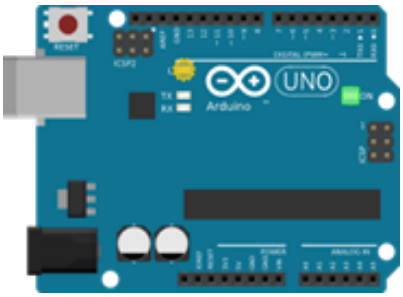
1. Плата Arduino Uno R3 * 1шт
2. Кабель USB типа A-B * 1шт

Установим и настроим Arduino в ОС Windows.

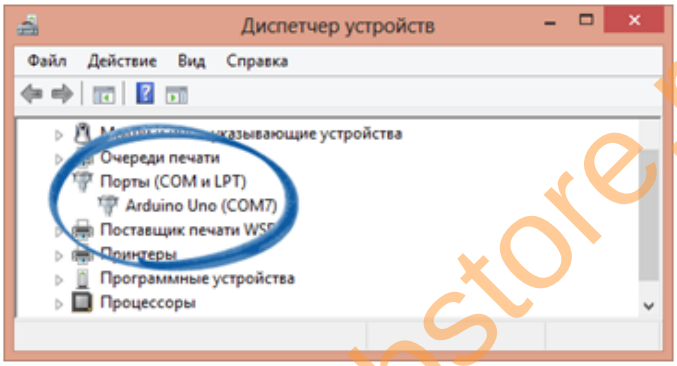
1. Установка Arduino IDE
   Первым делом установим на компьютер интегрированную среду разработки Arduino – Arduino IDE. Скачать самую свежую версию вы сможете по ссылке https://www.arduino.cc/en/Main/Donate.
   Устанавливая скачанную программу с помощью инсталлятора, вы не встретите никаких проблем и сложностей.
2. Запуск Arduino IDE
   После загрузки и установки запускаем нашу среду программирования!
   Перед нами открылось окно Arduino IDE. Мы ещё не подключили плату Arduino Uno к компьютеру, однако программа уже гордо показывает в правом нижнем углу строчку “Arduino Uno on COM1”. Это значит, что в данный момент Arduino IDE настроена на работу с портом COM1 и платой Arduino Uno. И когда вы попытаетесь загрузить программу, Arduino IDE будет искать Arduino Uno на порту COM1.
   Когда придёт время, мы изменим эти настройки.
   Если Arduino IDE не запускается, то вероятнее всего на компьютере неправильно установлена JRE (Java Runtime Environment). В этом случае следует вернуться к пункту (1) и переустановить Arduino IDE. Инсталлятор самостоятельно всё исправит.



3. Подключение Arduino к компьютеру
   Теперь пора подключить плату Arduino Uno к компьютеру. Для этого нам пригодится USB кабель. Как только вы подключите его к компьютеру, на плате загорится светодиод «ON» и начнёт мигать светодиод «L». Это говорит о том, что на плату подаётся питание и микроконтроллер исполняет прошитую в него на заводе программу «Blink» (мигает светодиодом раз в секунду).

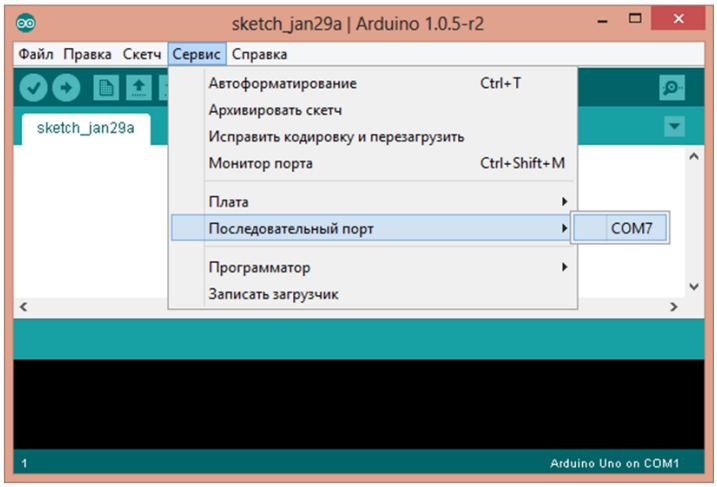


   Далее узнаем, какой номер COM-порта компьютер присвоил нашей Arduino Uno. Для этого заходим в «Диспетчер устройств» Windows и раскрываем вкладку «Порты (COM и LPT)».
   На картинке видно, что операционная система распознала нашу плату как COM-порт, подобрала под неё драйвер и назначила этому порту номер 7. Если теперь подключить к компьютеру другую плату, то операционная система присвоит ей уже другой номер. Поэтому очень важно внимательно относится к номеру COM-порта, иначе в них можно легко запутаться.

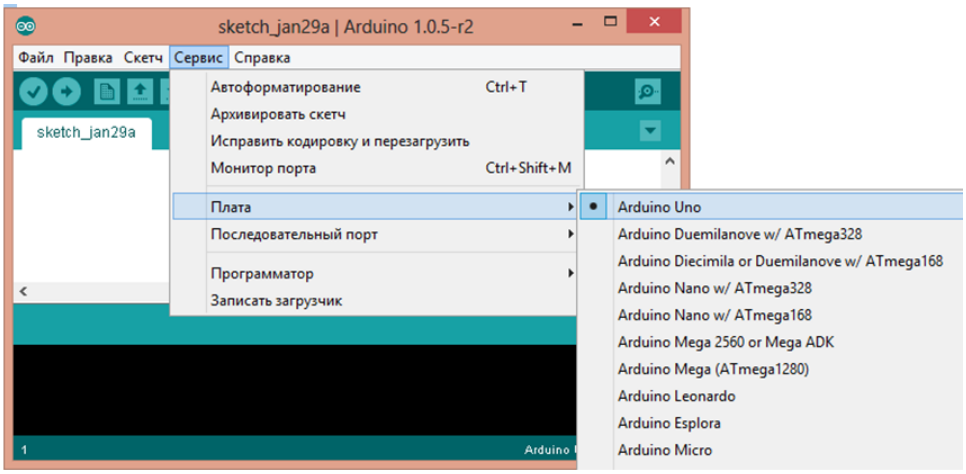


4. Настройка Arduino IDE на работу с Arduino Uno.

Теперь сообщим Arduino IDE, что та плата, с которой мы будем работать, находится на COM-порту «COM7».
Перейдём в меню «Сервис» - «Последовательный порт» и выберем порт «COM7». Так мы сообщили Arduino IDE, что какое то устройство находится на порту «COM7». И с ним ей и предстоит работать.



Далее объясним Arduino IDE, с каким же устройством ей предстоит работать. Для этого перейдём в меню «Сервис» - «Плата» и среди множества вариантов выберем «Arduino Uno».



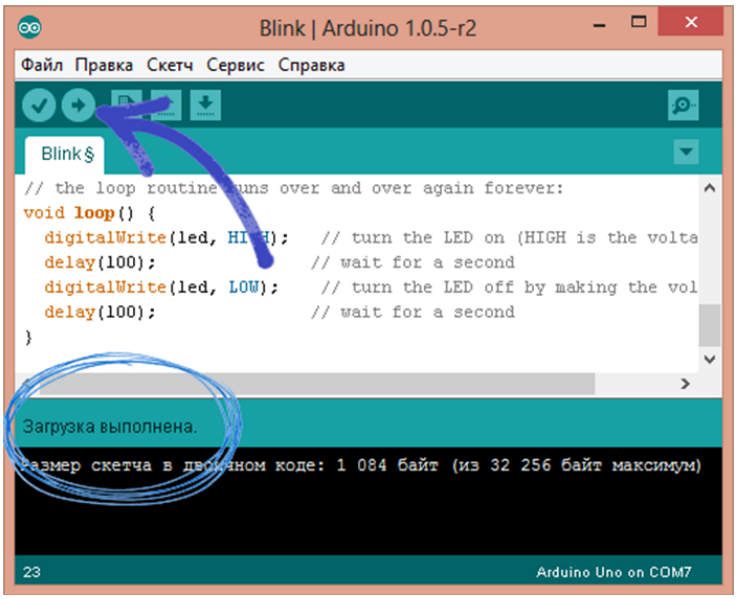
Если список последовательных портов пуст, значит Arduino Uno неправильно подключена. Вернитесь к пункту (3) и проверьте все шаги ещё раз.

5. Загрузка первого скетча
Arduino IDE настроили, плату успешно подключили. Теперь и скетч можно загрузить.
Скопируйте этот код и вставьте его в Arduino IDE:

```
int val ;
int ledpin = 13 ;
void setup ()            {
        Serial.begin (9600) ;
        pinMode (ledpin, OUTPUT) ;
}
void loop ()             {
```

```
        val = Serial.read () ;
if (val == 'R')             {
                            digitalWrite (ledpin, HIGH) ;
                            delay (500);
                            digitalWrite (ledpin, LOW) ;
                            delay (500);
                            Serial.println ("Hello World!") ;
        }
}
```

Мы специально не поясняем, что делает этот код. Он нужен лишь для первой демонстрации возможностей вашей Arduino UNO. Выполнив несколько уроков из этого руководства, вы сможете вернуться сюда и подробно разобрать каждую строчку.
Загрузите код в вашу Arduino Uno с помощью кнопки «Загрузить». После успешной загрузки Arduino IDE покажет сообщение «Загрузка выполнена».

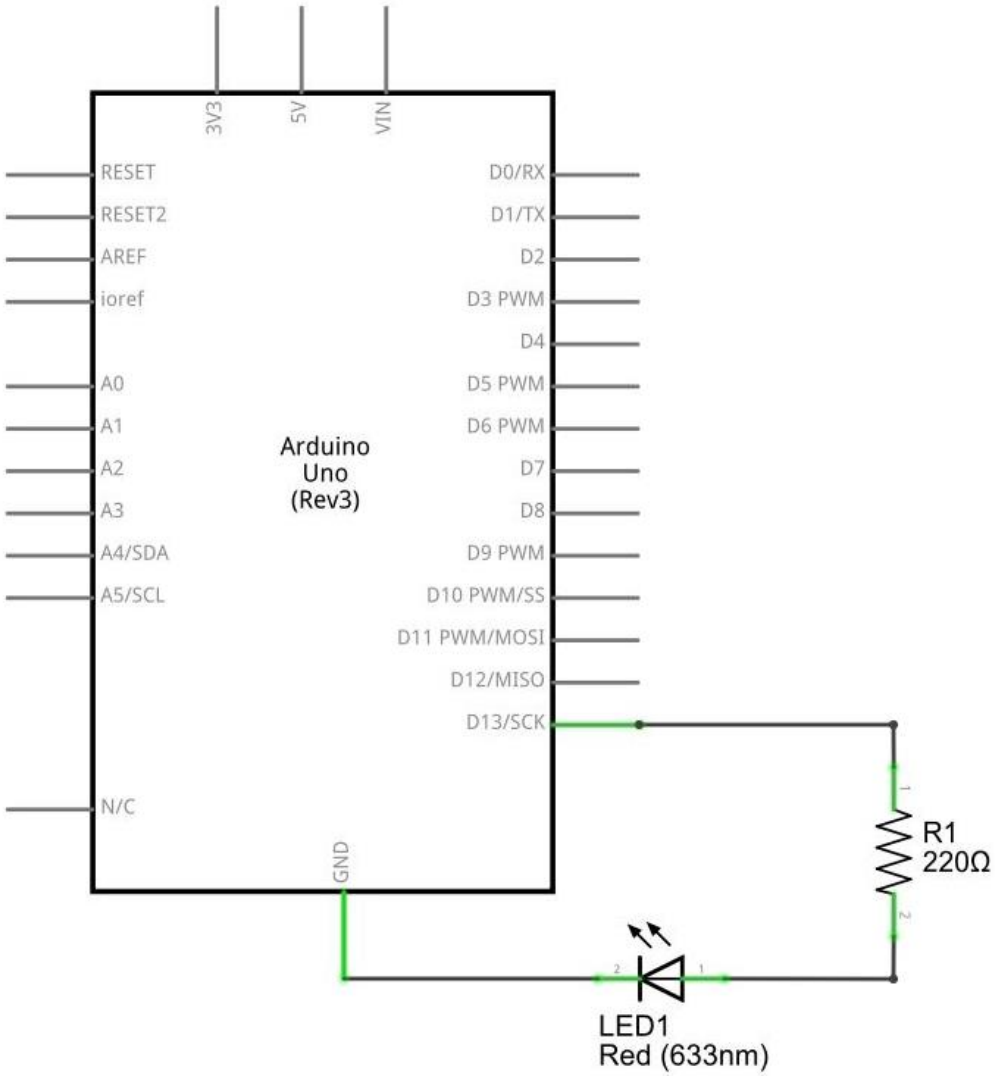


Теперь открываем Терминал (программа, с помощью которой мы сегодня и в следующих уроках будем отправлять данные нашему устройству и получать информацию с него). Для этого переходим «Сервис» - «Монитор порта».

Вводим символ «R» на клавиатуре и видим, как на плате загорелся и потух светодиод, и в ответ нам в терминал пришло сообщение «Hello World!». Вот так Arduino Uno поздоровалась с нами!

Урок 2 - Мигание встроенным светодиодом

Контроллер Arduino UNO уже содержит резистор и LED-светодиод, подключенный к 13 выводу, поэтому никаких других внешних радио-элементов нам не понадобится.



Ознакомившись со схемой, Вы можете приступать к написанию программы, в которой светодиод светит одну секунду и отключен другую.

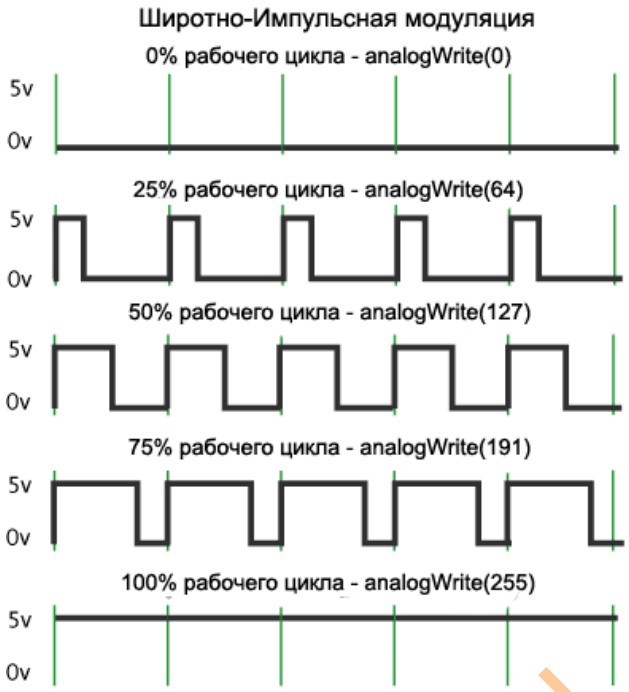
```
void setup ()
{
        pinMode (13, OUTPUT) ; // устанавливаем пин 13 как выход
}

void loop ()
{
        digitalWrite (13 , HIGH); // зажигаем светодиод
        delay (1000);  // задержка 1 секунда
        digitalWrite (13 , LOW); // выключаем светодиод
        delay (1000);  // задержка 1 секунда
}
```

После загрузки данной программы, Вы можете увидеть как мигает светодиод.

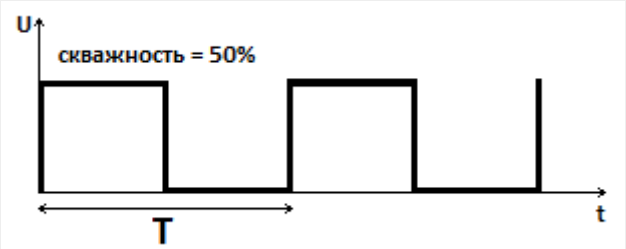
Урок 3 - Управление яркостью при помощи ШИМ

Широтно-Импульсная модуляция, или ШИМ, это операция получения изменяющегося аналогового значения посредством цифровых устройств. Устройства используются для получения прямоугольных импульсов - сигнала, который постоянно переключается между максимальным и минимальным значениями. Данный сигнал моделирует напряжение между максимальным значением (5 В) и минимальным (0 В), изменяя при этом длительность времени включения 5 В относительно включения 0 В. Длительность включения максимального значения называется шириной импульса. Для получения различных аналоговых величин изменяется ширина импульса. При достаточно быстрой смене периодов включения-выключения можно подавать постоянный сигнал между 0 и 5 В на светодиод, тем самым управляя яркостью его свечения.

Длительность периода обратно пропорциональна частоте ШИМ. Т.е. если частота ШИМ составляет 500 Гц, то зеленые линии будут отмечать интервалы длительностью в 2 миллисекунды каждый. Вызов функции analogWrite() с масштабом 0 – 255 означает, что значение analogWrite(255) будет соответствовать 100% рабочему циклу (постоянное включение 5 В), а значение analogWrite(127) – 50% рабочему циклу.



PWM (ШИМ) - используется во многих местах, например, изменение яркости света, скорости вращения двигателя и др. Вот основные параметры ШИМ:



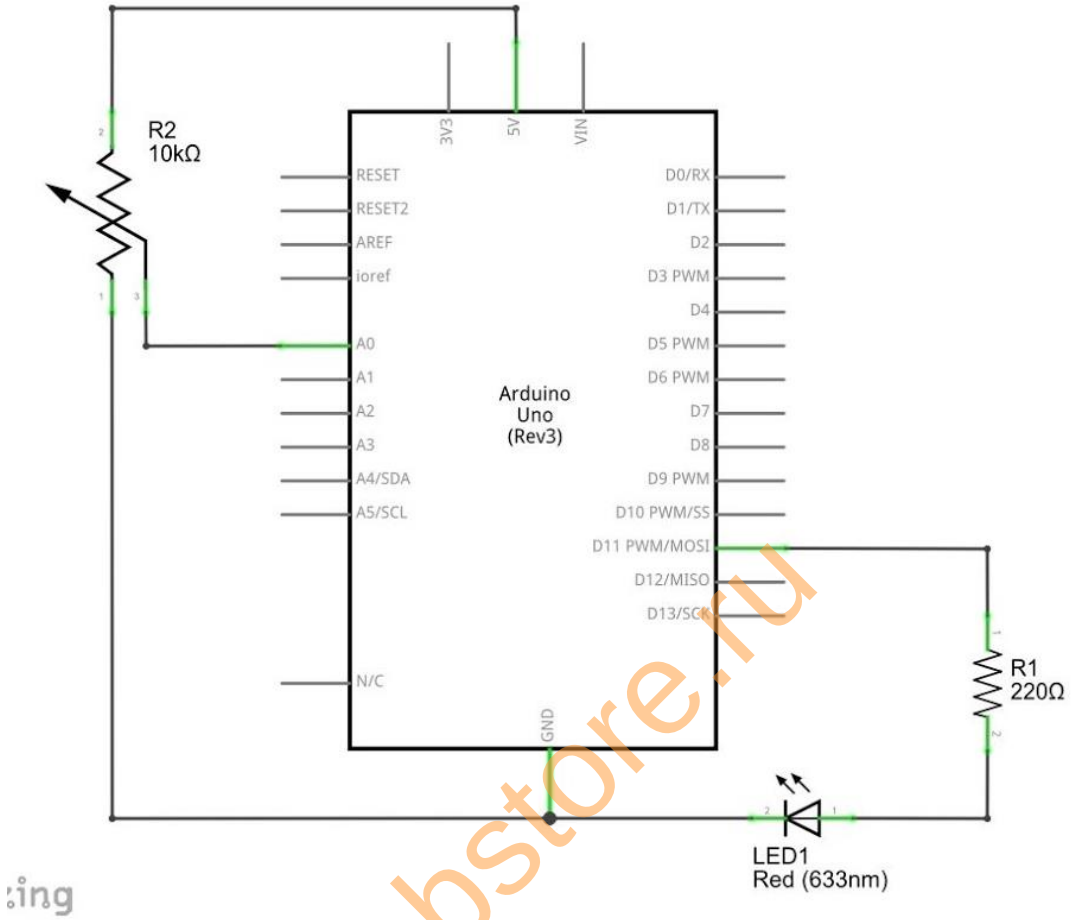
1. Скважность - отношение полного периода к времени включения
2. Период импульса
3. Высота напряжения (например: 0В-5В)

Arduino контроллер имеет шесть цифровых пинов ШИМ (3,5,6,9,10,11). Сделаем эксперимент по регулированию яркости при помощи ШИМ:

Необходимые компоненты:

1. Модуль Потенциометр * 1 шт.
2. Красный светодиод LED * 1 шт.
3. сопротивление 220Ω
4. Макетная плата * 1 шт.

Подключим потенциометр к аналоговому порту и будем считывать аналоговые значения. В зависимости от значения, считанного с потенциометра, с помощью ШИМ изменим яркость светодиода.



В приведенной ниже программе будем использовать функцию analogWrite (PWM интерфейс, аналоговое значение). Мы будем читать аналоговое значение с потенциометра и в зависимости от него изменять яркость светодиода.

```
int potpin = 0 ; // пин потенциометра 0
int ledpin = 11 ; // цифровой пин  11 (PWM)
int val = 0 ; // значение с потенциометра
void setup ()
{
        pinMode (ledpin, OUTPUT) ; // устанавливаем пин 11, как выход
        Serial.begin (9600) ; // устанавливаем скорость 9600
         / / Примечание: аналоговый интерфейс по умолчанию устанавливается как вход
 }

void loop ()
 {
        val = analogRead (potpin) ; // считываем значение с потенциометра в переменную val
        Serial.println (val) ; // отображаем полученное значение
        analogWrite (ledpin, val / 4) ; // возвращаем значение на ШИМ, чтобы установить яркость светодиода (ШИМ максимальное значение 255)
         delay (10) ; // задержка 0.01 секунд
}
```

После загрузки программы можно увидеть, что крутя ручку потенциометра - изменяется значение на экране, также меняется яркость светодиода.

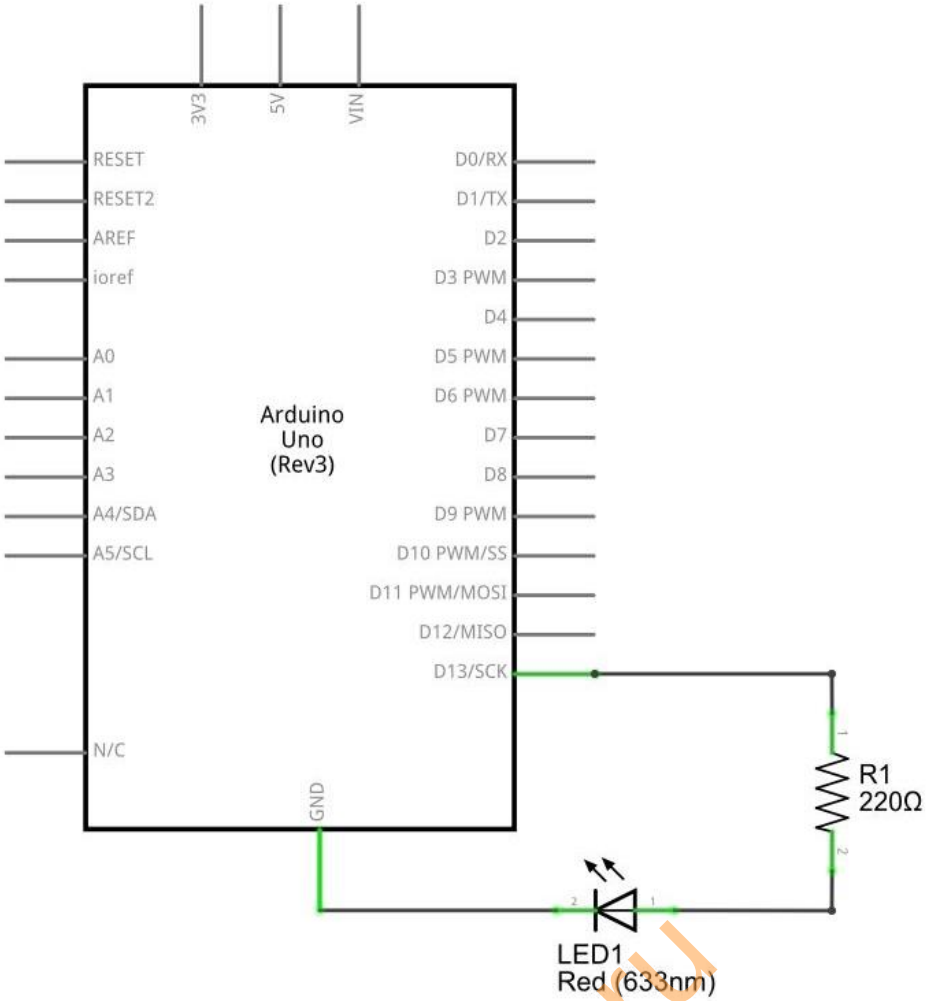
Урок 4 - Мигание LED светодиодом

В эксперименте “Hello World” был использован LED светодиод на самой плате Arduino. В этот раз мы будем использовать другие порты ввода/вывода и внешний светодиод.

Необходимые компоненты:

1. контроллер Arduino
2. светодиод
3. резистор 220 Ом

На следующей схеме будет использован 10 порт Arduino. Светодиод должен быть подключен через резистор 220 Ом, для ограничения тока.



После того, как Вы ознакомитесь со схемой, можете приступать к написанию программы. Во всех предыдущих урокам мы зажигали и выключали светодиод на одну секунды. В данном примере кода будет одно различие, светодиод будет подключен к 10 выводу Arduino, а не к 13.

```
int ledPin = 10; // вывод к которому подключен светодиод 10
void setup ()
```

```
{
        pinMode (ledPin, OUTPUT) ; // устанавливаем пин на выход
}
void loop () {
        digitalWrite (ledPin, HIGH); // зажимаем светодиод
        delay (1000); // задержка 1 сек.
        digitalWrite (ledPin, LOW); // выключаем светодиод
        delay (1000); // задержка 1 сек.
}
```

После загрузки программы Вы увидите мерцание светодиода, подключенного к 10 выводу Arduino. Эксперимент завершен.

Урок 5 - Мигание LED светодиодом. Эффект рекламы

Необходимые компоненты:

1. Led светодиоды: 6 шт.
2. 220Ω резисторы: 6 шт.
3. Разноцветные перемычки для макетной платы

Вы часто видите на рекламных щитах этот эффект. В следующем примере кода будет поочередно зажигаться светодиоды.

```
int BASE = 2; // пин, с которого начинать подключать
int NUM = 6; // всего светодиодов
void setup ()
{
for (int i = BASE; i <BASE+NUM; i)
{
pinMode (i, OUTPUT); // устанавливаем пины как OUTPUT
}
}
void loop ()
{
for (int i = BASE; i <BASE NUM; i)
{
digitalWrite (i, LOW); // выключаем светодиод на пине
delay (200); // задержка 200мс
}
for (int i = BASE; i <BASE NUM; i)
{
digitalWrite (i, HIGH); //  включаем светодиод на пине
delay (200); // задержка 200мс
}
}
```

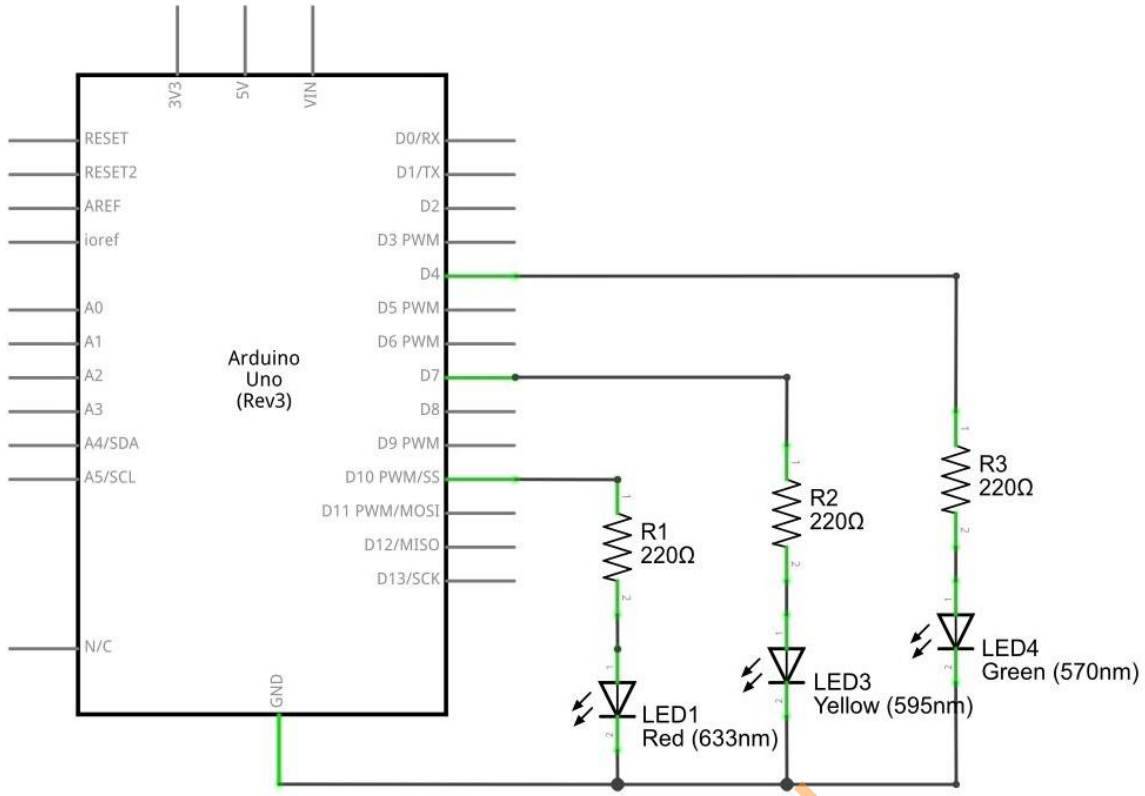
Урок 6 - Эксперимент с 3 светодиодами. Светофор.

Выше мы завершили небольшой эксперимент управления светом. Основываясь на полученных знаниях, вы можете реализовать более сложные световые эксперименты.

Необходимые компоненты:

1. Красный светодиод 5мм - 1 шт.
2. Зеленый светодиод 5мм - 1 шт.
3. Желтый светодиод 5мм - 1 шт.
4. Резистор 220Ω - 3 шт.
5. Макетная плата - 1 шт.
6. Перемычки макетной платы - несколько

В этом уроке будет рассмотрена имитация переключения сигнала светофора. Цвета будут соответствовать реальным.



Подключите светодиоды через резистор к цифровым выводам платы 10, 7, 4. Для красоты выставите светодиоды вдоль одной линии в порядке использующемся на светофоре (зеленый, желтый, красный).

```
int redled = 10; // красный светодиод пин 10
int yellowled = 7; // желтый светодиод пин 7
int greenled = 4; // зеленый светодиод пин 4
```

```
void setup () {
        pinMode (redled, OUTPUT) ; // выставляем пин в  OUTPUT
        pinMode (yellowled, OUTPUT); // выставляем пин в  OUTPUT
        pinMode (greenled, OUTPUT); // выставляем пин в  OUTPUT
 }
 void loop () {
        digitalWrite (redled, HIGH) ; // включаем красный
        delay (1000) ; // задержка 1 сек.
        digitalWrite (redled, LOW); // выключаем красный
        digitalWrite (yellowled, HIGH) ; // включаем желтый
        delay (200) ; // задержка 0.2 сек.
        digitalWrite (yellowled, LOW) ; // выключаем желтый
        digitalWrite (greenled, HIGH) ; // включаем зеленый
        delay (1000) ; // задержка 1 сек.
        digitalWrite (greenled, LOW) ; // выключаем зеленый
}
```

После загрузки программы можно увидеть как мигает маленький светофор.

Урок 7 - Эксперимент с пьезоэлементом

Пьезоэлемент (зумер, динамик), используется в робототехнике для управления звуками, издаваемыми роботом.

Пьезоэлемент — электромеханический преобразователь, одним из разновидностей которого является пьезоизлучатель звука, который также называют пьезодинамиком, просто звонком или английским buzzer. Пьезодинамик переводит электрическое напряжение в колебание мембраны. Эти колебания и создают звук (звуковую волну).

Необходимые компоненты:

1. Пьезоэлемент : 1 шт.
2. Кнопка: 1 шт.
3. Макетная плата : 1 шт.
4. Разноцветные перемычки для макетной платы

Подключить зумер очень просто - один вывод к земле, а второй к пину Arduino. После загрузки следующей программы зумер должен издавать звуки высокой и низкой частоты.

```
int buzzer = 8 ; // 8 пин Arduino для управления зумером
void setup ()
{
	pinMode (buzzer, OUTPUT) ; //  выставляем пин в  OUTPUT
}
void loop ()
{
	unsigned char i, j ; // локальные переменные
	while (1) {
		for (i = 0; i <80; i + +) // частота звука
		{
			digitalWrite (buzzer, HIGH) ; // издает звук
			delay (1) ; // задержка 1ms
			digitalWrite (buzzer, LOW) ; // не издает звук
			delay (1) ; // задержка  1ms
		}
		for (i = 0; i <100; i + +) // частота звука
		{
			digitalWrite (buzzer, HIGH) ; // издает звук
			delay (2) ; // задержка 2ms
			digitalWrite (buzzer, LOW) ; // не издает звук
			delay (2) ; // задержка 2ms
		}
	}
}
```

Урок 8 - Эксперимент с датчиком наклона

Необходимые компоненты:

1. Датчик наклона : 1 шт.
2. LED светодиод: 1 шт.
3. Макетная плата : 1 шт.
4. Разноцветные перемычки для макетной платы

Подключить LED светодиод к 8 пину Arduino, а датчик наклона к аналоговому 5 пину.

Принцип работы: если датчик наклонен чуть ниже горизонтального положения, то аналоговое напряжение составляет 5V (значение 1023), если датчик в горизонтальном положении - аналоговое напряжение 0V (значение 0). Среднее промежуточное значение 2.5V (значение 512).

В следующей программе мы будем зажигать светодиод, когда датчик будет ниже горизонтального (значение на аналоговом порту больше 512).

```
void setup ()
{
	pinMode (8, OUTPUT) ; // выставляем пин в  OUTPUT
}
void loop ()
{
	int i ; // значение с датчика
	while (1)
	{
		i = analogRead (5) ; // считать значение датчика наклона с порта 5
		if (i> 512 ) // если значение больше 512 (2.5V)
		{
			digitalWrite (8, HIGH) ; // зажечь светодиод
		}
		else / / Otherwise
		{
			digitalWrite (8, LOW) ; // выключить светодиод
		}
	}
}
```

После загрузки программы Вы увидите, что немного наклонив датчик, загорается светодиод.

Урок 9 - Эксперимент управление LED светодиодом при помощи кнопки.

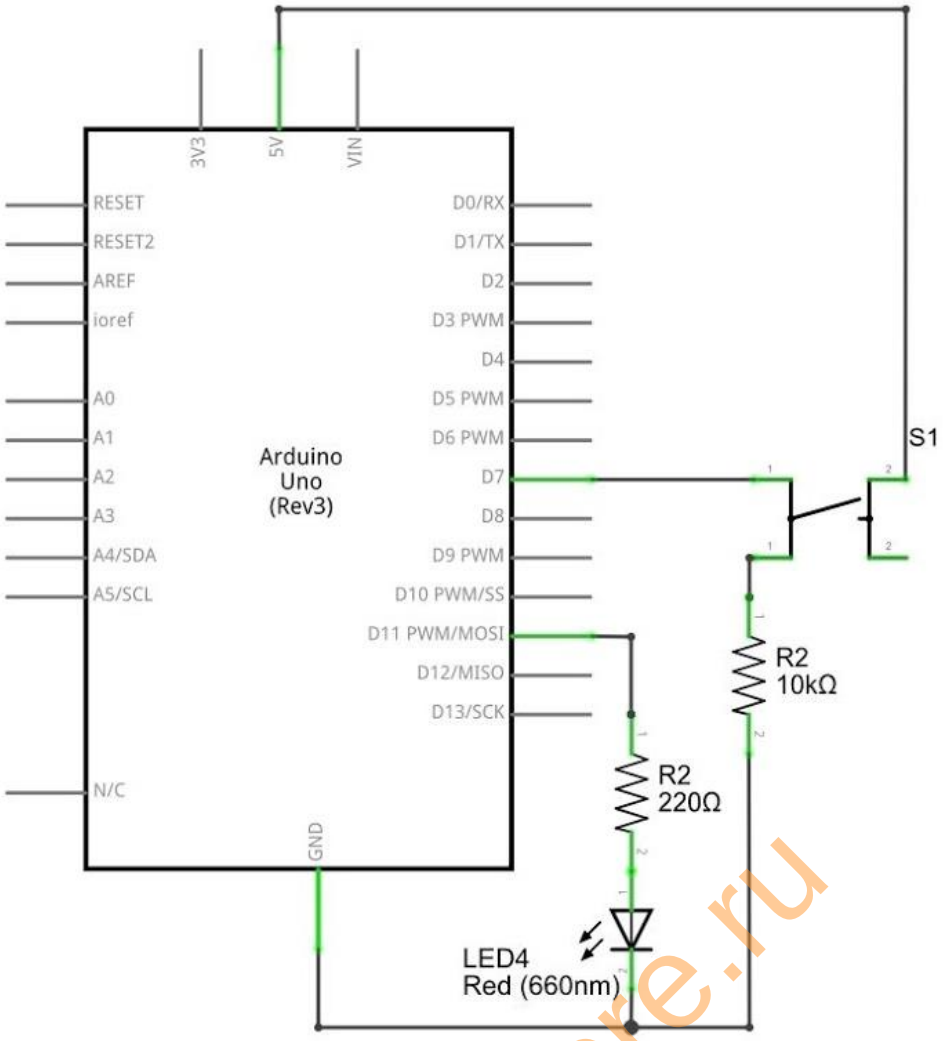
Подключаем к ардуино кнопку и светодиод (при нажатой кнопке светодиод будет гореть, при отжатой — не гореть). Это одна из базовых схем, которая неоднократно

пригодиться вам в будущем и может использоваться для управления роботом на Arduino.

Необходимые компоненты:

1. Кнопка: 1 шт.
2. Красный LED светодиод 5 мм: 1 шт.
3. Резистор 220Ω: 1 шт.
4. Резистор 10КΩ: 1 шт.
5. Макетная плата : 1 шт.
6. Разноцветные перемычки для макетной платы

Вы можете использовать цифровые любые пины Arduino (0-13) для подключения кнопок или светодиодов. Не рекомендуется использовать пины 0,1 (используются для связяи с ПК).



Подключите кнопку к 7 пину Arduino, а светодиод к 11. После того, как с 7 пина будет считываться ноль (кнопка будет нажата) - светодиод будет выключаться.

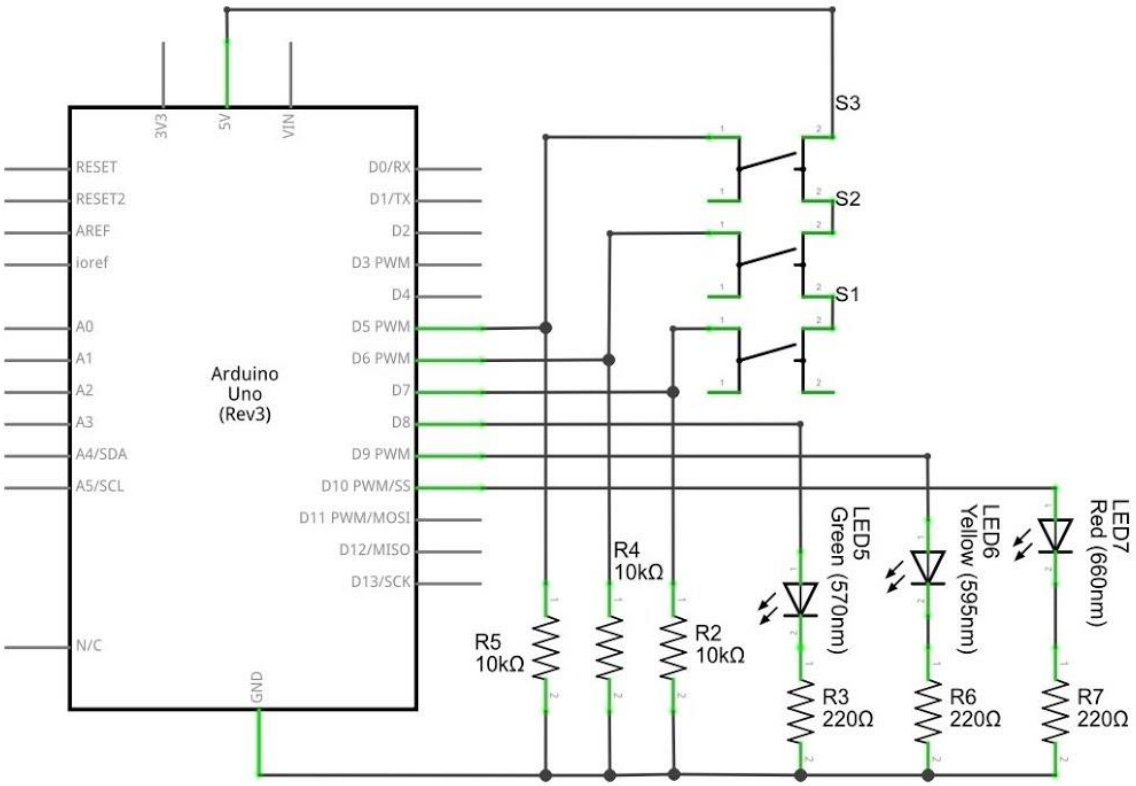
```
int ledpin = 11 ; // пин 11 - светодиод
```

```
int inpin = 7 ; // пин 7 - кнопка
int val ; // значение кнопки
void setup ()
{
        pinMode (ledpin, OUTPUT) ; // выставляем пин в  OUTPUT
        pinMode (inpin, INPUT) ; // выставляем пин в  INPUT
}
void loop ()
{
        val = digitalRead (inpin) ; // считываем значение с 7 пина в переменную val
        if (val == LOW) // нажатие кнопки
        {DigitalWrite (ledpin, LOW);}
        else {DigitalWrite (ledpin, HIGH);}
 }
```

После загрузки программы мы сможем управлять светодиодом при помощи кнопки. Данный метод часто встречается в реальной жизни, например, подсветка клавиш телефона.

Урок 10 - Эксперимент управление LED светодиодами при помощи кнопок.

После успешного выполнения предыдущего урока его можно усложнинить. Добавим еще светодиодов и кнопок. Принцип работы останется прежним, только теперь мы мы будем управлять тремя разноцветными светодиодами при помощи трех кнопок.




```
int redled = 10;
int yellowled = 9;
int greenled = 8;
int redpin = 7;
int yellowpin = 6;
int greenpin = 5;
int red;
int yellow;
int green;

void setup ()
{
        pinMode (redled, OUTPUT);
        pinMode (yellowled, OUTPUT);
        pinMode (greenled, OUTPUT);
        pinMode (redpin, INPUT);
        pinMode (yellowpin, INPUT);
```

```
        pinMode (greenpin, INPUT);
 }
void loop ()
{
        red = digitalRead (redpin);
        if (red == LOW) {DigitalWrite (redled, LOW);}
        else {DigitalWrite (redled, HIGH);}
        yellow = digitalRead (yellowpin);
        if (yellow == LOW) {DigitalWrite (yellowled, LOW);}
        else {DigitalWrite (yellowled, HIGH);}
        green = digitalRead (greenpin);
        if (green == LOW) {DigitalWrite (greenled, LOW);}
         else {DigitalWrite (greenled, HIGH);}
}
```

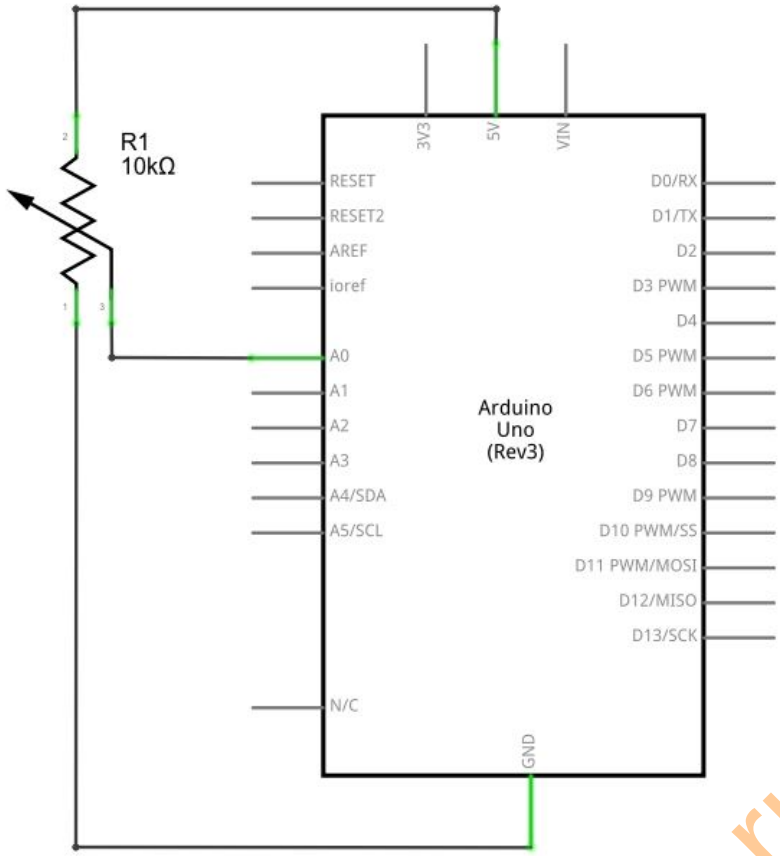
Данная программа является полной копией предыдущей с едиственным отличием - увеличилось количество компонентов.

Урок 11 - Эксперимент - чтение аналогового значения. Потенциометр.

В данном уроке мы должны изучить использование аналогового интерфейса ввода/вывода. На данной плате Arduino 6 аналоговых пинов (A0-A5), которые могут быть использованы, как цифровые. Эти пины обозначаются на плате номерами с 14 по 19.
Аналоговое значение будем считывать с потенциометра. Потенциометр — это переменный резистор с регулируемым сопротивлением. Потенциометры используются в робототехнике как регуляторы различных параметров — громкости звука, мощности, напряжения и т.п.

Необходимые компоненты:

1. Потенциометр: 1 шт.
2. Макетная плата : 1 шт.
3. Разноцветные перемычки для макетной платы



В данном уроке мы будет считывать аналоговое значение с потенциометра, а потом отображать на экране. Для подключения будет использован интерфейс A0.
Считать значение очень просто - для этого используется функция analogRead(). Данная плата Arduino позволяет считать 10-битное значение (0-1023).

Полученную информацию мы будем выводить на экран ПК. Для установки скорости передачи информации используется функция Serial.begin (). Для отображения значения на экране - функции Serial.print () или Serial.println (). Отличаются между собой лишь тем, что вторая имеет автоматический перенос коретки (перенос на след. строку).

```
int potpin = 0 ; // потенциометр пин 0
int ledpin = 13 ; // встроенный светодиод пин 13
int val = 0 ; // аналоговое значение
void setup ()
{
        pinMode (ledpin, OUTPUT) ;// выставляем пин в  OUTPUT
        Serial.begin (9600) ; // устанавливаем скорость передачи 9600
}
void loop ()
{
        digitalWrite (ledpin, HIGH) ; // зажигаем светодиод на пине 13
        delay (50) ; // задержка 0.05 секунд
        digitalWrite (ledpin, LOW) ; // выключаем светодиод на пине 13
        delay (50) ; // задержка 0.05 секунд
        val = analogRead (potpin) ; // аналоговое значение в переменную  val
        Serial.println (val) ; // показываем значение на экране
}
```

Каждый раз, когда значение считывается с аналогового порта - мигает встроенный светодиод Arduino. Если крутить ручку потенциометра - значение на экране будет изменяться.

Урок 12 - Эксперимент - звук, управление светом.

Необходимые компоненты:

1. Фоторезистор: 1 шт.
2. Пьезоэлемент : 1 шт.
3. Разноцветные перемычки для макетной платы

В прошлых уроках мы подключали компоненты непосредственно к цифровому порту, в этот раз подключим зумер и фоторезистор последовательно на 6 порт.
Если нет света, то звук тихий. Если больше света - сопротивление на фоторезисторе уменьшается - звук с зумера громче.

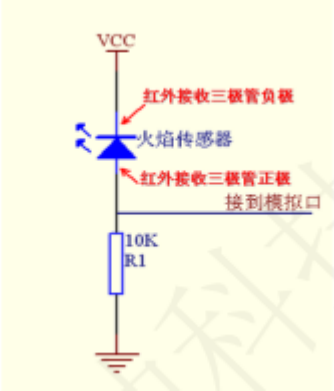
```
void setup ()
{
	pinMode (6, OUTPUT);
}

void loop ()
{
	while (1)
	{
		char i, j;
		while (1)
		{
			for (i = 0; i <80; i)  // частота звука
			{
				digitalWrite (6, HIGH);
				delay (1);
				digitalWrite (6, LOW);
				delay (1);
			}
			for (i = 0; i <100; i) // частота звука
			{
				digitalWrite (6, HIGH);
				delay (2);
				digitalWrite (6, LOW);
				delay (2);
			}
		}
	}
}
```

После загрузки программы Вы можете использовать, например, фонарик, чтобы сделать звук зумера громче. С помощью фоторезистора Вы так же сможете сами сделать эксперимент по управлению яркостью светодиода.

Урок 13 - Эксперимент - сигнализация.

Датчик обнаружения огня (т.е. инфракрасный датчик) очень чувствителен к пламени. Принцип действия - обнаружения изменения светимости и преобразование данного сигнала в значение.

Схема подключения:



Необходимые компоненты:

1. Датчик огня: 1 шт.
2. Пьезоэлемент: 1 шт.
3. 10 КΩ резистор: 1 шт.
4. Разноцветные перемычки для макетной платы

Подключите зумер к цифровому пину 8, а датчик пламени к аналоговому пину 5. В зависимости от удаленности пламени значение на аналоговом порте будет меняться. При отсутствии пламени напряжение на аналоговом порте 0.3V, когда пламя вблизи - напряжение 1.5V, чем ближе, тем напряжение больше. В следующей программе, если напряжение > 0.6V (цифровое значение 123), раздается звук для тревоги, если меньше - зумер не пищит.

```
int flame = A5 ; // аналоговый пин для датчика огня
int Beep = 8 ; // цифровой пин  7 для зуммера
int val = 0 ;// значение с датчика
void setup ()
{
        PinMode (Beep, OUTPUT) ; // выставляем пин в OUTPUT
        pinMode (flame, INPUT) ; // выставляем пин в INPUT
        Serial.begin (9600) ; // устанавливаем скорость передачи с пк -  9600
}
void loop ()
{
        val = analogRead (flame) ; // считываем значение с датчика огня
        Serial.println (val) ; // выводи значение
        if (val> = 123) // если значение больше 0.6V зумер пищит
        {DigitalWrite (Beep, HIGH);}
```

```
        else {digitalWrite (Beep, LOW);}
}
```

Данная программа может имитировать сигнал тревеги в случае пожара.

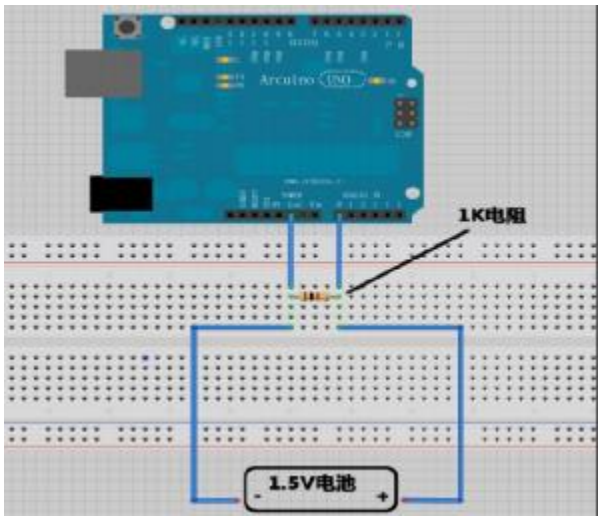
Урок 14 - Эксперимент - делаем вольтметр 0-5V из Arduino.

Примечание: в данной схеме нет защиты электрической цепи, поэтому не рекомендуется использовать две и более батареи типа AA.

Необходимые компоненты:

1. Макетная плата: 1 шт.
2. USB кабель: 1 шт.
3. 1KΩ резистор: 1 шт.
4. Разноцветные перемычки для макетной платы

Рассмотрим конструкцию схемы подключения схемы:



1K резистор в данной схеме подключения нужен для избежания помех.

```
float temp; // создаем переменную для временного хранения данных
void setup ()
{
        Serial.begin (9600); // 9600 baud скорость соединения
}
void loop ()
{
        int V1 = analogRead (A0);
        // читаем напряжение с A0, аналоговый порт выдает 0-5V и возвращает
        значение 0-1024
        float vol = V1 * (5.0 / 1023.0);
        // преобразуем в фактическое значение напряжение с плавающей точкой
        if (vol == temp)
        // используется для фильтрации повторяющихся данных
        {
                temp = vol;
                // После сравнения значение записывается в переменную
        }
        else
```

```
        {
                Serial.print (vol);
                // выводим значение напряжения
                Serial.println ("V");
                 // выводим символ V, переносим строку
                temp = vol;
                delay (1000);
                // ждем 1 секунду.
        }
}
```

Нажмите открыть serial monitor. Один провод присоедините к катоду батареи, второй к аноду. Данные будут появляться на экране один раз в секунду.

Урок 15 - Модуль обнаружения голоса

Обнаружение звука:

1) Аналоговый выход. В режиме реального времени сигнал с микрофона.
2) Есть крепежное отверстие под винт 3мм.
3) Использовать напряжение 5V постоянного тока
4) Микрофон очень чувствительный
5) Индикатор питания должен гореть.

ARDUINO код:

```
int sensorPin = A5; // выбирете пин для микрофона
int ledPin = 13; // пин для светодиода LED
int sensorValue = 0; // значение, которое приходит с датчика
void setup ()
{
        pinMode (ledPin, OUTPUT);
        Serial.begin (9600);
}
void loop ()
{
        sensorValue = analogRead (sensorPin);
        digitalWrite (ledPin, HIGH);
        delay (sensorValue);
        digitalWrite (ledPin, LOW);
        delay (sensorValue);
        Serial.println (sensorValue, DEC);
}
```

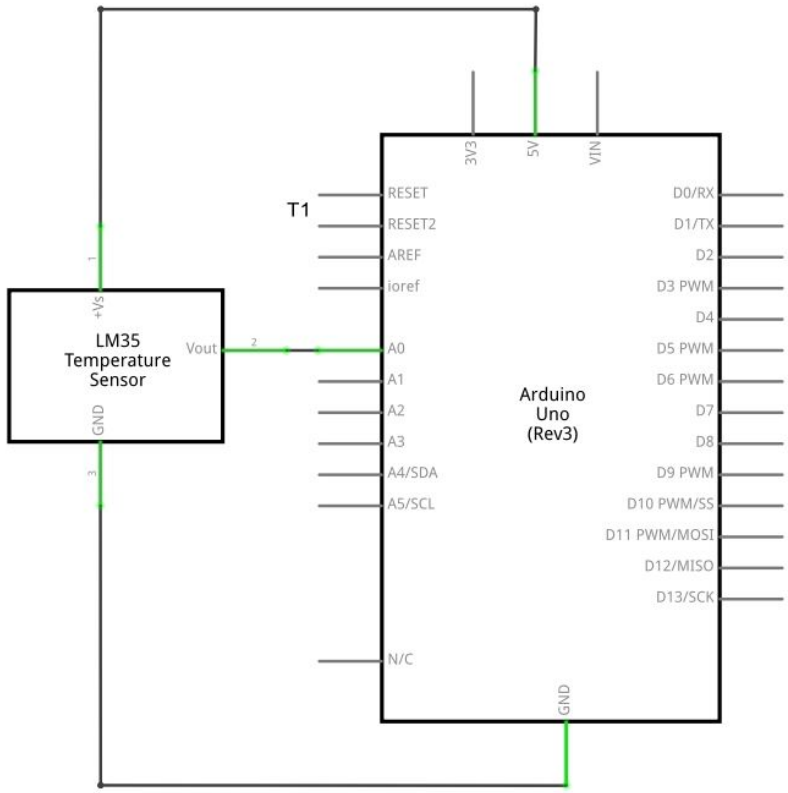
Урок 16 - Эксперимент - LM35 датчик температуры.

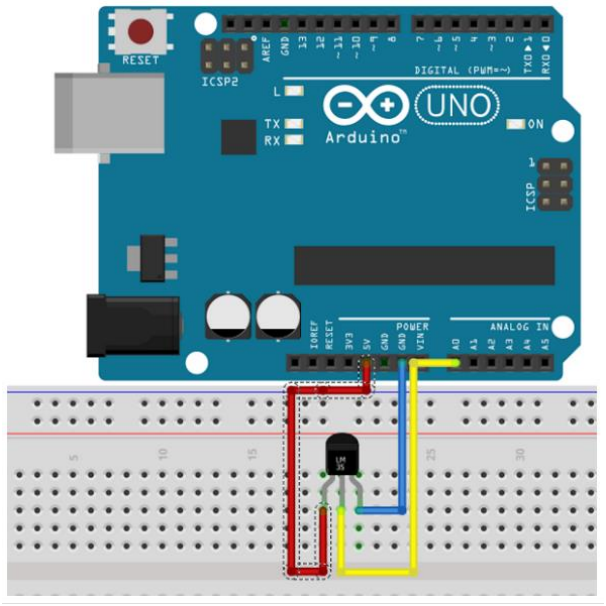
LM35 - достаточно распространенный и простой датчик температуры. Алгоритм считывания значения более сложный, чем просто считать значение с аналогового порта.

Необходимые компоненты:

1. Датчик температуры LM35
2. Макетная плата: 1 шт.
3. Разноцветные перемычки для макетной платы

Подключите схему в соответствии со следующей диаграммой.






```
int potPin = 0; // аналоговый пин для датчика
void setup () {
        Serial.begin (9600) ; // устанавливаем скорость
}
void loop ()
{
        int val ; // объявляем переменные
        int dat ; // объявляем переменные
        val = analogRead (0) ; // считываем аналоговое значение датчиков
        val dat = (125 * val) >> 8 ; // формула температуры
        Serial.print ("Тер:") ; // вывести строку Тер
        Serial.print (dat) ; // вывести значения температуры
        Serial.println ("C") ; // вывести строку градусов
        delay (500) ; // задержка 0.5 секунд
}
```

После загрузки программы откройте монитор порта и посмотрите значение температуры.

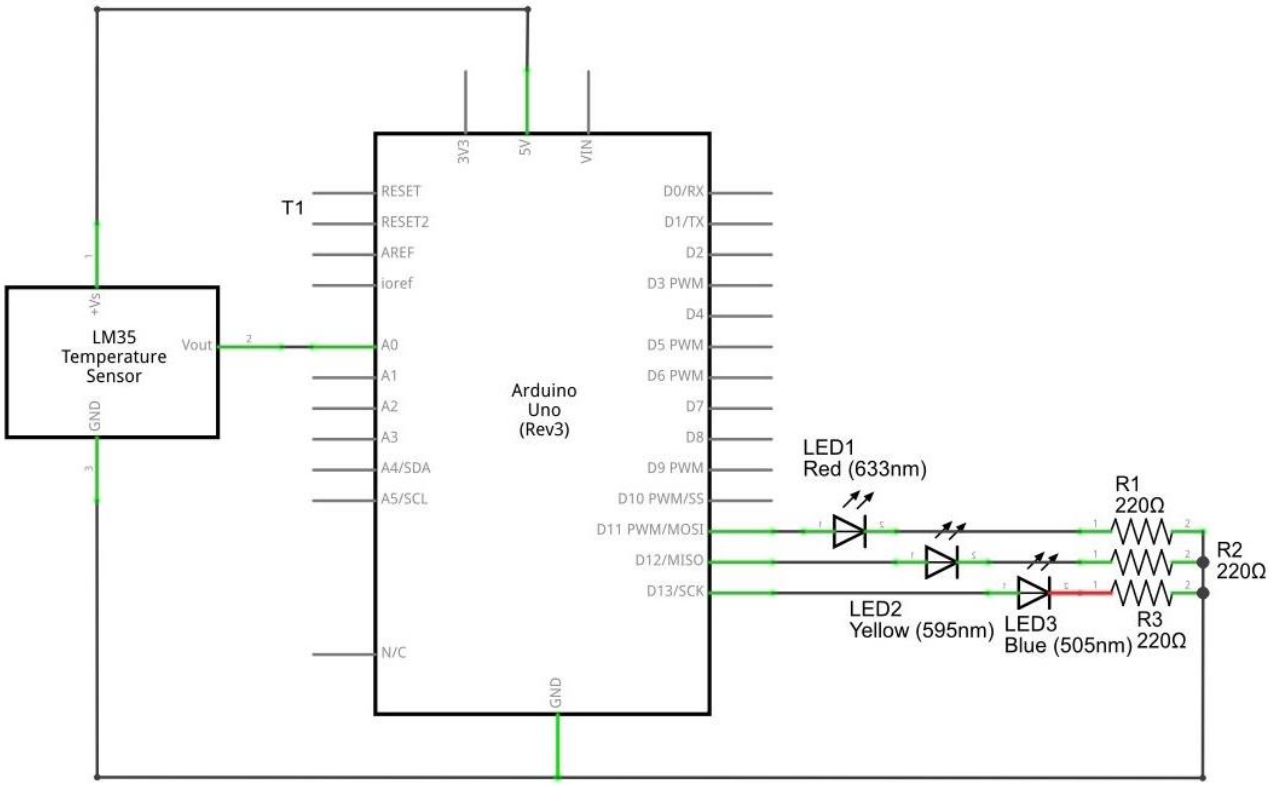
Урок 17 - Эксперимент - индикация температуры светодиодами.

Сделаем температурный анализатор, используя чувствительный датчик LM35. Разные цвета светодиодов будут отображать температуру.

Необходимые компоненты:

1. Arduino контроллер: 1 шт.
2. Макетная плата: 1 шт.
3. Красный, желтый, синий LED светодиод
4. 220 Ω резистор: 3 шт.
5. LM35 датчик температуры
6. Разноцветные перемычки для макетной платы

Схема:




```
void setup ()
{
        pinMode (13, OUTPUT);
        pinMode (12, OUTPUT);
        pinMode (11, OUTPUT);
}
void loop ()
{
        int vol = analogRead (A0) * (5.0 / 1023.0 * 100); // считываем температуру LM35
        if (vol <= 31)  // если значение температуры меньше указаного, зажигаем LED
        {
```

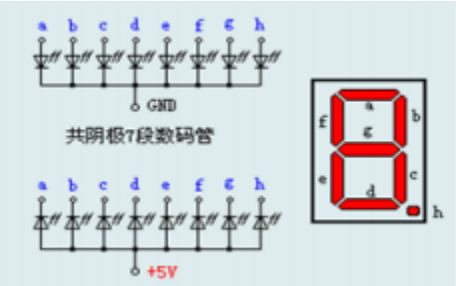
```
                digitalWrite (13, HIGH);
                digitalWrite (12, LOW);
                digitalWrite (11, LOW);
        }
        else if (vol> = 32 && vol <= 40)
        {
                digitalWrite (13, LOW);
                digitalWrite (12, HIGH);
                digitalWrite (11, LOW);
        }
        else if (vol> = 41) // если большая температура - зажигаем красный
        {
                digitalWrite (13, LOW);
                digitalWrite (12, LOW);
                digitalWrite (11, HIGH);
        }
}
```

После загрузки программы, в зависимости от температуры на датчике, будет зажигаться светодиод определенного цвета.

Урок 18 - Эксперимент - восьми сегментный индикатор.

Восьми сегментный индикатор - это полупроводниковый светоизлучающий прибор, основной блок которого представляет светоизлучающий диод. Цифровая трубка делится на количество сегментов, в данном случае - восемь.

Схема устройства:

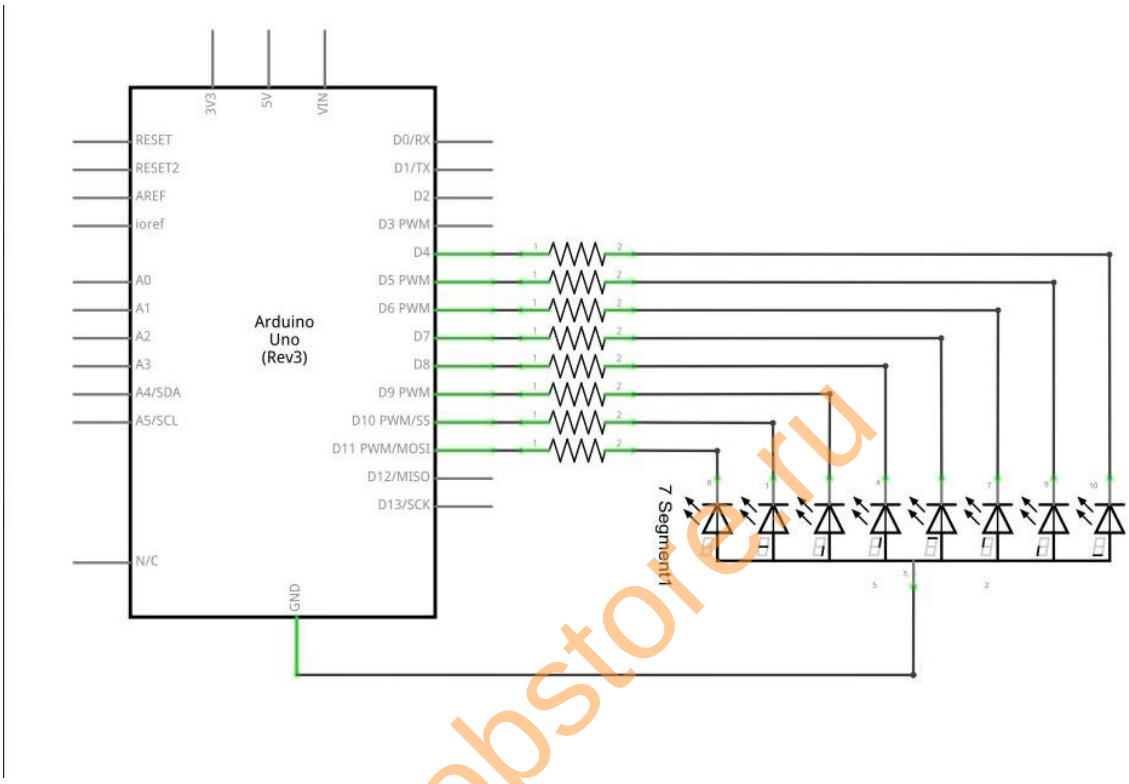


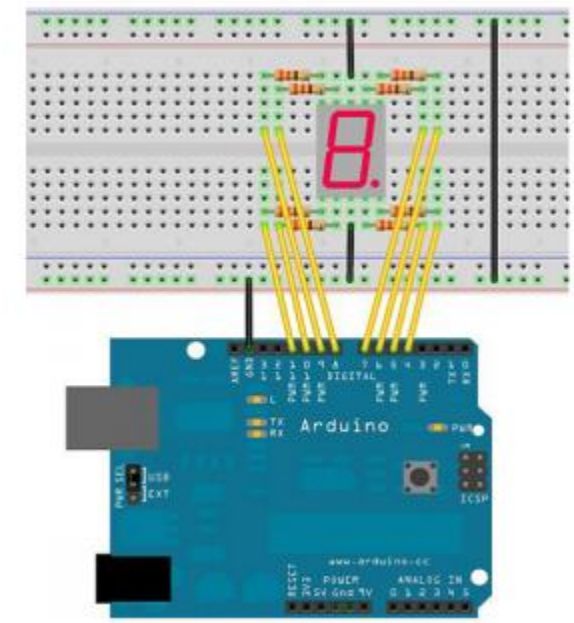
Каждая секция трубки состоит из светодиодов, значит необходимо последовательно подключить резистор к каждому выводу, чтобы не сжечь светодиоды. В данном эксперименте используется схема с общем катодом. Общий катод должен быть подключен к земле. Высокий сигнал на катоде секции соответствует свечению данного светодиода.

Необходимые компоненты:

1. Восьми сегментный индикатор: 1 шт.
2. Макетная плата: 1 шт.
3. 220 Ω резистор: 8 шт.
4. Разноцветные перемычки для макетной платы

Принципиальная схема подключения индикатора:





Есть семи сегментный цифровой десятичный индикатор и точка. Пусть 1 (b), будет освещен всегда, а остальные записаны в качестве подпрограммы. Каждые 2 секунды цикл отображает от 1 до 8 цифр.

```
int a = 7 ; // определяем пин сегмента LED
int b = 6 ; // определяем пин сегмента LED
int c = 5 ; // определяем пин сегмента LED
int d = 11 ; // определяем пин сегмента LED
int e = 10 ; // определяем пин сегмента LED
int f = 8 ; // определяем пин сегмента LED
int g = 9 ; // определяем пин сегмента LED
int dp = 4 ;// определяем пин сегмента LED
void digital_1 (void) // отображаем цифру 1
{
        unsigned char j;
        digitalWrite (c, HIGH) ; // подсвечиваем пин 5 (c)
        digitalWrite (b, HIGH) ; // подсвечиваем  b
        for (j = 7; j <= 11; j + +) // выключаем остальные
        digitalWrite (j, LOW);
        digitalWrite (dp, LOW) ; // выключаем сегмент DP
}
void digital_2 (void) // отображаем цифруr 2
{
        unsigned char j;
        digitalWrite (b, HIGH);
        digitalWrite (a, HIGH);
        for (j = 9; j <= 11; j + +)
        digitalWrite (j, HIGH);
        digitalWrite (dp, LOW);
        digitalWrite (c, LOW);
```

```
        digitalWrite (f, LOW);
}
void digital_3 (void) // отображаем цифру 3
{
        unsigned char j;
        digitalWrite (g, HIGH);
        digitalWrite (d, HIGH);
        for (j = 5; j <= 7; j + +)
        digitalWrite (j, HIGH);
        digitalWrite (dp, LOW);
        digitalWrite (f, LOW);
        digitalWrite (e, LOW);
}
void digital_4 (void) // отображаем цифру 4
{
        digitalWrite (c, HIGH);
        digitalWrite (b, HIGH);
        digitalWrite (f, HIGH);
        digitalWrite (g, HIGH);
        digitalWrite (dp, LOW);
        digitalWrite (a, LOW);
        digitalWrite (e, LOW);
        digitalWrite (d, LOW);
}
void digital_5 (void) // отображаем цифру 5
{
        unsigned char j;
        for (j = 7; j <= 9; j + +)
        digitalWrite (j, HIGH);
        digitalWrite (c, HIGH);
        digitalWrite (d, HIGH);
        digitalWrite (dp, LOW);
        digitalWrite (b, LOW);
        digitalWrite (e, LOW);
}
void digital_6 (void) // отображаем цифру 6
{
        unsigned char j;
        for (j = 7; j <= 11; j + +)
        digitalWrite (j, HIGH);
        digitalWrite (c, HIGH);
        digitalWrite (dp, LOW);
        digitalWrite (b, LOW);
}
void digital_7 (void) // отображаем цифрyr 7
{
        unsigned char j;
        for (j = 5; j <= 7; j + +)
```

```
        digitalWrite (j, HIGH);
        digitalWrite (dp, LOW);
        for (j = 8; j <= 11; j + +)
        digitalWrite (j, LOW);
}
void digital_8 (void) // отображаем цифруr 8
{
        unsigned char j;
        for (j = 5; j <= 11; j + +)
        digitalWrite (j, HIGH);
        digitalWrite (dp, LOW);
}
void setup ()
{
        int i ;// define variables
        for (i = 4; i <= 11; i + +)
        pinMode (i, OUTPUT) ; // устанавливаем пины в OUTPUT
}
void loop ()
{
        while (1)
        {
                digital_1 () ; // отображаем цифру 1
                delay (2000) ; // задержка 2s
                digital_2 () ; // отображаем цифру 2
                delay (1000); // задержка 1s
                digital_3 () ; // отображаем цифру 3
                delay (1000); // задержка 1s
                digital_4 () ; // отображаем цифру 4
                delay (1000); // задержка 1s
                digital_5 () ;// отображаем цифру 5
                delay (1000); // задержка 1s
                digital_6 () ;// отображаем цифру 6
                delay (1000); // задержка 1s
                digital_7 () ; // отображаем цифру 7
                delay (1000); // задержка 1s
                digital_8 () ; // отображаем цифру 8
                delay (1000); // задержка 1s
        }
}
```

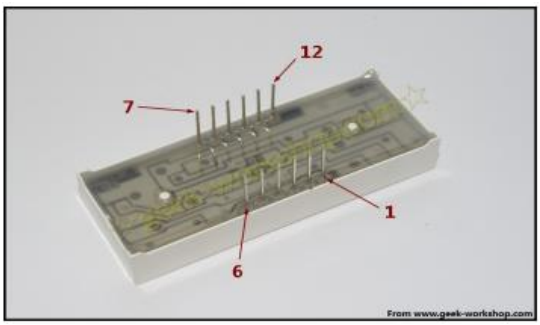
В цикле программы вызывается раз ранее описанных подпрограмм, которые отображают числа.

Урок 19 - Эксперимент - 4 разрядный индикатор.

У него 4 разряда по 8 сегментов (7 — цифра и 1 — точка). Для того что бы отобразить на нём значение необходимо на катод нужного разряда подать «-» а на необходимые сегменты «+» что бы получилась цифра.

Так как каждый сегмент разряда завязан на одну линию , то что бы вывести разные числа в разные разряды, необходимо зажигать разряды по очереди с соответствующими сегментами.

Индикатор имеет 12 цифровых выводов.



Инструкция по эксплуатации индикатора:

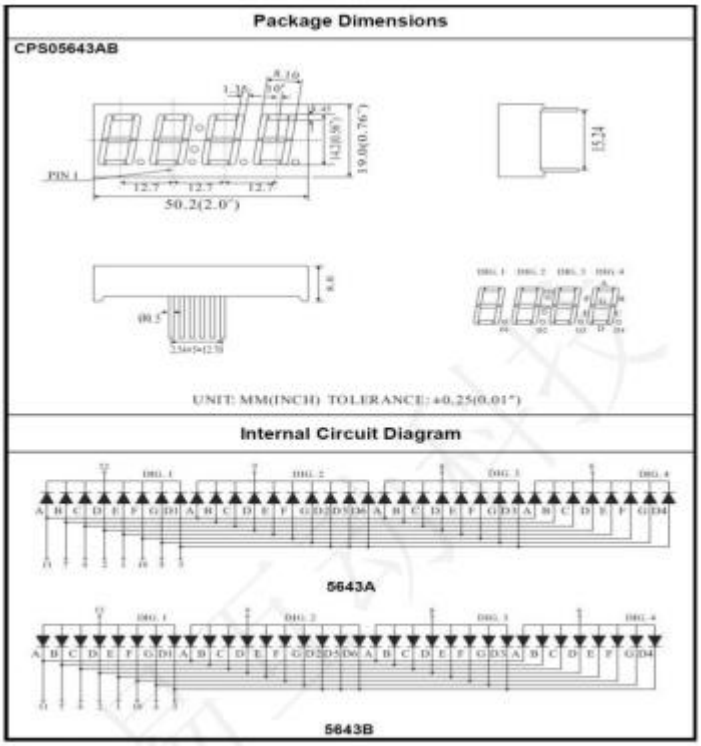
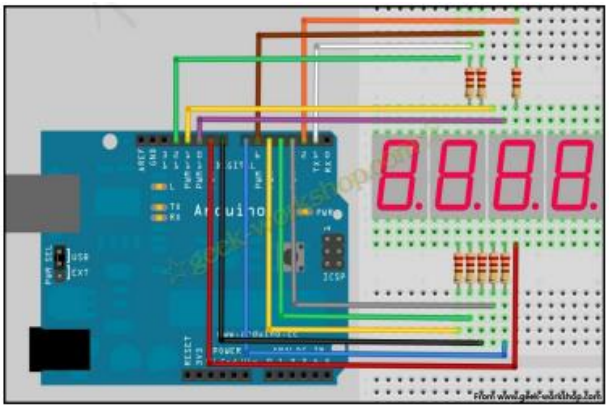


Схема подключения:



Код программы:

```
// сегменты индикатора
int a = 1;
int b = 2;
int c = 3;
int d = 4;
int e = 5;
int f = 6;
int g = 7;
int p = 8;
// катоды
int d4 = 9;
```

```
int d3 = 10;
int d2 = 11;
int d1 = 12;
// переменные
long n = 0;
int x = 100;
int del = 55; // точная настройка часов
void setup ()
{
	pinMode (d1, OUTPUT);
	pinMode (d2, OUTPUT);
	pinMode (d3, OUTPUT);
	pinMode (d4, OUTPUT);
	pinMode (a, OUTPUT);
	pinMode (b, OUTPUT);
	pinMode (c, OUTPUT);
	pinMode (d, OUTPUT);
	pinMode (e, OUTPUT);
	pinMode (f, OUTPUT);
	pinMode (g, OUTPUT);
	pinMode (p, OUTPUT);
}
void loop ()
{
	clearLEDs ();
	pickDigit (1);
	pickNumber ((n/x/1000)% 10);
	delayMicroseconds (del);
	clearLEDs ();
	pickDigit (2);
	pickNumber ((n/x/100)% 10);
	delayMicroseconds (del);
	clearLEDs ();
	pickDigit (3);
	dispDec (3);
	pickNumber ((n/x/10)% 10);
	delayMicroseconds (del);
	clearLEDs ();
	pickDigit (4);
	pickNumber (n / x% 10);
	delayMicroseconds (del);
	n + +;
	if (digitalRead (13) == HIGH) { n = 0; }
}
void pickDigit (int x) // объявляем pickDigit (x), нужна для открытия порта dx
{
	digitalWrite (d1, LOW);
	digitalWrite (d2, LOW);
```

```
	digitalWrite (d3, LOW);
	digitalWrite (d4, LOW);
	switch (x) {
		case 1:
			digitalWrite (d1, HIGH);
			break;
		case 2:
			digitalWrite (d2, HIGH);
			break;
		case 3:
			digitalWrite (d3, HIGH);
			break;
		default: digitalWrite (d4, HIGH); break;
	}
}
void pickNumber (int x) // объявляем pickNumber (x), показывает цифровое значение
{
	switch (x) {
		default:
			zero ();
			break;
		 case 1:
			one ();
			break;
		case 2:
			two ();
			break;
		case 3:
			three ();
			break;
		case 4:
			four ();
			break;
		case 5:
			five ();
			break;
		case 6:
			six ();
			break;
		case 7:
			seven ();
			break;
		case 8:
			eight ();
			break;
		case 9:
			nine ();
			break;
```

```
                }
}
void dispDec (int x) // чтобы отобразить десятичную точку
{
        digitalWrite (p, LOW);
}
void clearLEDs () // очистить экран
{
        digitalWrite (a, HIGH);
        digitalWrite (b, HIGH);
        digitalWrite (c, HIGH);
        digitalWrite (d, HIGH);
        digitalWrite (e, HIGH);
        digitalWrite (f, HIGH);
        digitalWrite (g, HIGH);
        digitalWrite (p, HIGH);
}
void zero () // сброс на начало 0:00
{
        digitalWrite (a, LOW);
        digitalWrite (b, LOW);
        digitalWrite (c, LOW);
        digitalWrite (d, LOW);
        digitalWrite (e, LOW);
        digitalWrite (f, LOW);
        digitalWrite (g, HIGH);
}
void one () // выставляем 1:00
{
        digitalWrite (a, HIGH);
        digitalWrite (b, LOW);
        digitalWrite (c, LOW);
        digitalWrite (d, HIGH);
        digitalWrite (e, HIGH);
        digitalWrite (f, HIGH);
        digitalWrite (g, HIGH);
}
void two () // выставляем  2:00
{
        digitalWrite (a, LOW);
        digitalWrite (b, LOW);
        digitalWrite (c, HIGH);
        digitalWrite (d, LOW);
        digitalWrite (e, LOW);
        digitalWrite (f, HIGH);
        digitalWrite (g, LOW);
}
void three () // выставляем 3:00
```

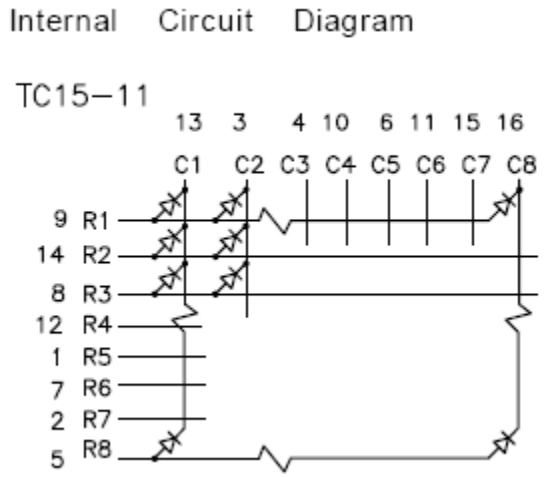
```
{
        digitalWrite (a, LOW);
        digitalWrite (b, LOW);
        digitalWrite (c, LOW);
        digitalWrite (d, LOW);
        digitalWrite (e, HIGH);
        digitalWrite (f, HIGH);
        digitalWrite (g, LOW);
}
void four () // выставляем  4:00
{
        digitalWrite (a, HIGH);
        digitalWrite (b, LOW);
        digitalWrite (c, LOW);
        digitalWrite (d, HIGH);
        digitalWrite (e, HIGH);
        digitalWrite (f, LOW);
        digitalWrite (g, LOW);
}
void five () // выставляем  5:00
{
        digitalWrite (a, LOW);
        digitalWrite (b, HIGH);
        digitalWrite (c, LOW);
        digitalWrite (d, LOW);
        digitalWrite (e, HIGH);
        digitalWrite (f, LOW);
        digitalWrite (g, LOW);
}
void six () // выставляем  6:00
{
        digitalWrite (a, LOW);
        digitalWrite (b, HIGH);
        digitalWrite (c, LOW);
        digitalWrite (d, LOW);
        digitalWrite (e, LOW);
        digitalWrite (f, LOW);
        digitalWrite (g, LOW);
}
void seven () // выставляем  7:00
{
        digitalWrite (a, LOW);
        digitalWrite (b, LOW);
        digitalWrite (c, LOW);
        digitalWrite (d, HIGH);
        digitalWrite (e, HIGH);
        digitalWrite (f, HIGH);
        digitalWrite (g, HIGH);
```

```
}
void eight () // выставляем 8:00
{
        digitalWrite (a, LOW);
        digitalWrite (b, LOW);
        digitalWrite (c, LOW);
        digitalWrite (d, LOW);
        digitalWrite (e, LOW);
        digitalWrite (f, LOW);
        digitalWrite (g, LOW);
}
void nine () // выставляем 9:00
{
        digitalWrite (a, LOW);
        digitalWrite (b, LOW);
        digitalWrite (c, LOW);
        digitalWrite (d, LOW);
        digitalWrite (e, HIGH);
        digitalWrite (f, LOW);
        digitalWrite (g, LOW);
}
```

После загрузки программы Вы сможете увидеть эффект течения времени.

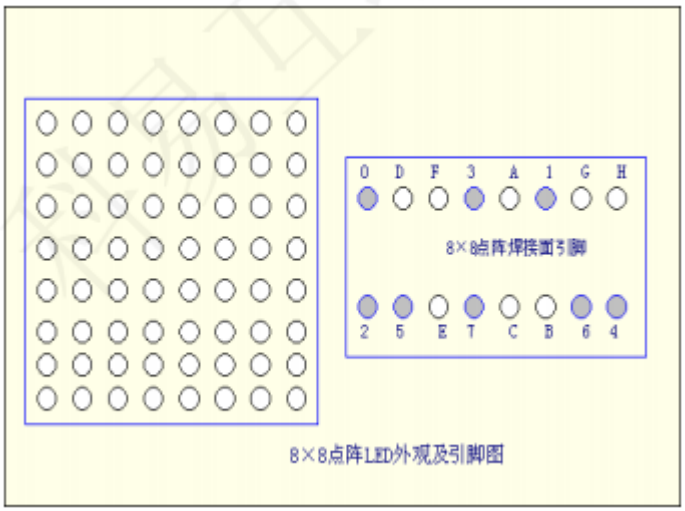
Урок 20 - Эксперимент - светодиодная матрица 8x8.

Диаграмма матрицы:



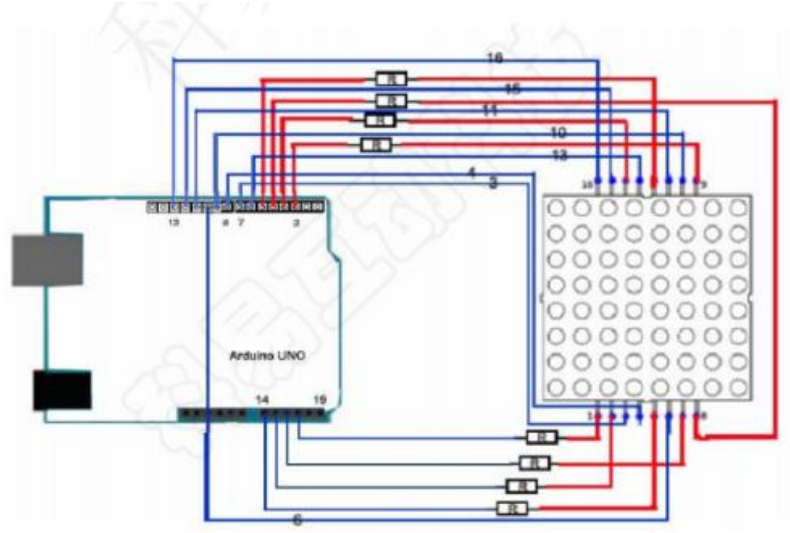
Снизу матрицы, находится два ряда пинов, со стандартным шагом 2.54, что позволяет удобно воткнуть светодиодную матрицу в две беспаечные макетные платы.

Карта выводов матрицы:

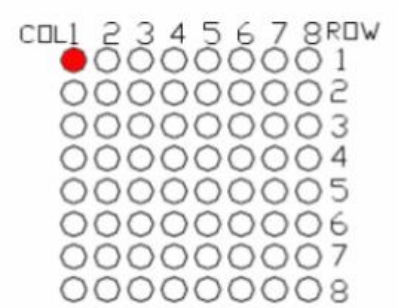


На рисунке изображена карта выводов 8 × 8 LED матрицы, в соответствии с ней строки и столбцы (точка - ряд). Подключив столбец на землю и строку на +5V, мы зажжем определенный светодиод.

Схема подключения к контроллеру:



Зажжем определенный светодиод:




```
// выводы для строк
const int row1 = 2; // 9  вывод строки
const int row2 = 3; //  14  вывод строки
const int row3 = 4; // 8  вывод строки
const int row4 = 5; // 12 вывод строки
const int row5 = 17; // 1 вывод строки
const int row6 = 16; // 7вывод строки
const int row7 = 15; // 2 вывод строки
const int row8 = 14; // 5 вывод строки
// выводы для столбцов
const int col1 = 6; // 13 вывод столбца
const int col2 = 7; // 3 вывод столбца
const int col3 = 8; // 4 вывод столбца
const int col4 = 9; // 10 вывод столбца
const int col5 = 10; // 6 вывод столбца
const int col6 = 11; // 11вывод столбца
const int col7 = 12; // 15 вывод столбца
const int col8 = 13; // 16 вывод столбца
void setup ()
{
        int i = 0; for (i = 2; i <18; i + +)
```

```
{
        pinMode (i, OUTPUT);
}
        pinMode (row5, OUTPUT);
        pinMode (row6, OUTPUT);
        pinMode (row7, OUTPUT);
        pinMode (row8, OUTPUT);
        for (i = 2; i <18; i + +)
        {
                digitalWrite (i, LOW);
        }
        digitalWrite (row5, LOW);
        digitalWrite (row6, LOW);
        digitalWrite (row7, LOW);
        digitalWrite (row8, LOW);
}
void loop ()
{
        int i; // загорается светодиод на первой строке в первом столбце
        digitalWrite (row1, HIGH);
        digitalWrite (row2, LOW);
        digitalWrite (row3, LOW);
        digitalWrite (row4, LOW);
        digitalWrite (row5, LOW);
        digitalWrite (row6, LOW);
        digitalWrite (row7, LOW);
        digitalWrite (row8, LOW);
        digitalWrite (col1, LOW);
        digitalWrite (col2, HIGH);
        digitalWrite (col3, HIGH);
        digitalWrite (col4, HIGH);
        digitalWrite (col5, HIGH);
        digitalWrite (col6, HIGH);
        digitalWrite (col7, HIGH);
        digitalWrite (col8, HIGH);
        delay (1000); // выключить все
        for (i = 2; i <18; i + +)
        {
                digitalWrite (i, LOW);
        }
        delay (1000);
}
```

Более сложный пример кода. Отображает буквы от “A” до I”:

```
# Define display_array_size 8 // Ascii 8x8 dot font
# Define data_null 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 // null char
```

```
# Define data_ascii_A 0x02, 0x0C, 0x18, 0x68, 0x68, 0x18, 0x0C, 0x02 / * "A", 0 * /
        / **
        ** "A"
        # Define A { //
                {0, 0, 0, 0, 0, 0, 1, 0}, // 0x02
                {0, 0, 0, 0, 1, 1, 0, 0}, // 0x0C
                {0, 0, 0, 1, 1, 0, 0, 0}, // 0x18
                {0, 1, 1, 0, 1, 0, 0, 0}, // 0x68
                {0, 1, 1, 0, 1, 0, 0, 0}, // 0x68
                {0, 0, 0, 1, 1, 0, 0, 0}, // 0x18
                {0, 0, 0, 0, 1, 1, 0, 0}, // 0x0C
                {0, 0, 0, 0, 0, 0, 1, 0} // 0x02
        }
        ** /
# Define data_ascii_B 0x00, 0x7E, 0x52, 0x52, 0x52, 0x52, 0x2C, 0x00 / * "B", 1 * /
# Define data_ascii_C 0x00, 0x3C, 0x66, 0x42, 0x42, 0x42, 0x2C, 0x00 / * "C", 2 * /
# Define data_ascii_D 0x00, 0x7E, 0x42, 0x42, 0x42, 0x66, 0x3C, 0x00 / * "D", 3 * /
# Define data_ascii_E 0x00, 0x7E, 0x52, 0x52, 0x52, 0x52, 0x52, 0x42 / * "E", 4 * /
# Define data_ascii_F 0x00, 0x7E, 0x50, 0x50, 0x50, 0x50, 0x50, 0x40 / * "F", 5 * /
# Define data_ascii_G 0x00, 0x3C, 0x66, 0x42, 0x42, 0x52, 0x16, 0x1E / * "G", 6 * /
# Define data_ascii_H 0x00, 0x7E, 0x10, 0x10, 0x10, 0x10, 0x7E, 0x00 / * "H", 7 * /
# Define data_ascii_I 0x00, 0x00, 0x00, 0x7E, 0x00, 0x00, 0x00, 0x00 / * "I", 8 * /
// Display array
byte data_ascii [] [display_array_size] = {
        data_null,
        data_ascii_A,
        data_ascii_B,
        data_ascii_C,
        data_ascii_D,
        data_ascii_E,
        data_ascii_F,
         data_ascii_G,
        data_ascii_H,
        data_ascii_I,
};
// The pin to control ROW
const int row1 = 2; // the number of the row pin 24
const int row2 = 3; // the number of the row pin 23
const int row3 = 4; / / the number of the row pin 22
const int row4 = 5; // the number of the row pin 21
const int row5 = 17; / / the number of the row pin 4
const int row6 = 16; // the number of the row pin 3
const int row7 = 15; // the number of the row pin 2
const int row8 = 14; // the number of the row pin 1
// выводы столбцов
const int col1 = 6; // the number of the col pin 20
const int col2 = 7; // the number of the col pin 19
const int col3 = 8; // the number of the col pin 18
```

```
const int col4 = 9; // the number of the col pin 17
const int col5 = 10; // the number of the col pin 16
const int col6 = 11; // the number of the col pin 15
const int col7 = 12; // the number of the col pin 14
const int col8 = 13; // the number of the col pin 13

void displayNum (byte rowNum, int colNum)
{
	int j;
	byte temp = rowNum;
	for (j = 2; j <6; j + +)
	{
		digitalWrite (j, LOW);
	}
	digitalWrite (row5, LOW);
	digitalWrite (row6, LOW);
	digitalWrite (row7, LOW);
	digitalWrite (row8, LOW);
	for (j = 6; j <14; j + +)
	{
		digitalWrite (j, HIGH);
	}
	switch (colNum)
	{
		case 1:
			digitalWrite (col1, LOW);
			break;
		case 2:
			digitalWrite (col2, LOW);
			break;
		case 3:
			digitalWrite (col3, LOW);
			break;
		case 4:
			digitalWrite (col4, LOW);
			break;
		case 5:
			digitalWrite (col5, LOW);
			break;
		case 6:
			digitalWrite (col6, LOW);
			break;
		case 7:
			digitalWrite (col7, LOW);
			break;
		case 8:
			digitalWrite (col8, LOW);
			break;
```

```
                default: break;
        }

        for (j = 1; j <9; j + +)
        {
                temp = (0x80) & (temp);
                if (temp == 0)
                {
                        temp = rowNum << j; continue;
                }
                switch (j)
                {
                        case 1:
                                digitalWrite (row1, HIGH);
                                break;
                        case 2:
                                digitalWrite (row2, HIGH);
                                break;
                        case 3:
                                digitalWrite (row3, HIGH);
                                break;
                        case 4:
                                digitalWrite (row4, HIGH);
                                break;
                        case 5:
                                digitalWrite (row5, HIGH);
                                break;
                        case 6:
                                digitalWrite (row6, HIGH);
                                break;
                        case 7:
                                digitalWrite (row7, HIGH);
                                break;
                        case 8:
                                digitalWrite (row8, HIGH);
                                break;
                        default:
                                break;
                }
                temp = rowNum << j;
        }
 }

void setup ()
{
        int i = 0; for (i = 2; i <18; i + +)
        {
                pinMode (i, OUTPUT);
```

```
        }
        for (i = 2; i <18; i + +)
        {
                digitalWrite (i, LOW);
        }
}
void loop ()
{
        int t1;
        int l;
        int arrage;
        for (arrage = 0; arrage <10; arrage + +)
        {
                for (l = 0; l <512; l + +)
                {
                        for (t1 = 0; t1 <8; t1 + +)
                        {
                                displayNum (data_ascii [arrage] [t1], (t1 +1));
                        }
                }
        }
 }
```

Урок 21 - Трехцветный LED RGB модуль.

Модуль имеет три выхода:
1. R, красный выход,
2. G, зеленый выход,
3. B, синий выход.

Возможности модуля:

Три сигнальных вывода могут быть запрограммированы микроконтроллером (R, G, B). Три цвета смешиваются для достижения полного цветового эффекта.

Код программы:

```
int ledPin = 13; // LED is connected to digital pin 13
int redPin = 11; // R petal on RGB LED module connected to digital pin 11
int greenPin = 9; // G petal on RGB LED module connected to digital pin 9
int bluePin = 10;  // B petal on RGB LED module connected to digital pin 10
void setup ()
{
        pinMode (ledPin, OUTPUT); // устанавливаем выводы в  output
        pinMode (redPin, OUTPUT); // sets the redPin to be an output
        pinMode (greenPin, OUTPUT); // sets the greenPin to be an output
        pinMode (bluePin, OUTPUT); // sets the bluePin to be an output
}
void loop () // run over and over again
{
        // Basic colors:
        color (255, 0, 0); // turn the RGB LED red
        delay (1000); // delay for 1 second
        color (0,255, 0); // turn the RGB LED green
        delay (1000); // delay for 1 second
        color (0, 0, 255); // turn the RGB LED blue
        delay (1000); // delay for 1 second
        // Example blended colors:
        color (255,255,0); // turn the RGB LED yellow
        delay (1000); // delay for 1 second
        color (255,255,255); // turn the RGB LED white
        delay (1000); // delay for 1 second
        color (128,0,255); // turn the RGB LED purple
        delay (1000); // delay for 1 second
        color (0,0,0); // turn the RGB LED off
        delay (1000); // delay for 1 second
}
void color (unsigned char red, unsigned char green, unsigned char blue) // the color
generating function
{
        analogWrite (redPin, 255-red);
```

```
        analogWrite (bluePin, 255-blue);
        analogWrite (greenPin, 255-green);
}
```

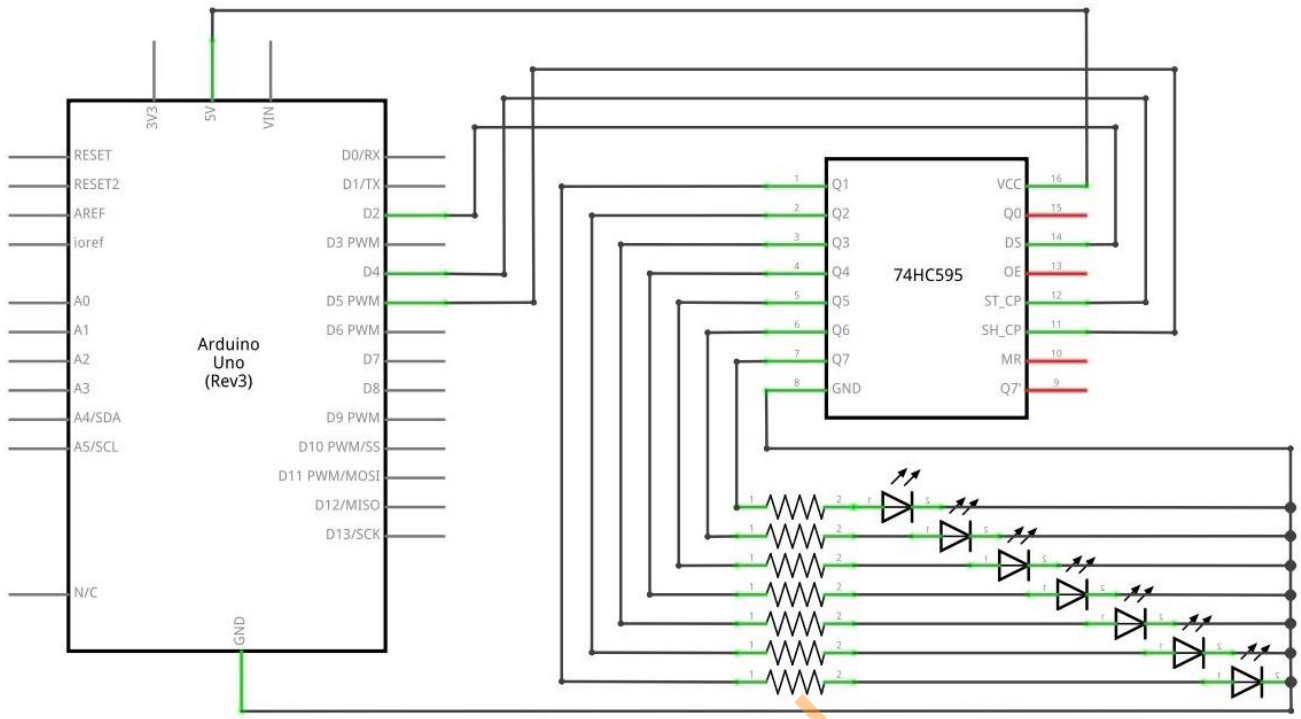
Урок 22 - Эксперимент 74HC595.

74HC595 представляет собой простой 8-разрядный регистр сдвига с тремя состояниями на выход. Мы будем использовать для управления 8 LED светодиодами. Зачем использовать 74HC595 для такой простой задачи, ведь можно и без него ? Ответ прост - Arduino имеет не так много выводов, а сдвиговый регистр позволяет значительно сократить количество используемых. Мы можем использовать всего 3 цифровых вывода для управления 8 светодиодами.

Необходимые компоненты:

1. Сдвиговый регистр 74HC595: 1 шт.
2. Красный LED светодиод 5мм: 4 шт.
3. Зеленый LED светодиод 5мм: 4 шт.
4. 220 Ω резистор: 8 шт.
5. Макетная плата: 1 шт.
6. Разноцветные перемычки для макетной платы

Схема подключения:



Пример кода:

```
int data = 2;
int clock = 4;
int latch = 5;
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;
```

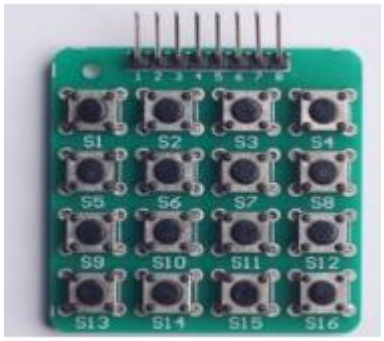
```
void setup ()
{
	pinMode (data, OUTPUT);
	pinMode (clock, OUTPUT);
	pinMode (latch, OUTPUT);
}
void loop ()
{
	int delayTime = 100;
	for (int i = 0; i <256; i + +)
	{
		updateLEDs (i);
		delay (delayTime);
	}
}
void updateLEDs (int value)
{
	digitalWrite (latch, LOW);
	shiftOut (data, clock, MSBFIRST, value);
	digitalWrite (latch, HIGH);
}
void updateLEDsLong (int value)
{
	digitalWrite (latch, LOW);
	for (int i = 0; i <8; i + +)
	{
		int bit = value & B10000000;
		value = value << 1;
		if (bit == 128)
		{
			digitalWrite (data, HIGH);
		}
		else
		{
			digitalWrite (data, LOW);
		}
		digitalWrite (clock, HIGH);
		delay (1);
		digitalWrite (clock, LOW);
	}
	digitalWrite (latch, HIGH);
}
int bits [] = {B00000001, B00000010, B00000100, B00001000, B00010000, B00100000,
B01000000, B10000000};
int masks [] = {B11111110, B11111101, B11111011, B11110111, B11101111, B11011111,
B10111111, B01111111};
void changeLED (int led, int state)
{
```

```
        ledState = ledState & masks [led];
        if (state == ON)
        {
                ledState = ledState | bits [led];
        }
        updateLEDs (ledState);
}
```

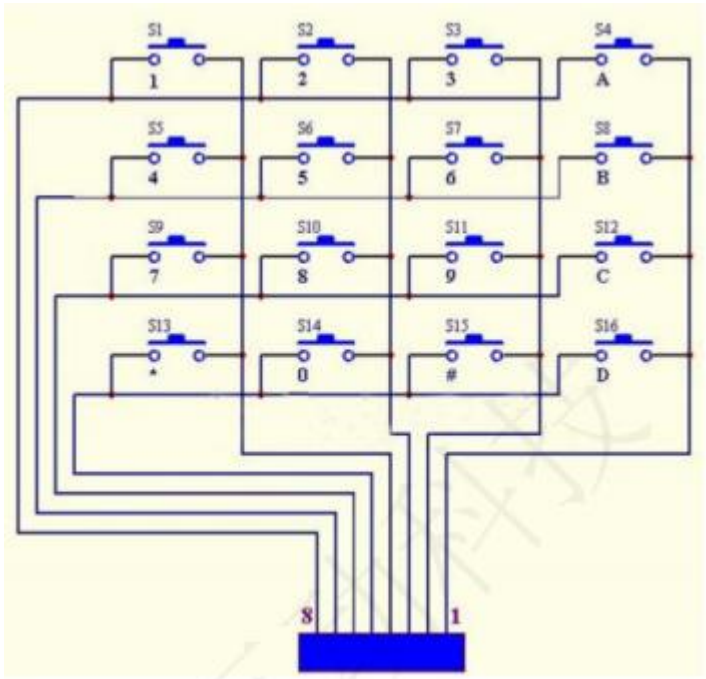
После загрузки программы мы можем увидеть восемь небольших вспышек света.

Урок 23 - Эксперимент модуль кнопок 4x4.

Внешний вид:



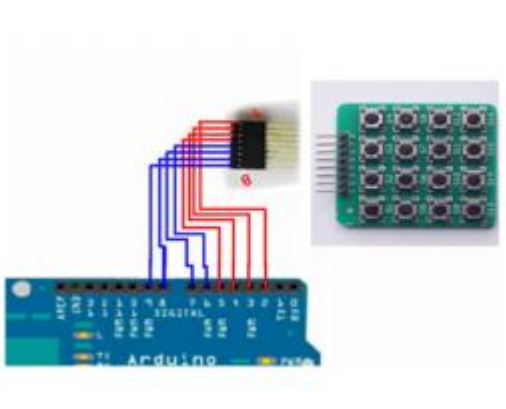
Принципиальная схема:



Необходимые компоненты:

1. Модуль кнопок 4x4: 1 шт.
2. Макетная плата: 1 шт.
3. Разноцветные перемычки для макетной платы

Подключим модуль к цифровым выводам Arduino:



Для работы данного программного кода необходимо подключить библиотеку “Keypad”:

```
# Include const byte ROWS = 4; // объявл. 4 строки
const byte COLS = 4; // 4 столбца
char keys [ROWS] [COLS] = { {'1 ', '2', '3 ',' A '}, {'4 ', '5', '6 ',' B '}, {'7 ', '8', '9 ',' C '}, {'*', '0 ',' #
',' D '} }; // карта клавиш, значения которых будут выводиться при нажатии
byte rowPins [ROWS] = {2,3,4,5}; // номера выводов к которым подключены столбцы
byte colPins [COLS] = {6,7,8,9};
// вызываем функцию  Keypad
keypad = Keypad (makeKeymap (keys), rowPins, colPins, ROWS, COLS);
void setup () {
	Serial.begin (9600);
}
void loop ()
{
	char key = keypad.getKey ();
	if (key! = NO_KEY)
	{
		Serial.println (key);
	}
}
```

После загрузки программы, по нажатии на кнопку, будет отображаться значение в мониторе порта.

Управление светом с помощью модуля клавиатуры 4x4. Используя предыдущую схема задействуем вывод 13 для управление светодиодом.

```
# Include const byte ROWS = 4; // объявл. 4 строки
const byte COLS = 4; // 4 столбца
char keys [ROWS] [COLS] = { {'1 ', '2', '3 ',' A '}, {'4 ', '5', '6 ',' B '}, {'7 ', '8', '9 ',' C '}, {'*', '0 ',' #
',' D '} }; // карта клавиш, значения которых будут выводиться при нажатии
byte rowPins [ROWS] = {2,3,4,5}; // номера выводов к которым подключены столбцы
byte colPins [COLS] = {6,7,8,9};
// вызываем функцию  Keypad
keypad = Keypad (makeKeymap (keys), rowPins, colPins, ROWS, COLS);
```

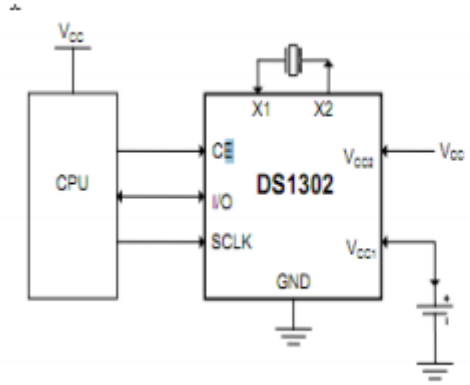
```
byte ledPin = 13;
boolean blink = false;
void setup ()
{
	Serial.begin (9600);
	pinMode (ledPin, OUTPUT); // sets the digital pin as output
	digitalWrite (ledPin, HIGH); // sets the LED on
	keypad.addEventListener (keypadEvent); // add an event listener for this keypad
}
void loop ()
{
	char key = keypad.getKey ();
	if (key! = NO_KEY) { Serial.println (key); }
	if (blink) {
		digitalWrite (ledPin,! digitalRead (ledPin));
		delay (100);
	}
}
// Take care of some special events
void keypadEvent (KeypadEvent key)
{
	switch (keypad.getState ())
	{
		case PRESSED:
			switch (key)
			{
				case '#':
					digitalWrite (ledPin,! digitalRead (ledPin));
					break;
				case '*':
					digitalWrite (ledPin,! digitalRead (ledPin));
					break;
			}
			break;
		case RELEASED:
			switch (key)
			{
				case '*':
					digitalWrite (ledPin,! digitalRead (ledPin));
					blink = false;
					break;
			}
			break;
		case HOLD:
			switch (key)
			{
				case '*':
					blink = true;
```

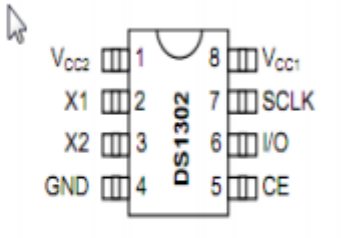
```
                                        break;
                                }
                                break;
                        }
}
```

После загрузки программы при нажатии на кнопку "*" - светодиод будет тухнуть, пока вы ее держите. При нажатии на "#" один раз свет загорается, если нажать еще раз - тухнет.

Урок 24 - Подключение модуля часов DS1302 к Arduino.

Модуль DS1302 поддерживает отображение года, месяца, дня, часа, минуты, секунды. Поддерживает подключение батареи для резервного сохранения данных. Можно подключить к Arduino при помощи трех проводов.





Техническая документация микросхемы:
https://datasheets.maximintegrated.com/en/ds/DS1302.pdf

Схема подключения:
CE (DS1302 pin5) -> Arduino D5
IO (DS1302 pin6) -> Arduino D6
SCLK (DS1302 pin7) -> Arduino D7
Vcc2 (DS1302 pin1) -> Arduino +5 v
GND (DS1302 pin4) -> Arduino GND

Пример кода:

```
# Include <stdio.h>
# Include <DS1302.h>

/ * Interface Definition
CE (DS1302 pin5) -> Arduino D5
IO (DS1302 pin6) -> Arduino D6
SCLK (DS1302 pin7) -> Arduino D7 * /

uint8_t CE_PIN = 5;
uint8_t IO_PIN = 6;
uint8_t SCLK_PIN = 7;

/ * переменные для буфера * /
char buf [50];
char day [10];

/ * данные для кеша * /
String comdata = "";
int numdata [7] = {0}, j = 0, mark = 0;
```

```
/ * создание объекта DS1302 * /
DS1302 rtc (CE_PIN, IO_PIN, SCLK_PIN);

void print_time () {
/ * получить время с  DS1302 * /
Time t = rtc.time ();
/ * название для недели  * /
memset (day, 0, sizeof (day));
        switch (t.day)
        {
                case 1:
                        strcpy (day, "Sunday");
                        break;
                case 2:
                        strcpy (day, "Monday");
                        break;
                case 3:
                        strcpy (day, "Tuesday");
                        break;
                case 4:
                        strcpy (day, "Wednesday");
                        break;
                case 5:
                        strcpy (day, "Thursday");
                        break;
                case 6:
                        strcpy (day, "Friday");
                        break;
                case 7:
                        strcpy (day, "Saturday");
                        break;
                } / * Формат для вывода* /
snprintf (buf, sizeof (buf), "% s% 04d-% 02d-% 02d% 02d:% 02d:% 02d", day, t.yr, t.mon,
t.date, t.hr, t. min, t.sec);
/ * Выводим данные в serial port * /
Serial.println (buf);
}

void setup ()
{
        Serial.begin (9600);
        rtc.write_protect (false);
        rtc.halt (false);
}
void loop ()
{
        / * Если на порту есть данные - помещает в переменную comdata * /
```

```
        while (Serial.available ()> 0)
        {
                comdata + = char (Serial.read ());
                delay (2);
                mark = 1;
        }
        / * значение  comdata разделенное запятыми преобразется в цифровое
        значение массива  numdata [] * /
        if (mark == 1)
        {
                Serial.print ("You inputed:");
                Serial.println (comdata);
                for (int i = 0; i <comdata.length (); i + +)
                {
                        if (comdata [i] == ',' | | comdata [i] == 0x10 | | comdata [i] == 0x13)
                        {
                                j + +;
                        }
                        else
                        {
                                numdata [j] = numdata [j] * 10 + (comdata [i] - '0 ');
                        }
                }
                / * Преобразует формат данных  DS1302 * /
                Time t (numdata [0], numdata [1], numdata [2], numdata [3], numdata [4],
                numdata [5],numdata [6]);
                rtc.time (t);
                mark = 0; j = 0;
                / * пустое значение, ждать следующего ввода */
                comdata = String ("");
                for (int i = 0; i <7; i + +) numdata [i] = 0;
        }
        / * выводим время * /
        print_time ();
        delay (1000);
}
```

Откройте отладчик порта Arduino и введите туда начальную дату в необходимом формате.

Формат:
Год, месяц, день, час, минута, секунда, день недели
Количество недель: воскресенье = 1, ПОНЕДЕЛЬНИК = 2, ... Суббота = 7
Например, сегодня в 11:23:40 по 17 ноября 2011 года четверг
Вы можете заполнить 2011,11,17,11,22,40,5

Урок 25 - Датчик уровня воды.

Датчик позволяет определить уровень воды. Подключается к аналоговому входу Arduino. Считываемые значения от 0 - при идеально сухом датчике, до 1023. Чем выше столб воды по датчику, тем значение выше. При падении уровня воды, значения уменьшаются.

Параметры:

1. Рабочее напряжение 3-5V.
2. Рабочий ток - менее 20 мА.
3. Тип датчика аналоговый.
4. Зона обнаружения: 40 мм х 16 мм.
5. Рабочая температура: 10 °C - 30 °C.
6. Размеры: 20 мм х 62 мм х 8 мм
7. Влажность: 10% - 90% без конденсации

Необходимые компоненты:

1. Arduino контроллер 1 шт.
2. Шилд под датчики для Arduino: 1 шт.
3. Датчик уровня воды
4. 3-пиновый провод для датчика
5. IR & LED Модуль (красный)

Датчик уровня воды подключается при помощи проводом с шилдом для датчиков.
Код программы:

```
int analogPin = 1; // датчик воды подключается к аналоговому порту
int led = 12; // LED подключается к 12 выводу
int val = 0; // переменная значения, по умолчанию 0
int data = 0; // переменная данных
void setup ()
{
        pinMode (led, OUTPUT); // выставляем пин в output
        Serial.begin (9600); // скорость передачи данных 9600
}
void loop ()
{
        val = analogRead (analogPin); // считать аналоговое значение с датчика
        if (val> 700)
        {
        // если значение больше700
        digitalWrite (led, HIGH); // зажигаем  LED
        }
        else
        {
                digitalWrite (led, LOW); // если меньше 700 тушим
        }
        data = val;
```

```
        Serial.println (data); // выводим значение
        delay (100);
}
```

После загрузки программы можно увидеть, что если датчик опущен вводу ниже определенного уровня - горит светодиод.

Урок 26 - Датчик температуры и влажности DHT11.

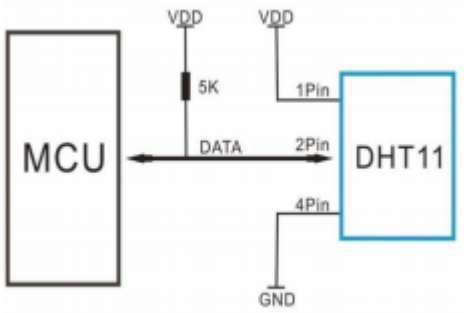
Стабильный датчик, имеет высокую точность, скоростью реакции, способность анти-помех.Имеет маленький размер, низкое энергопотребление.
Выполнен из двух частей — емкостного датчика влажности и термистора. Чип, находящийся внутри, выполняет аналого-цифровое преобразование и выдает цифровой сигнал, который можно считать с помощью любого микроконтроллера.

Технические параметры:

1. Рабочее напряжение: 3.3V-5V
2. Размер: 3.2 см * 1.4 см
3. Индикатор питания: красный
4. Выход: цифровой
5. Диапазон измерения температуры: 0 градусов - 50 градусов. Погрешность измерения температуры: + - 2 градуса
6. Диапазон измерения влажности: от 20% до 95% (диапазон 0 градусов -50 градусов). Погрешность измерения влажности: + - 5%

Примечание: долгосрочная температура хранения 10-40 ℃, влажность ниже 60%.

Схема подключения:



Необходимые компоненты:

1. Arduino контроллер 1 шт.
2. DHT 11 модуль

Код программы:

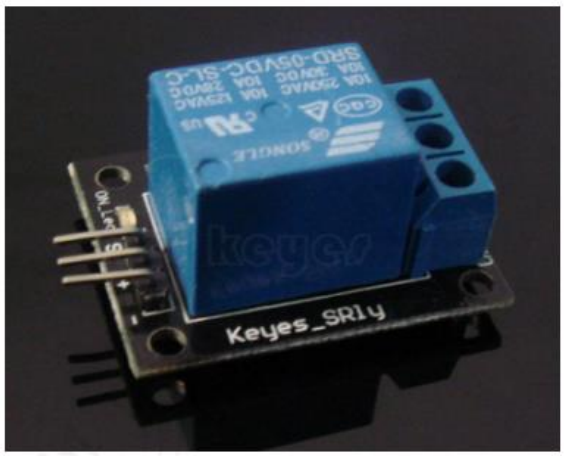
```
int DHpin = 8;
byte dat [5];
byte read_data ()
{
        byte data;
        for (int i = 0; i <8; i + +)
        {
                if (digitalRead (DHpin) == LOW)
                {
                        while (digitalRead (DHpin) == LOW);  // ожидание 50us;
                        delayMicroseconds (30); // определяет длительность высокого
                        уровня '0 или '1';
```

```
                        if (digitalRead (DHpin) == HIGH)
                                data | = (1 << (7-i)); // высокий фронт и низкий в отправке
                        while (digitalRead (DHpin) == HIGH); // данные '1 ', ждем
                        следующего приема
                        }
        } return data;
}
void start_test ()
{
        digitalWrite (DHpin, LOW); // шина на низкий уровень, сигнал запуска
        delay (30); // задержка, чтобы обнаружить сигнал
        digitalWrite (DHpin, HIGH);
        delayMicroseconds (40); // ожидание DHT11 ответа;
        pinMode (DHpin, INPUT);
        while (digitalRead (DHpin) == HIGH);
        delayMicroseconds (80); // DHT11 ответ80us;
        if (digitalRead (DHpin) == LOW);
        delayMicroseconds (80); // DHT11 80us задержка передачи данных;
        for (int i = 0; i <4; i + +) // получаем данные температуры и влажности, бит
четности не считается
        dat [i] = read_data ();
        pinMode (DHpin, OUTPUT);
        digitalWrite (DHpin, HIGH); // отправляем данные и ждем следующий сигнал
старт
}
void setup ()
{
        Serial.begin (9600);
        pinMode (DHpin, OUTPUT);
}
void loop ()
{
        start_test ();
        Serial.print ("Current humdity =");
        Serial.print (dat [0], DEC); // отображаем бит влажности, целое число
        Serial.print ('.');
        Serial.print (dat [1], DEC); // отображение влажности после запятой
        Serial.println ('%');
        Serial.print ("Current temperature =");
        Serial.print (dat [2], DEC); // отображаем бит температуры
        Serial.print ('.');
        Serial.print (dat [3], DEC); // отображение температуры после запятой
        Serial.println ('C'); delay (700);
}
```

После загрузки программы, открыв монитор порта, Вы увидите данные с датчика. На экране отображается комнатная температура и влажность.

Если взять датчик в ладонь и подержать некоторое время - температура будет увеличиваться. Так же, если подышать на датчик, должна измениться влажность.

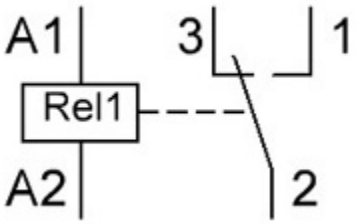
Урок 27 - Модуль реле.



Рано или поздно появляется желание поуправлять чем-то более мощным чем светодиод, либо создать нечто на подобие умного дома своими руками. В этом нам поможет такая радио деталь как реле.
Реле представляет собой электрически управляемое устройство. Он имеет систему управления.
Данное реле управляется напряжением 5V и способно коммутировать до 10A 30V DC и 10A 250V AC.

Реле имеет две раздельных цепи: цепь управления, представленная контактами A1, A2 и управляемая цепь, контакты 1, 2, 3. Цепи никак не связаны между собой.

Между контактами A1 и A2 установлен металлический сердечник, при протекании тока по которому к нему притягивается подвижный якорь(2). Контакты же 1 и 3 неподвижны. Стоит отметить что якорь подпружинен и пока мы не пропустим ток через сердечник, якорь будет удерживается прижатым к контакту 3. При подаче тока, как уже говорилось, сердечник превращается в электромагнит и притягивается к контакту 1. При обесточивании пружина снова возвращает якорь к контакту 3.



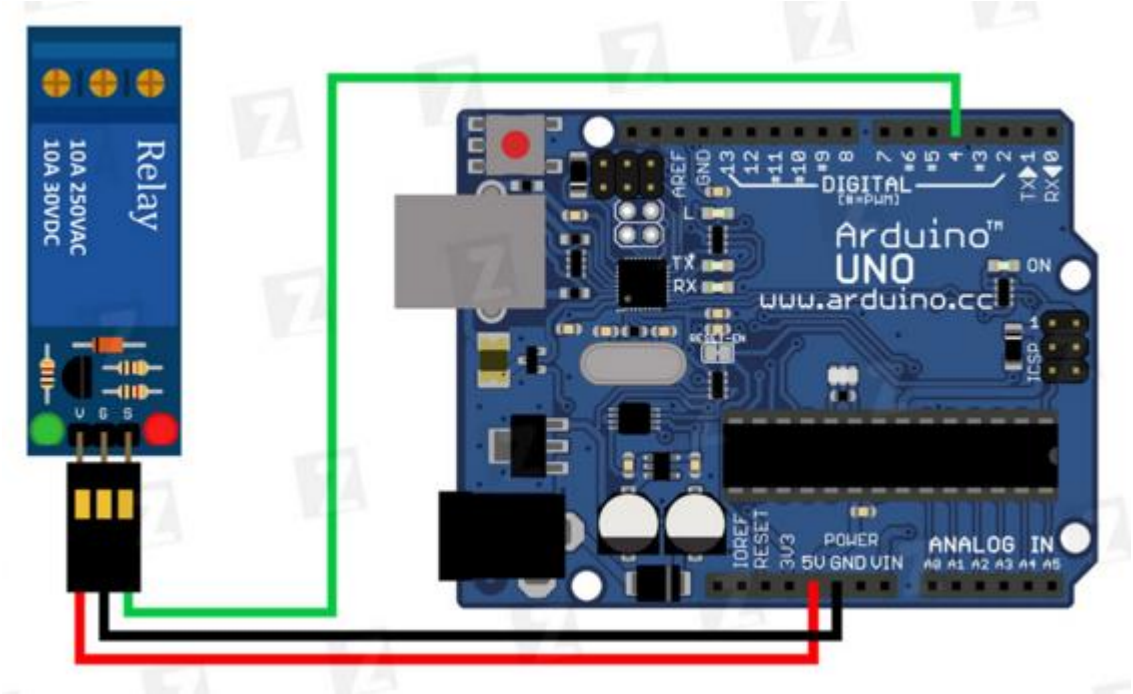
Модуль имеет 3 вывода (стандарта 2.54мм):

1. VCC: "+" питания
2. GND: "-" питания
3. IN: вывод входного сигнала

Схема подключения:

1. VCC на + 5 вольт на Ардуино.

2. GND на любой из GND пинов--- ардуино.
3. IN на любой из цифровых входов/выходов ардуино (в примерах подсоединено к 4).



Необходимые компоненты:

1. Arduino контроллер: 1 шт.
2. Реле модуль: 1 шт.
3. LED индикатор: 1 шт.
4. 330Ω резисторы: 1 шт.

Пример кода:

```
int relay = 10; // пин реле;
void setup ()
{
        pinMode (relay, OUTPUT); // выставляем пин в  OUTPUT
}
void loop ()
{
        digitalWrite (relay, HIGH); // реле проводит
        delay (1000);
        digitalWrite (relay, LOW); // выключен;
        delay (1000);
}
```

После загрузки программы Вы увидите как LED светодиод зажигается 1 сек. и тухнет.

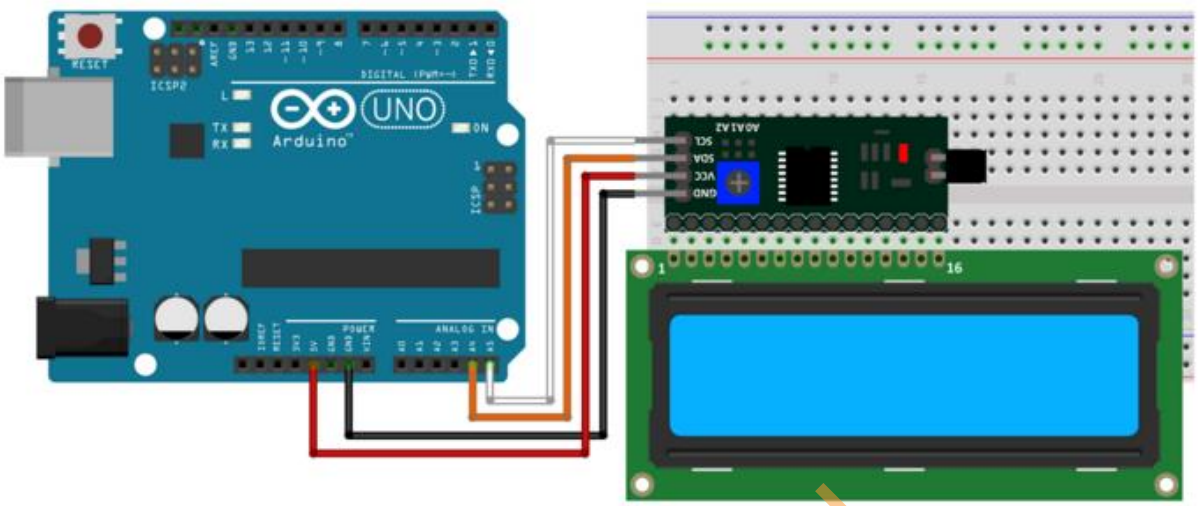
Урок 28 - Модуль I2C LCD1602.

LCD и LED дисплеи великолепная вещь, но у всех них есть одна большая проблема - для подключения требуется слишком много портов ввода/вывода.
Модуль I2C LCD1602 может быть хорошем решением этой проблемы, тем более он достаточно прост в использовании.

Технические характеристики:

1. Напряжение питания: 2.5-6V
2. Размер: 41.5 мм (длина) * 19 мм (ширина) * 15.3 мм (высота)
3. Вес: 5 г
4. Протокол - I2C
5. Регулировка контраста - при помощи встроенного потенциометра

Схема подключения:



Необходимые компоненты:

1. Arduino контроллер: 1 шт.
2. I2C LCD1602 модуль: 1 шт.

Проведем простой тест - выведем на экран надпись Robstore.ru
Пример кода:

```
#include <wire.h>
#include <LiquidCristal_I2C.h>
```

```
LiquidCrystal_I2C lcd(0x27,16,2);   /* Задаем адрес и размерность дисплея.
При использовании LCD I2C модуля с дисплеем 20x04 ничего в коде изменять не
требуется, следует только задать правильную размерность */

void setup()
{
  lcd.init();                      // Инициализация lcd
  lcd.backlight();                 // Включаем подсветку
  // Курсор находится в начале 1 строки
  lcd.print("Hello, world!");      // Выводим текст
  lcd.setCursor(0, 1);             // Устанавливаем курсор в начало 2 строки
  lcd.print("robstore.ru");        // Выводим текст
}

void loop()
{
}
```

Для работы программы необходимо подключать заголовки:Wire, LiquidCrystal_I2C.
После загрузки программы надписи выведутся на дисплей.

Урок 29 - Шаговый двигатель.

Управление шаговым двигателем представляет собой электрический импульс - угловое смещение привода. Попросту говоря: При приеме сигнала импульсного шагового привода, он будет управлять шаговым двигателем, чтобы установить направление вращения фиксированного угла (и угол шага). Можно контролировать количество импульсов для управления угловым смещением, за тем, чтобы достичь цели точного позиционирования; и вы также можете контролировать частоту импульсов для управления скоростью вращения двигателя и ускорение таким образом, чтобы достичь точной скорости.

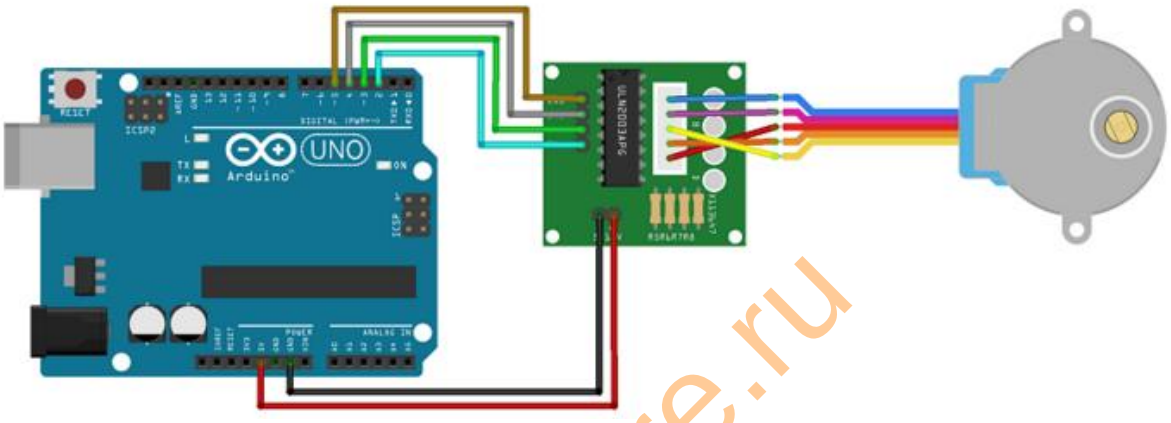
28BYJ-48 - Один из самых распространенных шаговых двигателях. Небольшой размер и сравнительно большой крутящий момент.

Технические характеристики:

1. Рабочее напряжение: 5V DC
2. Количество фаз: 4
3. Входное сопротивление: 31 Ом
4. Self Positioning Torque: ~34mN·m
5. Friction Torque: 60~120mN·m
6. Pull-in Torque: ~30mN·m
7. Уровень шума: 35dB (без нагрузки)

Схема подключения:

1. Подключите двигатель к специальному разъёму на плате управления
2. Следующим шагом соедините пины IN1, IN2, IN3, IN4 и выходы Arduino D8, D9, D10, D11 соответственно.
3. Подайте питание на пины "+" и "-" модуля управления двигателем.



Управление двигателем осуществляется при помощи микросхемы (UL2003).

Примеры кода:

/ *

* Stepper motor rotating follower potentiometer

```
* (Or other sensors) using the input analog port number 0
* Use the arduino IDE comes Stepper.h library files
* /
# Include <Stepper.h>
// число шагов вращения
# Define STEPS 100
// число шагов и пины
Stepper stepper (STEPS, 8, 9, 10, 11);
// переменные
int previous = 0;
void setup ()
{
// скорость двигателя 90 шагов в минуту
stepper.setSpeed (90);
}
void loop ()
{
// считываем значение
int val = analogRead (0);
// переход к текущему значению минус число шагов
stepper.step (val - previous);
// сохраняем
previous = val;
}
```

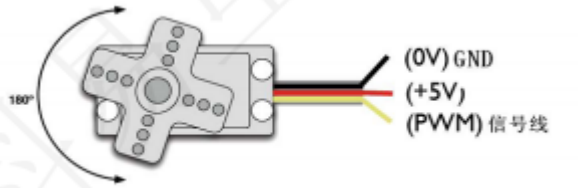
Урок 30 - Эксперемент. Управление серводвигателем.

Сервопривод - мотор-редуктор, способный поворачивать выходной вал строго в заданное положение (на угол) и удерживать его там, вопреки сопротивлениям и возмущениям недружелюбной среды.

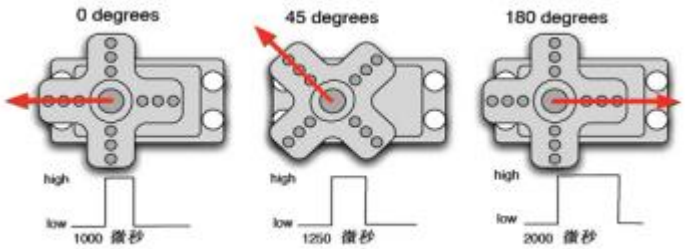
Сервопривод SG90 используется в основном для управления небольшими легкими механизмами, угол поворота которых ограничен диапазоном от 0 до 180 градусов.

Чтобы чутко отслеживать положения вала и воспринимать сигналы управления сервоприводы имеют на борту “электронику”:
Вал с качалкой жёстко связан с двигателем переменного резистора. Резистор подключен к схеме контроля и сообщает ей своим текущим сопротивлением о текущем положении вала. На схему, в свою очередь, поступают сигналы управления, сообщающие в какое положение нужно повернуть выходной вал. Схема подаёт питание на моторчик и крутит им.

В простейших аналоговых сервах, угол задаётся длительностью импульсов идуших с определённой частотой. В более продвинутых — используется протокол I2C. У сервопривода есть три провода:



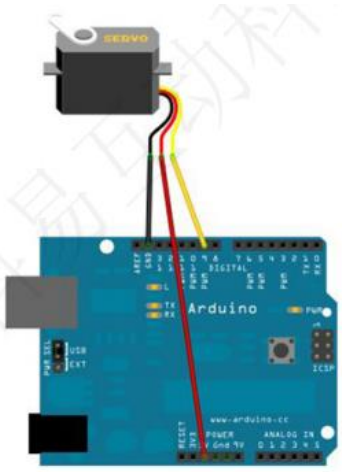
Угол поворота задается при помощи ШИМ (широтно-импульсной модуляции). Длительность импульса соответствует углу. У различных моделей серводвигателей эти значения могут различаться.



Необходимые компоненты:

1. Сервопривод: 1 шт.
2. Макетная плата: 1 шт.

Сервопривод в моменты работы потребляет значительное количество энергии. Для подключения серводвигателей рекомендуется использовать внешний источник питания, т.к. возможности платы Arduino ограничены.



Сигнальный провод подключите к 9 выводу платы Arduino.

```
int servopin = 9 ; // цифровой вывод для подключения сигнального провода
int myangle ; // переменная угла
int pulsewidth ; // ширина импульса
int val;
void servopulse (int servopin, int myangle) // функция импульса
{
        pulsewidth = (myangle * 11) +500 ; // перевод значения угра в ширину импульса
        500-2480
        digitalWrite (servopin, HIGH) ; // включить серво
        delayMicroseconds (pulsewidth) ; // задать значение ширины импульса
        digitalWrite (servopin, LOW) ; // выключить серво
        delay (20-pulsewidth/1000);
}
void setup ()
{
        pinMode (servopin, OUTPUT) ; // установить вывод в OUTPUT
        Serial.begin (9600) ; // скорость соединения 9600
        Serial.println ("servo = o_seral_simple ready");
}
void loop () // нужно вводить число от 0 до 9, которое соответствует от  0-180
{
        val = Serial.read () ; // считать введеное значение
        if (val> '0 '&& val <= '9')
        {
                val = val-'0 ';// преобразовать в числовые переменные
                val = val * (180/9) ; // преобразовать значение
                Serial.print ("moving servo to");
                Serial.print (val, DEC);
                Serial.println ();
                for (int i = 0; i <= 50; i + +) // дать достаточно времени, чтобы повернулось
                {
```

```
                    servopulse (servopin, val) ; // вызов функции
                }
        }
}
```

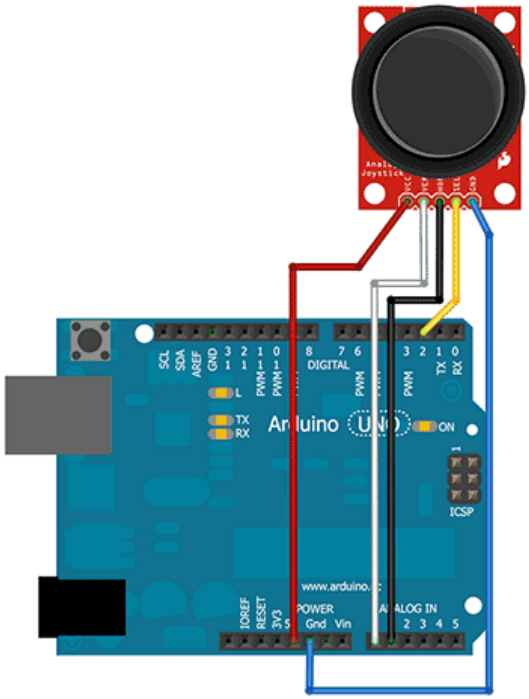
Первый пример кода был приведен для понимания принципа работы серводвигателей. Во втором примере будет использоваться библиотека для удобного управления сервоприводом.

```
# Include <Servo.h>  // подключение библиотеки управления
Servo myservo ; // переменная серво
void setup ()
{
        myservo.attach (9) ; // определяет параметры серво (недостаток - работает только с 2 пинами: 9,10)
}
void loop ()
 {
        myservo.write (90) ; //  установить угол поворота
 }
```

В данном эксперименте с серводвигателями было приведено два примера кода. Каждый из них имеет свои достоинства и недостатки. Каким пользоваться - решать Вам.

Урок 31 - Модуль игрового джойстика.

Модуль джойстика имеет ось X, Y и кнопку - ось Z. На модуле 5 пинов: Vcc, Ground, X, Y, Key.
Вы можете получать значения с выводов модуля и , в зависимости от полученных значений, реализовывать ответное действие. Например, движение и поровот робота.
Z необходимо подключить к цифровому выводу, а X, Y к аналоговому.



Выводы X, Y это обычные потенциометры, работу которых мы рассматривали в более ранних уроках. Z - это цифровая кнопка, которая имеет два состояния - вкл/выкл (1/0).

Необходимые компоненты:

1. Arduino контроллер: 1 шт.
2. модуль джойстика: 1 шт.

Пример работы с джойстиком:

```
int sensorPin = 5;
int value = 0;
void setup ()
{
        pinMode (7, OUTPUT);
        Serial.begin (9600);
}
void loop ()
{
        value = analogRead (0);
        Serial.print ("X:");
```

```
        Serial.print (value, DEC);
        value = analogRead (1);
        Serial.print ("| Y:");
        Serial.print (value, DEC);
        value = digitalRead (7);
        Serial.print ("| Z:");
        Serial.println (value, DEC);
        delay (100);
}
```

Приведенная выше программа отображает значения джойстика на экране компьтера.

Урок 32 - Инфракрасное дистанционное управление.

Модуль ИК Приемника в связке и ИК пультом дистанционного управления позволит легко реализовать дистанционное управление платой Arduino.
Инфракрасный сигнал дистанционного управления представляют собой последовательность импульсов двоичного кода. Для передачи сигнала сигнал модулируется на определенной частоте, а на другой стороне демодулируется на другой стороне.

Модуль оборудован трехпиновым разъемом стандарта 2.54мм:

1. подключается к выводу GND
2. подключается к выводу +5V
3. подключается к цифровому выводу (в примере D2)

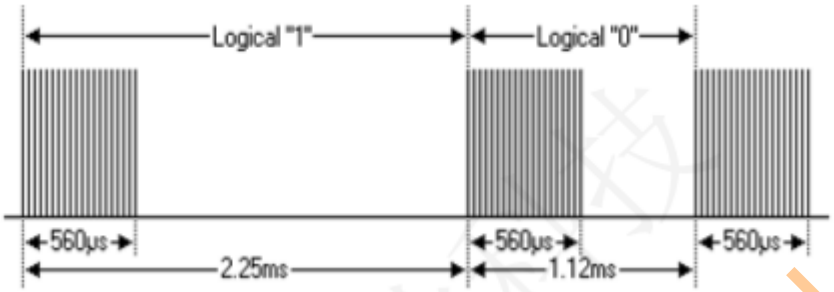
Необходимые компоненты:

1. ИК-пульт дистанционного управления: 1 шт.
2. инфракрасный приемник: 1 шт.
3. LED светодиоды: 6 шт.
4. резистор 220Ω: 6 шт.
5. разноцветные перемычки: несколько

Для декодирования используется NEC протокол:

- 8 адресных битов, 8-битная команда
- биты и командные биты надежности передаются в два раза
- Импульсная установка модуляции
- частота несущей 38KHz
- длина 1.125ms 2.25ms

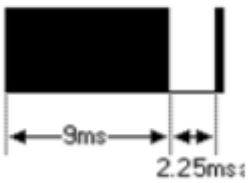
логический 0 и 1 определяется, как показано ниже:



Кнопка нажата - передается испульс:

На рисунке выше показана типичная последовательность протокола импульсов NEC. Сообщение генерирует высокий уровень 9ms запуска, а затем низкий уровень 4.5ms, а затем с помощью кода адреса и кода команды. Адрес и передача команды дважды.Общее время передачи постоянно.

Команда отправляется один раз, даже если кнопки на пульте дистанционного управления все еще нажата. Когда кнопка нажата более 110 мс, отправляется каждые 110 мс высокий уровень и низкий уровень.



Пример кода:

```
# Include <IRremote.h>
int RECV_PIN = 11;
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;
long on1 = 0x00FFA25D;
long off1 = 0x00FFE01F;
long on2 = 0x00FF629D;
long off2 = 0x00FFA857;
long on3 = 0x00FFE21D;
long off3 = 0x00FF906F;
long on4 = 0x00FF22DD;
long off4 = 0x00FF6897;
long on5 = 0x00FF02FD;
long off5 = 0x00FF9867;
long on6 = 0x00FFC23D;
long off6 = 0x00FFB047;
IRrecv irrecv (RECV_PIN);
decode_results results;
// вызываем перед IRrecv :: decode ()
// Void dump (void * v) {
```

```
// Decode_results * results = (decode_results *) v
void dump (decode_results * results) {
	int count = results-> rawlen;
	if (results-> decode_type == UNKNOWN)
	{
		Serial.println ("Could not decode message");
	}
	else
	{
		if (results-> decode_type == NEC)
		{
			Serial.print ("Decoded NEC:");
		}
		else if (results-> decode_type == SONY)
		{
			Serial.print ("Decoded SONY:");
		}
		else if (results-> decode_type == RC5)
		{
			Serial.print ("Decoded RC5:");
		}
		else if (results-> decode_type == RC6)
		{
			Serial.print ("Decoded RC6:");
		}
		Serial.print (results-> value, HEX);
		Serial.print ("(");
		Serial.print (results-> bits, DEC);
		Serial.println ("bits)");
	}
	Serial.print ("Raw (");
	Serial.print (count, DEC);
	Serial.print ("):");
	for (int i = 0; i <count; i + +)
	{
		if ((i% 2) == 1) {
			Serial.print (results-> rawbuf [i] * USECPERTICK, DEC);
		}
		else
		{
			Serial.print (- (int) results-> rawbuf [i] * USECPERTICK, DEC);
		}
		Serial.print ("");
	}
	Serial.println ("");
}
void setup ()
{
```

```
	pinMode (RECV_PIN, INPUT);
	pinMode (LED1, OUTPUT);
	pinMode (LED2, OUTPUT);
	pinMode (LED3, OUTPUT);
	pinMode (LED4, OUTPUT);
	pinMode (LED5, OUTPUT);
	pinMode (LED6, OUTPUT);
	pinMode (13, OUTPUT);
	Serial.begin (9600);
	irrecv.enableIRIn (); // запустить приемник
}
int on = 0;
unsigned long last = millis ();
void loop ()
{
	if (irrecv.decode (& results))
	{
		// Если это было, по крайней мере 1/4 секунды с момента последнего
		// ИК-приемник, переключает реле
		if (millis () - last> 250)
		{
			on =! on;
			// DigitalWrite (8, on? HIGH: LOW);
			digitalWrite (13, on? HIGH: LOW);
			dump (& results);
		}
		if (results.value == on1)
			digitalWrite (LED1, HIGH);
		if (results.value == off1)
			digitalWrite (LED1, LOW);
		if (results.value == on2)
			digitalWrite (LED2, HIGH);
		if (results.value == off2)
			digitalWrite (LED2, LOW);
		if (results.value == on3)
			digitalWrite (LED3, HIGH);
		if (results.value == off3)
			digitalWrite (LED3, LOW);
		if (results.value == on4)
			digitalWrite (LED4, HIGH);
		if (results.value == off4)
			digitalWrite (LED4, LOW);
		if (results.value == on5)
			digitalWrite (LED5, HIGH);
		if (results.value == off5)
			digitalWrite (LED5, LOW);
		if (results.value == on6)
			digitalWrite (LED6, HIGH);
```

```
                if (results.value == off6)
                        digitalWrite (LED6, LOW);
                last = millis ();
                irrecv.resume (); // Получить следующее значение
        }
}
```

Импульс передается, а на другой стороне декодируется. В зависимости от полученных данных зажигаются светодиоды.

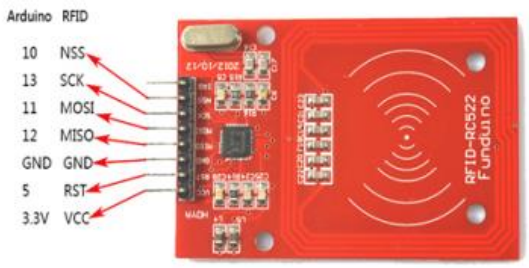
Урок 33 - Эксперимент. RFID считыватель.

RF технология также упоминается как RFID, называется радиочастотной идентификацией.

Как работает:

Состоит из RFID меток, RFID считывателей с антеннами и хост-компьютера. RFID метка состоит из 2 частей - чипа для хранения и обработки информации и антенны для приема и передачи данных. RFID считыватель дает возможность получать и обрабатывать данные с RFID метки, а также записывать данные на RFID метку.

Модуль RFID для Arduino должен быть подключен к модулю питания 3.3в, иначе может сгореть.




```
# Include <SPI.h>
# Define uchar unsigned char
# Define uint unsigned int
// максимальная длина массива
# Define MAX_LEN 16
```

```
// установить пины
const int chipSelectPin = 10; // пин Arduino uno
const int chipSelectPin = 53; // пин Arduino  mega 2560,1280
const int NRSTPD = 5;
// MF522 слова команды
# Define PCD_IDLE 0x00 // Бездействие; отменить текущую команду
# Define PCD_AUTHENT 0x0E // ключ аутентификации
# Define PCD_RECEIVE 0x08 // получать данные
# Define PCD_TRANSMIT 0x04 // передавать данные
# Define PCD_TRANSCEIVE 0x0C // Передача и прием данных
# Define PCD_RESETPHASE 0x0F // Reset
# Define PCD_CALCCRC 0x03 // CRC подсчет
// Mifare_One card командное слово
# Define PICC_REQIDL 0x26 // поиск антены, не должно уходить в спящий режим
# Define PICC_REQALL 0x52 // найти все антенны
# Define PICC_ANTICOLL 0x93 // anti-collision
# Define PICC_SElECTTAG 0x93 // election card
# Define PICC_AUTHENT1A 0x60 // authentication key A
# Define PICC_AUTHENT1B 0x61 // authentication key B
# Define PICC_READ 0x30 // читать блок
# Define PICC_WRITE 0xA0 // записать блок
# Define PICC_DECREMENT 0xC0
# Define PICC_INCREMENT 0xC1
# Define PICC_RESTORE 0xC2 // передать блок данных в буфер
# Define PICC_TRANSFER 0xB0 // сохранить данные в буфере
# Define PICC_HALT 0x50 // Sleep
// MF522 код ошибки возвращается, когда
# Define MI_OK 0
# Define MI_NOTAGERR 1
# Define MI_ERR 2
// ------------------ MFRC522 регистр---------------
// Page 0: команды и статусы
# Define Reserved00 0x00
# Define CommandReg 0x01
# Define CommIEnReg 0x02
# Define DivlEnReg 0x03
# Define CommIrqReg 0x04
# Define DivIrqReg 0x05
# Define ErrorReg 0x06
# Define Status1Reg 0x07
# Define Status2Reg 0x08
# Define FIFODataReg 0x09
# Define FIFOLevelReg 0x0A
# Define WaterLevelReg 0x0B
# Define ControlReg 0x0C
# Define BitFramingReg 0x0D
# Define CollReg 0x0E
# Define Reserved01 0x0F
```

// Page 1: Command
Define Reserved10 0x10
Define ModeReg 0x11
Define TxModeReg 0x12
Define RxModeReg 0x13
Define TxControlReg 0x14
Define TxAutoReg 0x15
Define TxSelReg 0x16
Define RxSelReg 0x17
Define RxThresholdReg 0x18
Define DemodReg 0x19
Define Reserved11 0x1A
Define Reserved12 0x1B
Define MifareReg 0x1C
Define Reserved13 0x1D
Define Reserved14 0x1E
Define SerialSpeedReg 0x1F
// Page 2: CFG
Define Reserved20 0x20
Define CRCResultRegM 0x21
Define CRCResultRegL 0x22
Define Reserved21 0x23
Define ModWidthReg 0x24
Define Reserved22 0x25
Define RFCfgReg 0x26
Define GsNReg 0x27
Define CWGsPReg 0x28
Define ModGsPReg 0x29
Define TModeReg 0x2A
Define TPrescalerReg 0x2B
Define TReloadRegH 0x2C
Define TReloadRegL 0x2D
Define TCounterValueRegH 0x2E
Define TCounterValueRegL 0x2F
/ / Page 3: TestRegister
Define Reserved30 0x30
Define TestSel1Reg 0x31
Define TestSel2Reg 0x32
Define TestPinEnReg 0x33
Define TestPinValueReg 0x34
Define TestBusReg 0x35
Define AutoTestReg 0x36
Define VersionReg 0x37
Define AnalogTestReg 0x38
Define TestDAC1Reg 0x39
Define TestDAC2Reg 0x3A
Define TestADCReg 0x3B
Define Reserved31 0x3C

```
# Define Reserved32 0x3D
# Define Reserved33 0x3E
# Define Reserved34 0x3F
// ---------------------------------------------
// 4 байта серийный номер карты, первые 5 байт для контрольной суммы байт
uchar serNum [5];
uchar writeDate [16] = {'T', 'e', 'n', 'g', '', 'B', 'o', 0, 0, 0, 0, 0, 0, 0, 0, 0};
// Sector A password,  16 секторов, каждый сектор пароль  6Byte
uchar sectorKeyA [16] [16] = {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
};
uchar sectorNewKeyA [16] [16] = {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xff, 0x07, 0x80, 0x69, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xff, 0x07, 0x80, 0x69, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
};
void setup () {
        Serial.begin (9600); // RFID reader SOUT pin connected to Serial
        RX pin at 2400bps
        // Start SPI library:
        SPI.begin ();
        pinMode (chipSelectPin, OUTPUT); // установить пин в OUTPUT
        digitalWrite (chipSelectPin, LOW); // активировать считывание RFID
        pinMode (NRSTPD, OUTPUT); //установить пин в OUTPUT
        digitalWrite (NRSTPD, HIGH);
        MFRC522_Init ();
}
void loop ()
{
        uchar i, tmp;
        uchar status;
        uchar str [MAX_LEN];
        uchar RC_size;
        uchar blockAddr; // выберите рабочий блок адресов 0 до 63
        // поиск карты
        status = MFRC522_Request (PICC_REQIDL, str);
        if (status == MI_OK)
        {
        }
        // Anti-collision, возвращает серийный номер карты 4 байта
        status = MFRC522_Anticoll (str);
        memcpy (serNum, str, 5);
        if (status == MI_OK)
        {
                Serial.println ("The card's number is:");
                Serial.print (serNum [0], BIN);
```

```
        Serial.print (serNum [1], BIN);
        Serial.print (serNum [2], BIN);
        Serial.print (serNum [3], BIN);
        Serial.print (serNum [4], BIN);
        Serial.println ("");
}
// Выбор карты
RC_size = MFRC522_SelectTag (serNum);
if (RC_size! = 0)
{}
// запись данных карты
blockAddr = 7; / / data block 7
status = MFRC522_Auth (PICC_AUTHENT1A, blockAddr, sectorKeyA [blockAddr /
4],serNum); // сертификация
if (status == MI_OK)
{
        // запись данных
        status = MFRC522_Write (blockAddr, sectorNewKeyA [blockAddr / 4]);
        Serial.print ("set the new card password, and can modify the data of
        the Sector: ");
        Serial.print (blockAddr / 4, DEC);
        // запись данных
        blockAddr = blockAddr - 3;
        status = MFRC522_Write (blockAddr, writeDate);
        if (status == MI_OK)
        {
        Serial.println ("OK!");
        }
}
// считывание
blockAddr = 7; / / data block 7
status = MFRC522_Auth (PICC_AUTHENT1A, blockAddr,
sectorNewKeyA [blockAddr / 4], serNum); // сертификация
if (status == MI_OK)
{
        // считывание данных
        blockAddr = blockAddr - 3;
        status = MFRC522_Read (blockAddr, str);
        if (status == MI_OK)
        {
                Serial.println ("Read from the card, the data is:");
                for (i = 0; i <16; i + +)
                {
                        Serial.print (str [i]);
                }
                Serial.println ("");
        }
}
```

```
        Serial.println ("");
        MFRC522_Halt (); // Команда в спящий режим
}
/ *
* Имя функции: Write_MFRC5200
* Описание: MFRC522 of a register to write a byte of data
* Параметры: addr - register address; val - the value to be written
* Возвращаемоезначение: None
* /
```

В данном эксперименте, если поднести карту к считывателю - на карту записываются данные, потом они считываются и выводятся на экран.