МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ ДЕТЕЙ

КАФЕДРЫ «СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ И ПОИСКОВОГО КОНСТРУИРОВАНИЯ»

РОБОФАБРИКА

Основы работы с Arduino

Методические указания к лабораторной работе

Волгоград

2020

УДК 621.1.016 (075)

Р е ц е н з е н т

К.В. Приходьков

Печатается по решению редакционно-издательского совета

Волгоградского государственного технического университета

Основы работы с Arduino : метод. указания к лабораторной работе / сост.: Р.О.Фокин; ВолгГТУ. – Волгоград, 2020. – 30 с.

Излагаются основные принципы работы с платой Arduino и подключаемыми к ней электронными компонентами. Для работы с материалом необходимо наличие компьютера с установленной программой Arduino IDE, плата Arduino и представленные в лабораторной работе электронные компоненты.

Предназначены в помощь слушателям центров дополнительного образования, изучающим программы, связанные с проектированием, сборкой и программированием роботов.

Ил. 20. Табл. . Библиогр.: назв.

Учимся работать с Ардуино

В нашем курсе обучения мы будем работать с платой Arduino Uno

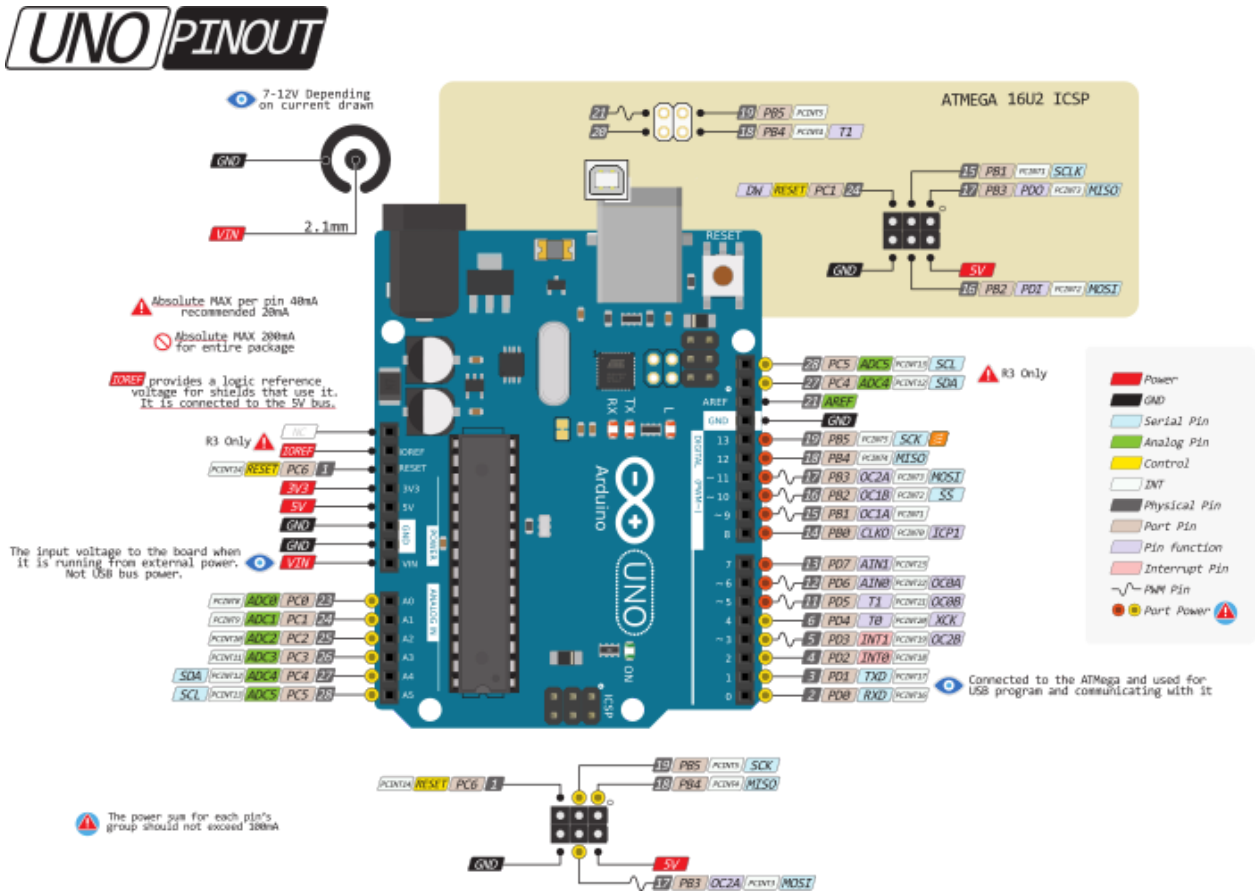


Рисунок 1 - Плата Arduino Uno

Дальше в курсе мы будем разбирать разные особенности ардуино, которые показаны на этой картинке. Сама плата подключается компьютеру по USB.

Чтобы запрограммировать плату на компьютер нужно поставить специальную программу Arduino IDE. Для этого мы переходим по адресу https://www.arduino.cc/en/Main/Software. Далее находим строчку **Windows Installer for Windows XP and up** и нажимаем на нее.

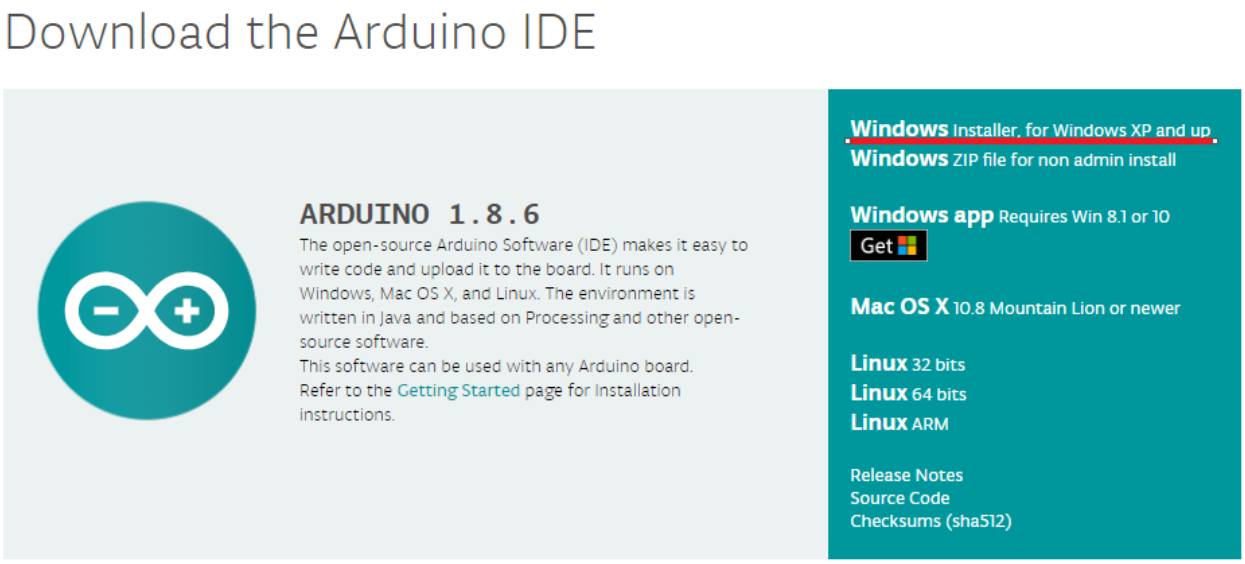


Рисунок 2 - Выбор платформы, для которой скачивается установщик

После нажатия мы переходим на следующую страницу, с которой можно бесплатно скачать саму программу. Для этого нужно нажать на незаметную надпись JUST DOWNLOAD и скачиваем установщик на свой компьютер.

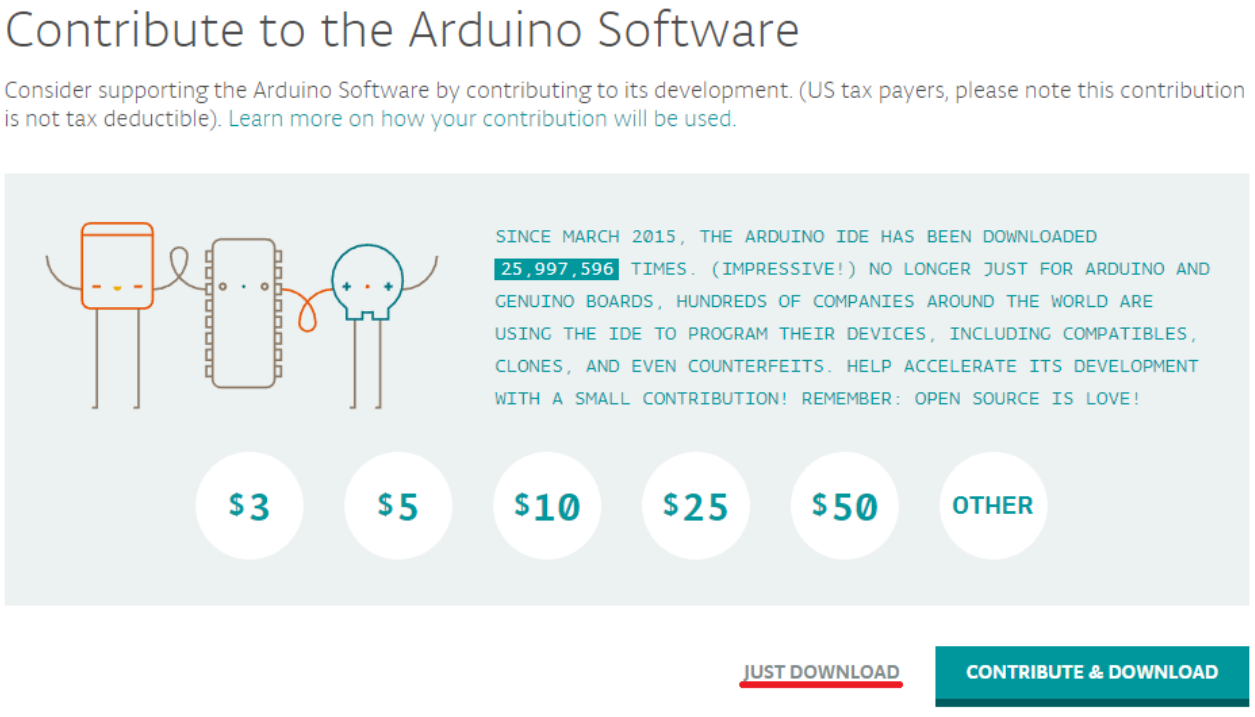


Рисунок 3 - Окно загрузки установщика Arduino IDE

Установка происходит достаточно просто, соглашаемся с лицензионным соглашением, подтверждаем путь установки программы и подтверждаем установку необходимых драйверов.

Давайте запустим программу.

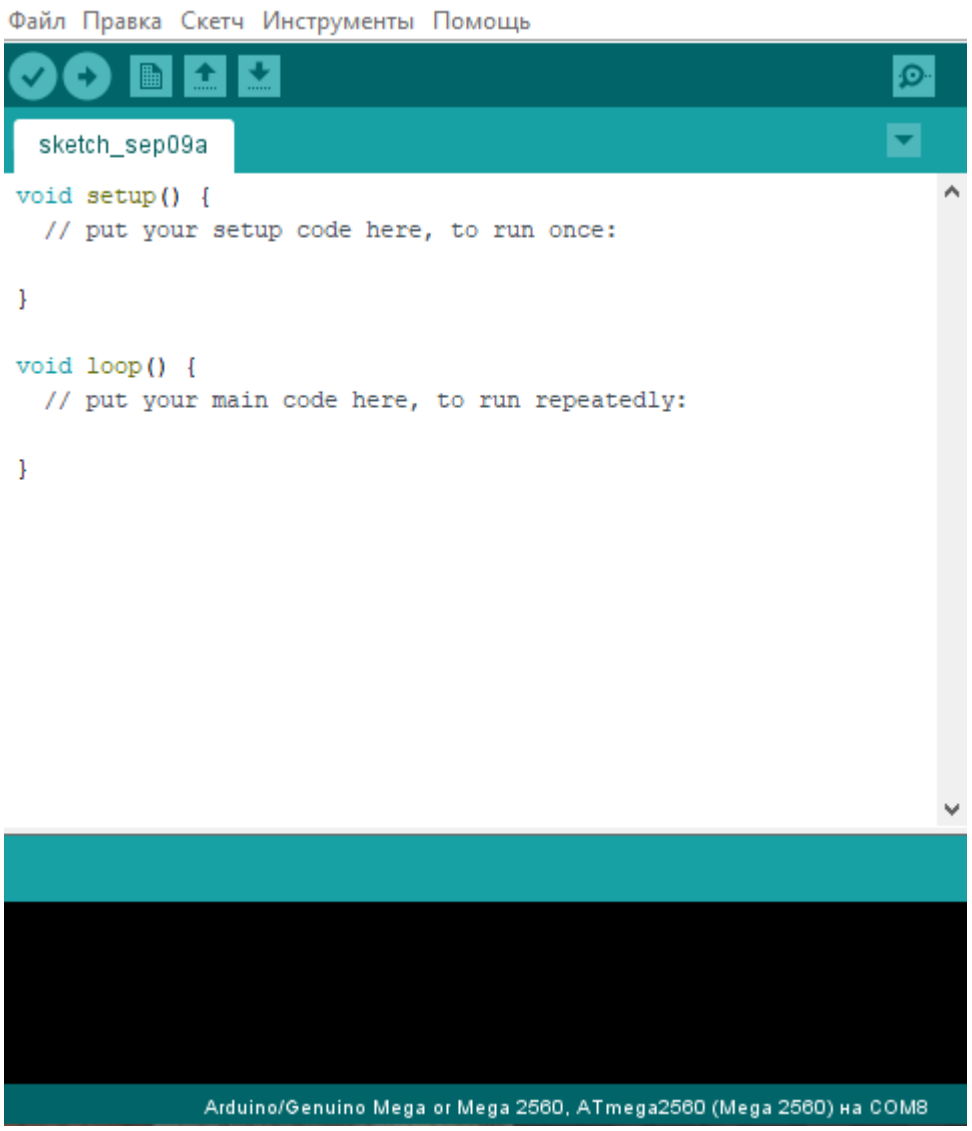


Рисунок 4 - Пустая программа в Arduino IDE

Сама программа, которую мы будем загружать на плату, состоит из двух основных функций:

void setup() – здесь мы пишем настройки для ардуино;

void loop() – здесь мы пишем саму программу.

Обе функции ограничены специальными скобочками {}, в которых мы будем записывать настройки и код программ. В обеих функциях написаны строчки, начинающиеся с двух косых линий (слэш) // **put your setup code here, to run once**. Это называется комментарий. Комментарии программисты пишут для самих себя, чтобы было легче разобраться в коде. При запуске программы наша плата Ардуино их просто не увидит.

После загрузки программы сама плата запустит функцию setup только один раз, а функция loop будет работать постоянно, пока включена плата.

Давайте попробуем написать нашу первую программу. Все, кто когда-то начинал программировать платы, сначала пытались просто помигать одним светодиодом. На плате уже есть светодиод, которым можно управлять, он подключен к 13 пину. Вот сама программа.

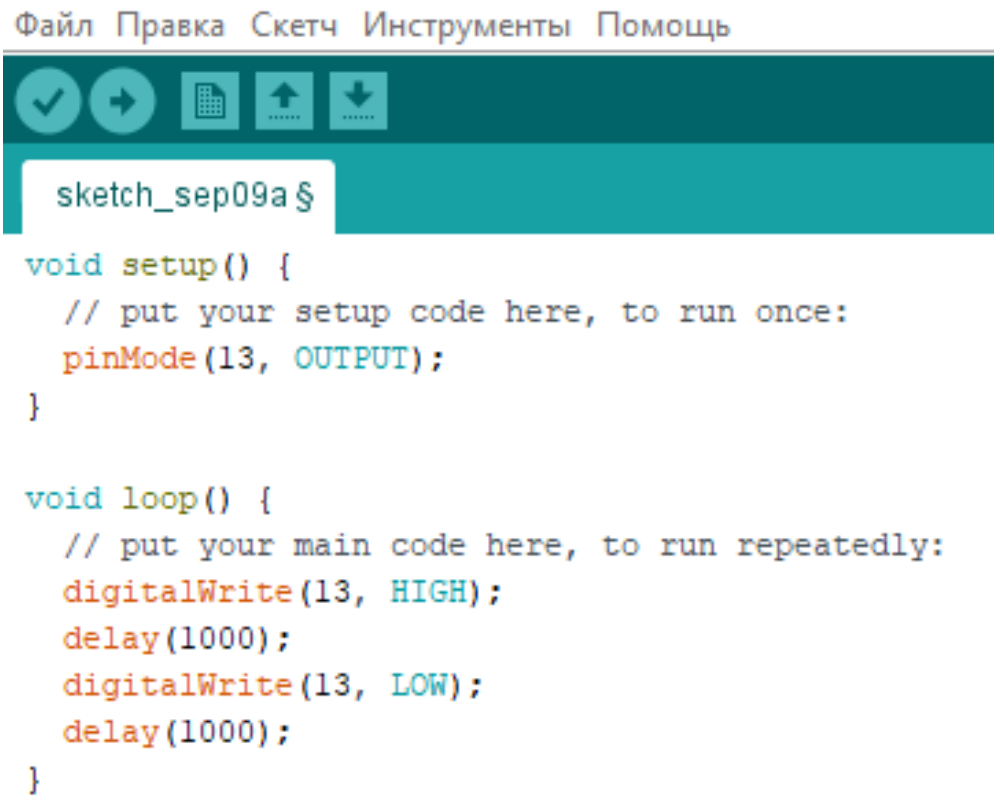


Рисунок 5 - Код программы мигания светодиодом

Давайте разберем все, что написали.

pinMode(13, OUTPUT) – это настройка пина, который мы будем использовать. В скобочках записываем номер пина, а через запятую записываем режим работы. OUTPUT – мы работаем на вывод, подаем питание для устройств, которые требуют электричество. INPUT – работаем на ввод, мы получаем данные с разных датчиков (температура, свет, влажность и т.д.)

digitalWrite(13, HIGH) – в этой строчке мы включаем 13 пин.

delay(1000) – Ардуино сама по себе работает очень быстро, и если бы мы постоянно включали и выключали светодиод, то просто бы не заметили, с какой скоростью это происходит. Чтобы нам было удобней работать, в программу нужно добавить задержку, она измеряется в миллисекундах(1/1000 секунды). В этой строчке мы сказали программе подождать 1 секунду.

digitalWrite(13, LOW) – тут мы выключили 13 пин.

Каждая команда в программе должна заканчиваться точкой-запятой, как в русском языке каждое предложение заканчивается точкой, вопросом или восклицательным знаком.

Помните о том, что программа записанная в функции loop выполняется постоянно? Так вот, после запуска программы мы увидим, как это происходит.

Теперь чтобы загрузить программу на плату, нужно кое-что сделать. Прежде всего, подключите плату к компьютеру. После нужно выбрать порт, к которому подключена Ардуино.

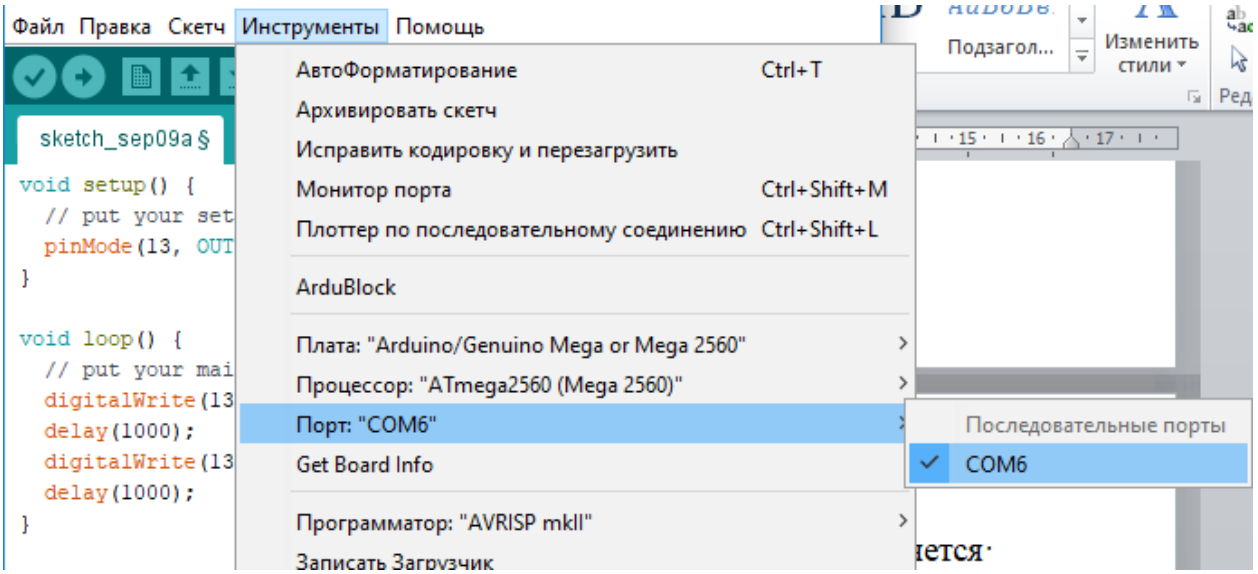


Рисунок 6 - Выбор порта подключения платы Arduino

После нужно проверить, какая версия Ардуино выбрана в программе.

Рисунок 7 - Отображение модели платы, под которую производится сборка

Как видим, в программе выбрана Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560), мы же программируем на плате Arduino Uno

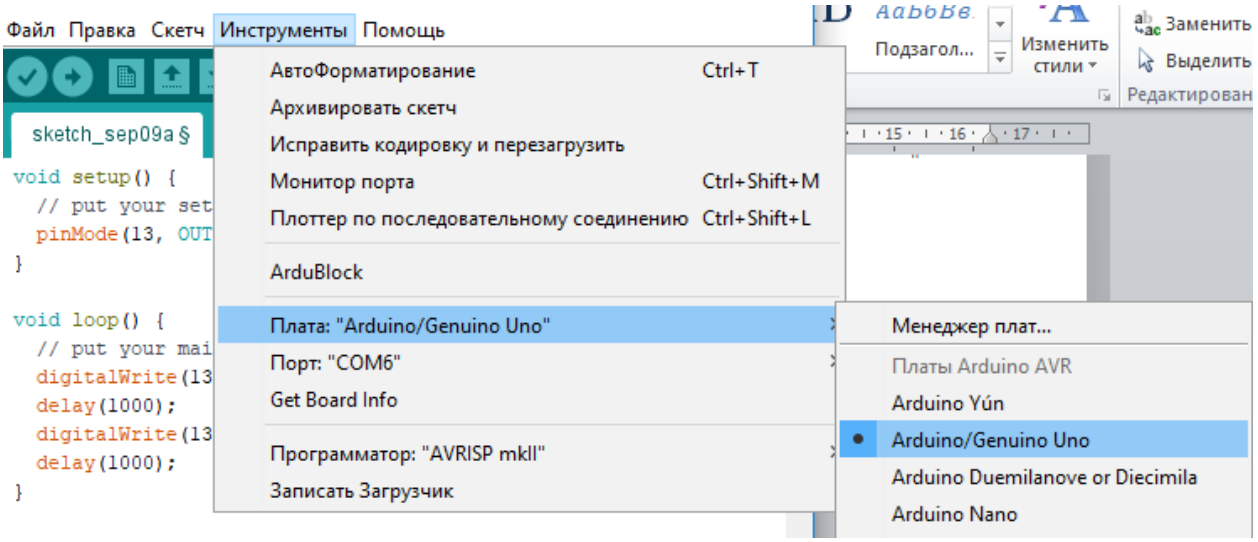


Рисунок 8 - Выбор модели платы, под которую будет производиться сборка

После чего можно попробовать загрузить нашу программу на плату Arduino. Сначала скомпилировать программу. Для этого нажмите на галочку. После компиляции нужно загрузить эту программу на плату, для этого нажмите на стрелку, которая находится немного правее.

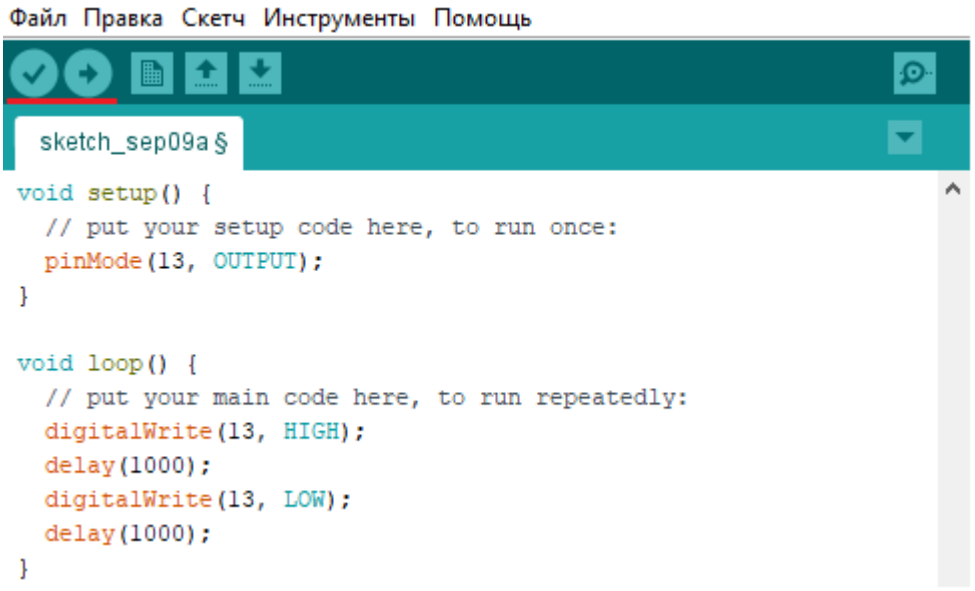


Рисунок 9 - Сборка и загрузка проекта на плату

При написании программы вы могли ошибиться, это называется баг, тогда программа не скомпилируется и не загрузится. Сравните то, что вы написали и то, что в методичке.

После загрузки посмотрите на плату. Мигание светодиодом происходит бесконечно, пока включена плата. Наша программа выполняется постоянно, от начала и до самого конца, а потом заново.

Самостоятельная работа:

Сделайте программу, в которой светодиод будет мигать сначала 1 раз пол секунды, 1 раз 1 секунду и 1 раз 1,5 секунды.

Мигаем несколькими светодиодами

В этом уроке попробуем помигать несколькими светодиодами, изучим циклы и функции. Для начала подключим их, как показано на изображении.

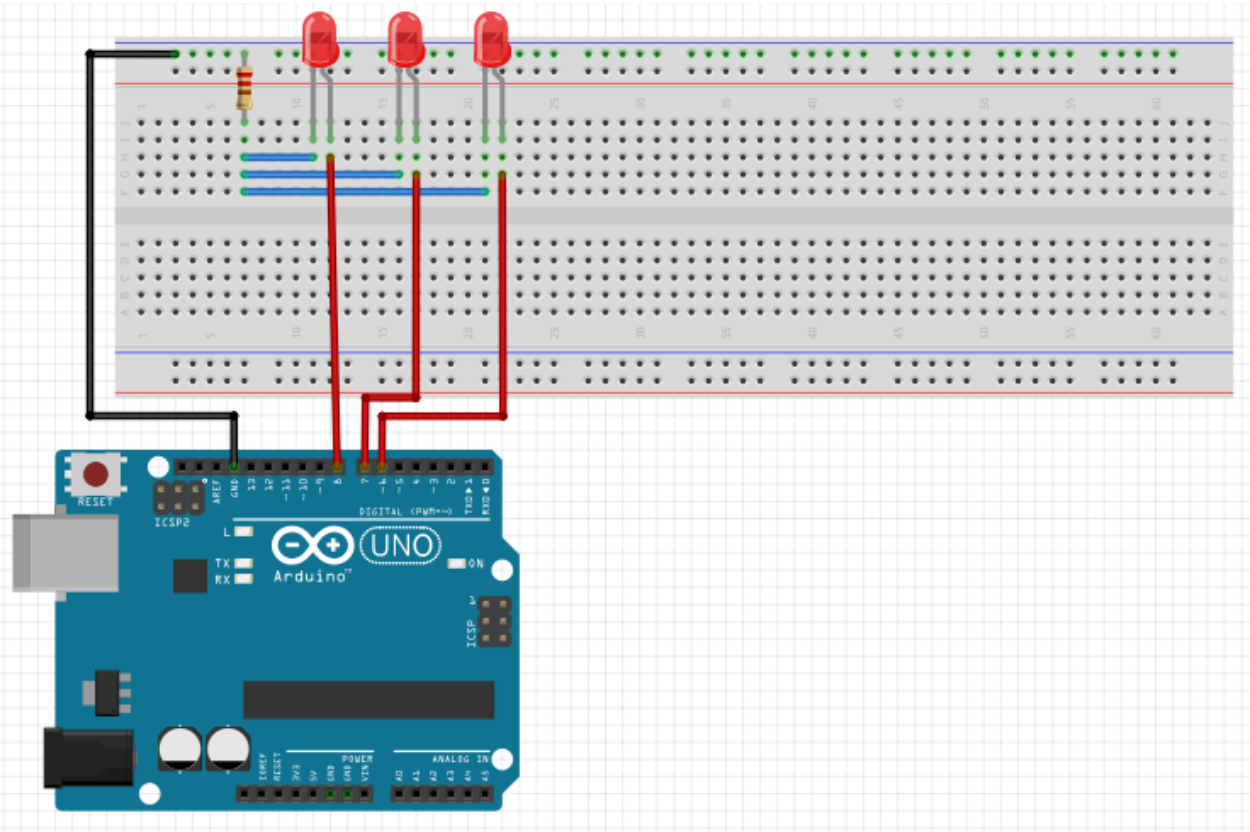


Рисунок 10 - Схема подключения трех светодиодов к плате Arduino Uno

Длинные ножки светодиодов – это «+», а короткие – «-». Подключите плюс светодиодов к 6, 7 и 8 пинам, а минус – к одной ножке резистора, вторую ножку резистора нужно вывести к пину GND.

После подключения попробуйте самостоятельно сделать программу, которая будет мигать разными светодиодами. Если получилось, попробуйте сделать так, чтобы 7 светодиод мигал 3 раза.

Достаточно странно и скучно писать один и тот же код несколько раз, поэтому программисты придумали такую вещь, как цикл. Цикл – это многократное или бесконечное повторение одной и той же или нескольких команд. На самом деле, мы уже встречались с циклом в нашей первой программе – это была функция loop, в ней бесконечно выполнялся код нашей программы.

Давайте теперь сделаем 3 мигания светодиодом при помощи собственного цикла.

```
void setup() {
  // put your setup code here, to run once:
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(6, HIGH);
  delay(1000);
  digitalWrite(6, LOW);
  delay(1000);
  for(int i = 0; i < 3; i++)
  {
    digitalWrite(7, HIGH);
    delay(1000);
    digitalWrite(7, LOW);
    delay(1000);
  }
  digitalWrite(8, HIGH);
  delay(1000);
  digitalWrite(8, LOW);
  delay(1000);
}
```

Рисунок 11 - Код программы для управления тремя светодиодами

Вот так будет выглядеть код нашей программы с использованием циклов.

Тут представлен специальный цикл со счетчиком, цикл for. Для того, чтобы начать работать с этим циклом обязательно нужна специальная переменная, её нужно хранить где-то в памяти, как изображения или музыку на компьютере. Только переменную в программе мы создаем самостоятельно, для этого используется специальное слов **int**. При помощи этого слова объявляется переменная и её задается формат. Переменные типа int могут хранить только целые числа. В них нельзя записать ни буквы, ни слова, ни дробное число. После объявления переменной нужно дать название, по которому мы её найдем. Название обязательно должно быть записано одним словом. В этой программе у переменной очень короткое название i. После названия переменной ей нужно дать какое-то значение. Раз у нас счетчик, то отчет мы начнем с нуля. Полностью объявление переменной будет выглядеть так: int i = 0; Изменяя переменную, нужно обязательно записывать изменения обратно в память, увеличение размера переменной на 2 будет выглядеть примерно так:

i = i + 2;

Увеличение переменной на 1 программисты решили сократить и записывают так:

i++;

Цикл for состоит из нескольких частей, если кратко, то все части имеют свое назначение:

for(начало счетчика; условие завершения; увеличение/уменьшение счетчика)

{

Код цикла

}

Давайте попробуем сократить код программы. Для этого можно использовать свои собственные функции. Функции, как и переменные, программист может объявлять сам. Мы даем названия этим функция и записываем внутри них части программы, чтобы просто вызвать их.

```
void blinkLed(int pin, int del)
{
  digitalWrite(pin, HIGH);
  delay(del);
  digitalWrite(pin, LOW);
  delay(del);
}

void setup() {
  // put your setup code here, to run once:
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  blinkLed(6, 1000);
  for(int i = 0; i < 3; i++)
  {
    blinkLed(7, 1000);
  }
  blinkLed(8, 1000);
}
```

Рисунок 12 - Код программы для управления тремя светодиодами при помощи циклов и функций

Объявляя функцию, мы начинаем со специального слова, как при объявлении переменной, только здесь можно использовать еще один вариант **void**. Далее как и переменной, даем название функции, чтобы дальше её можно было вызвать. После названия в скобочках указываем параметры, которые мы передаем в функцию. Этих параметров может и не быть, тогда скобочки остаются пустыми. В фигурных скобочках мы пишем сам код функции. После этого можно несколько раз вызвать функцию в программе, так можно сильно сократить код.

Обработка нажатия кнопки, борьба с дребезгом

Работа с кнопками – важный этап в программировании микроконтроллеров и платы Ардуино. Также мы рассмотрим понятия дребезги и методы его устранения. Но для начала подключим все как показано на изображении.

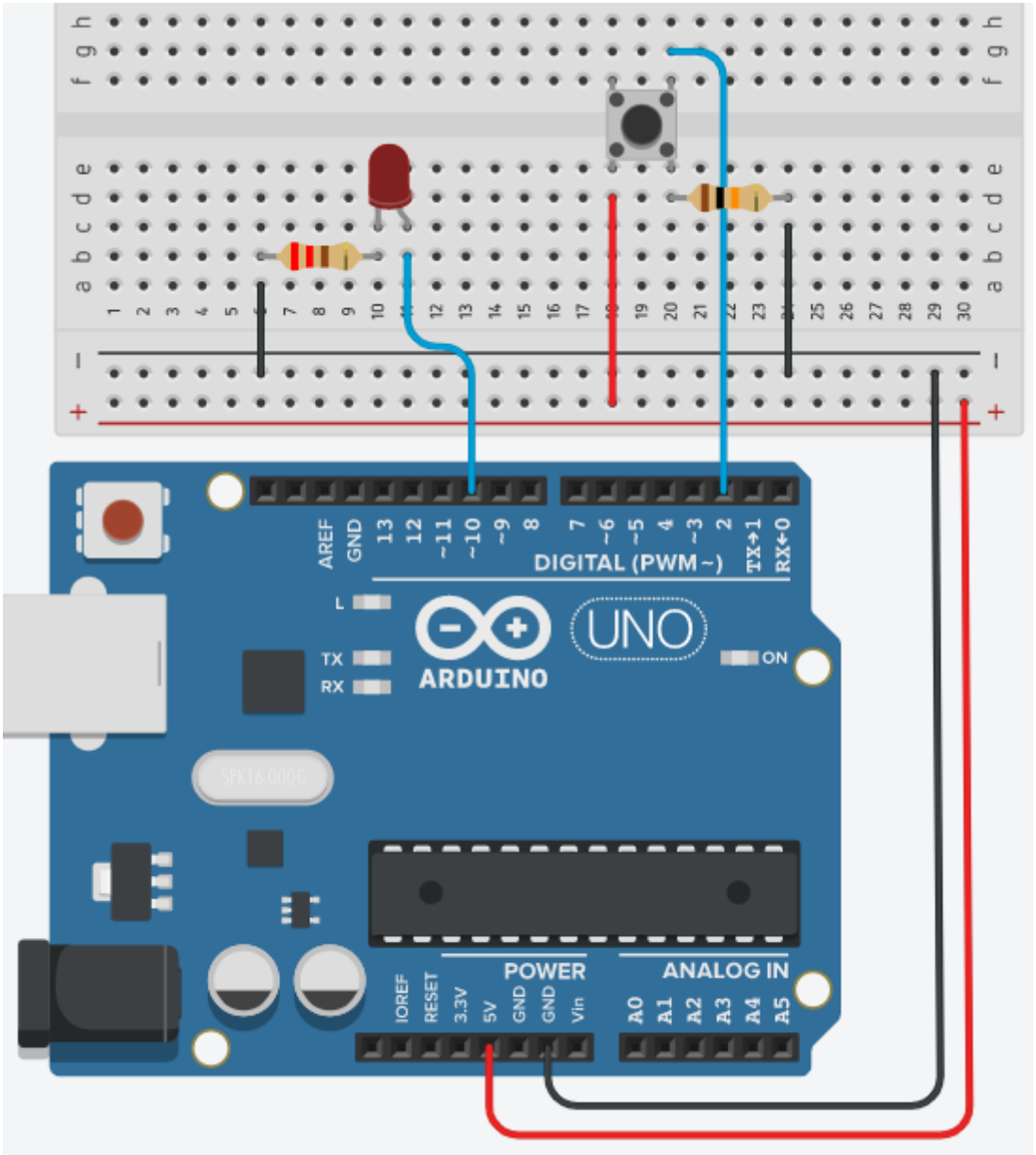


Рисунок 13 - Схема подключения кнопки и светодиода к плате Arduino Uno

В схеме используются два разных резистора: к светодиоду подключаем резистор на 220 Ом, а к кнопке – на 10 кОм.

После подключения начинаем программировать. В этой программе при нажатии на кнопку будет загораться светодиод, а при отпускании – гаснуть.

```
const int LED = 10;   // Контакт 10 для подключения светодиода
const int BUTTON = 2; // Контакт 2 для подключения кнопки

void setup() {
  // Настроим контакт светодиода как выход
  pinMode (LED, OUTPUT);
  // Настроим контакт кнопки как вход
  pinMode (BUTTON, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (digitalRead(BUTTON) == LOW)
  {
    // Включаем светодиод, подавая на вывод 1 (HIGH)
    digitalWrite(LED, LOW);
  }
  else
  {
    // Включаем светодиод, подавая на вывод 0 (LOW)
    digitalWrite(LED, HIGH);
  }
}
```

Рисунок 14 - Код для включения светодиода при нажатии кнопки

В самых первых строчках у нас появились константы. Константа – это именованный участок памяти, который в отличии от переменных нельзя изменять. Объявление константы начинается со специального слова const, так же константе всегда при объявлении должны дать значение, или появится баг и программа не запустится.

В этой программе появилась также возможность в зависимости от условия выполнять то или иное действие.

if(условие)

{

Команды

}

else

{

Команды

}

Как мы видим, тут задается определенное условие, при котором выполняется определенное действие, если же это условие не будет выполняться (второе можно не прописывать), то выполняется другое условие (else).

Такое устройство можно сделать и без использования Ардуино, давайте попробуем переключать светодиод при каждом коротком нажатии на кнопку.

```
const int LED = 10;    // Контакт 10 для подключения светодиода
const int BUTTON = 2; // Контакт 2 для подключения кнопки
int tekButton = LOW;  // Переменная для сохранения состояния кнопки
int prevButton = LOW; // Переменная для сохранения предыдущего состояни кнопки

boolean ledOn = false;// Текущее состояние светодиода

void setup() {
  // Настроим контакт светодиода как выход
  pinMode (LED, OUTPUT);
  // Настроим контакт кнопки как вход
  pinMode (BUTTON, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (tekButton == HIGH && prevButton == LOW)
  {
    // нажатие кнопки - изменить состояние светодиода
    ledOn = !ledOn
    digitalWrite(LED, ledOn);
  }
  prevButton = tekButton
}
```

Рисунок 15 - Код переключения состояния светодиода при нажатии на кнопку

Сначала разберем отдельные новые моменты программы. Появился новый тип данных для функции, **boolean**. Этот тип данных может принимать только только два варианта: **true** или **false**. Это можно использовать при возможности выбора из двух вариантов. Если условие в скобочках *if(условие),* верно (*if(true)*), то только в этом случае будет выполняться часть кода в фигурных скобках.

Немного ниже мы можем увидеть строчку *if (tekButton == HIGH && prevButton == LOW)*. Используя два амперсанда (&&), можно использовать два или больше условий и только когда оба условия верны, код в фигурных скобках будет выполняться.

При нажатии кнопки светодиод должен будет менять свое состояние. Но это происходит не всегда. Виной этому – дребезг кнопок. Кнопка – механический датчик на пружине. Такой механизм может срабатывать несколько раз на протяжении 5 миллисекунд

```
const int LED = 10;   // Контакт 10 для подключения светодиода
const int BUTTON = 2; // Контакт 2 для подключения кнопки
int tekButton = LOW;  // Переменная для сохранения состояния кнопки
int prevButton = LOW; // Переменная для сохранения предыдущего состояни кнопки

boolean ledOn = false;// Текущее состояние светодиода

// Функция сглаживания дребезга. Принимает в качеству
// аргумента предыдущее состояние кнопки и выдает фактическое.
boolean debounce(boolean last)
{
  boolean current = digitalRead(BUTTON);  // Считать состояние кнопки,
  if (last != current)                    // если изменилось...
  {
    delay(5);                             // ждем 5 мс
    current = digitalRead(BUTTON);        // возвращаем состояние кнопки
    return current;
  }
}

void setup() {
  // Настроим контакт светодиода как выход
  pinMode (LED, OUTPUT);
  // Настроим контакт кнопки как вход
  pinMode (BUTTON, INPUT);
}

void loop() {
  tekButton = debounce(prevButton);
  if (tekButton == HIGH && prevButton == LOW)
  {
    ledOn = !ledOn;               // инвертировать значение состояния светодиода
  }
  prevButton = tekButton;
  digitalWrite(LED, ledOn);       // изменить статус состояния светодиода
}
```

Рисунок 16 - Код переключения состояния светодиода с фильтрацией дребезга кнопки

Программирование датчика линии

Датчик линии позволяет различать светлые и темные поверхности, если расстояние до них остается неизменным. Давайте подключим и запрограммируем датчики, как показано на рисунках.

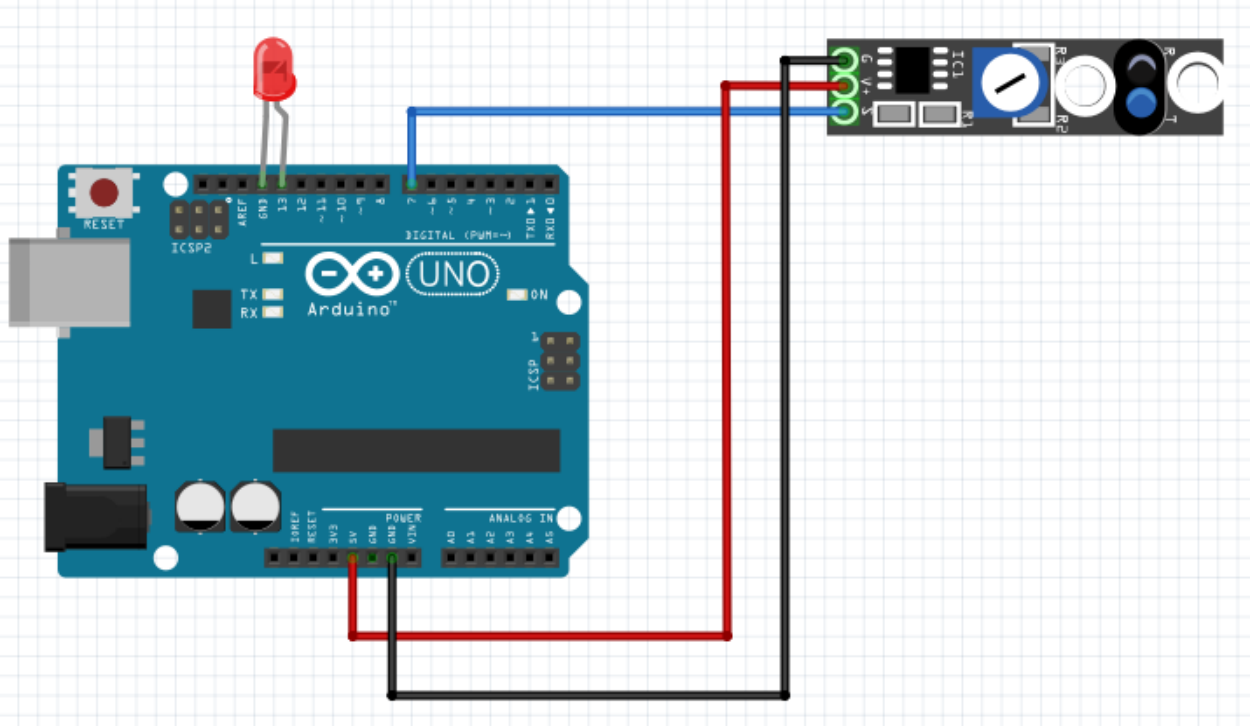


Рисунок 17 - Схема подключения датчика линии

Синий – сигнальный

Красный – 5V

Черный – GND

```
const int buttonPin = 7;
const int ledPin = 13;

int buttonState = 0;

void setup() {
    // put your setup code here, to run once:
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
   buttonState = digitalRead(buttonPin);

   if (buttonState == HIGH) {
      digitalWrite(ledPin, HIGH);
   }
   else {
      digitalWrite(ledPin, LOW);
   }
}
```

Рисунок 18 - Код управления состоянием светодиода при помощи датчика линии

В этой программе, как только датчик засечет светлую поверхность, загорится светодиод, если же он будет на темной поверхности, то светодиод не будет гореть.

Программирование серводвигателя

Серводвигатель – это специальный мотор, который можно поворачивать на определенный угол. Обычно серводвигатель можно повернуть на угол от 0 до 360 градусов (сделать полный круг), но часто попадаются и серводвигатели, которые поворачиваются максимум на 180 градусов (полукруг).

У серводвигателя 3 провода для подключения к ардуино. Каждый провод имеет свой цвет, по которому можно понять куда его подключать:

Черный или коричневый – GND

Красный – 5V

Оранжевый или желтый – Пин.

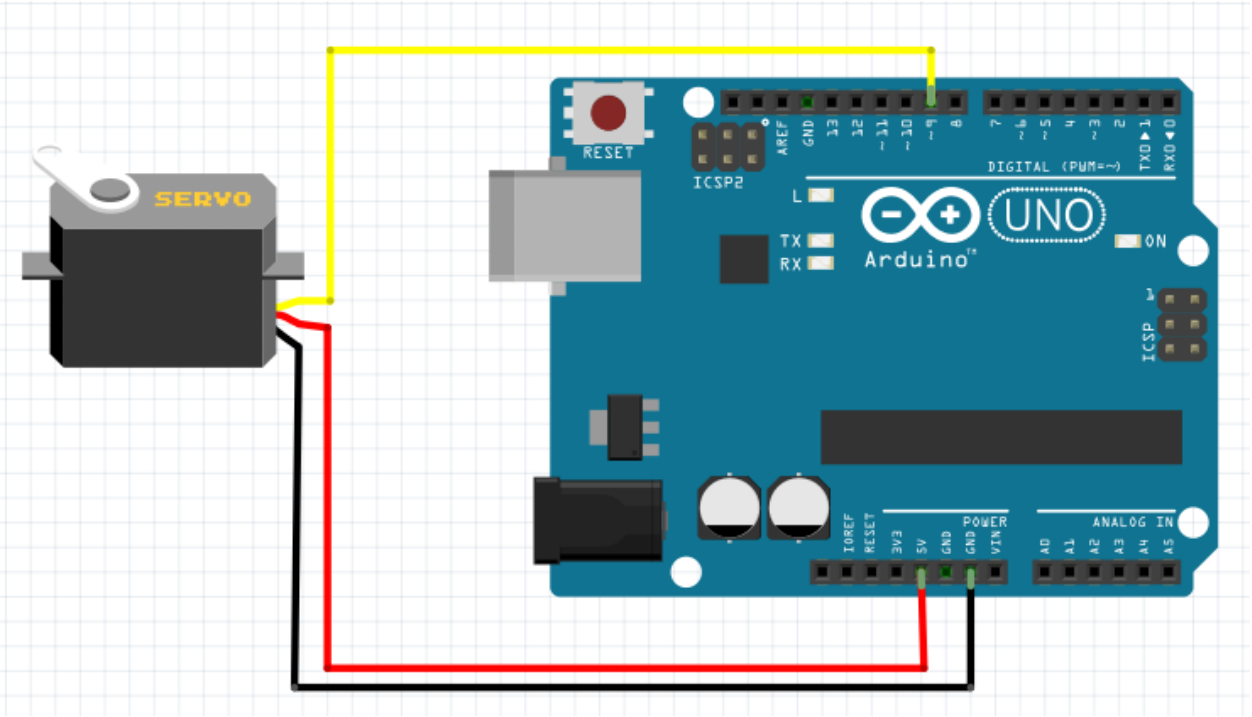


Рисунок 19 - Схема подключения сервопривода к плате Arduino Uno

Желтый – сигнальный

Красный – 5V

Черный – GND

Давайте запрограммируем нашу плату.

```
#include <Servo.h>

Servo myservo;

void setup() {
  // put your setup code here, to run once:
  myservo.attach(9);
}

void loop() {
  // put your main code here, to run repeatedly:
  myservo.write(0);
  delay(1000);
  myservo.write(90);
  delay(1000);
  myservo.write(180);
  delay(1000);
}
```

Рисунок 20 - Код управления сервоприводом

Запрограммировать серводвигатель напрямую достаточно сложно, поэтому вскоре программисты придумали специальные функции и параметры, которые очень упрощают эту задачу. Когда они это закончили, то решили все свою наработки добавить в библиотеку, которую мы с вами подключили.

Для подключения библиотеки есть специальная директива (указание программе) – **#include**, после этой директивы нужно в специальных скобках указать название библиотеки которую мы подключаем, в целом это будет выглядеть так:

#include <Servo.h>

В функции setup мы даем название нашему серводвигателю и указываем, к какому пину его подключили:

myservo.attach(9);

Для изменения угла есть специальная команда, выглядит она примерно так: *название.write(угол);*

В качестве самостоятельного задания напишите программу, в которой будет цикл, постепенно изменяющий угол поворота на 5 градусов больше, пока не дойдет до 180 градусов. Сам счетчик увеличения цикла будет выглядеть так:

for(int i = 0; i <= 180; i = i + 5)

После написания этой программы создайте еще цикл, работающий в обратном порядке, он будет уменьшать угол поворота со 180 до 0. Конечная программа должна работать так, чтобы постепенно увеличивать, а котом уменьшать угол поворота сервопривода.

Программирование дальномера

Дальномер или ультразвуковой датчик измеряет расстояние до объекта.

Перед началом работы с библиотеками нужно их скачать. Для этого перейдите во вкладку Скетч > Подключить библиотеку > Управлять библиотеками.

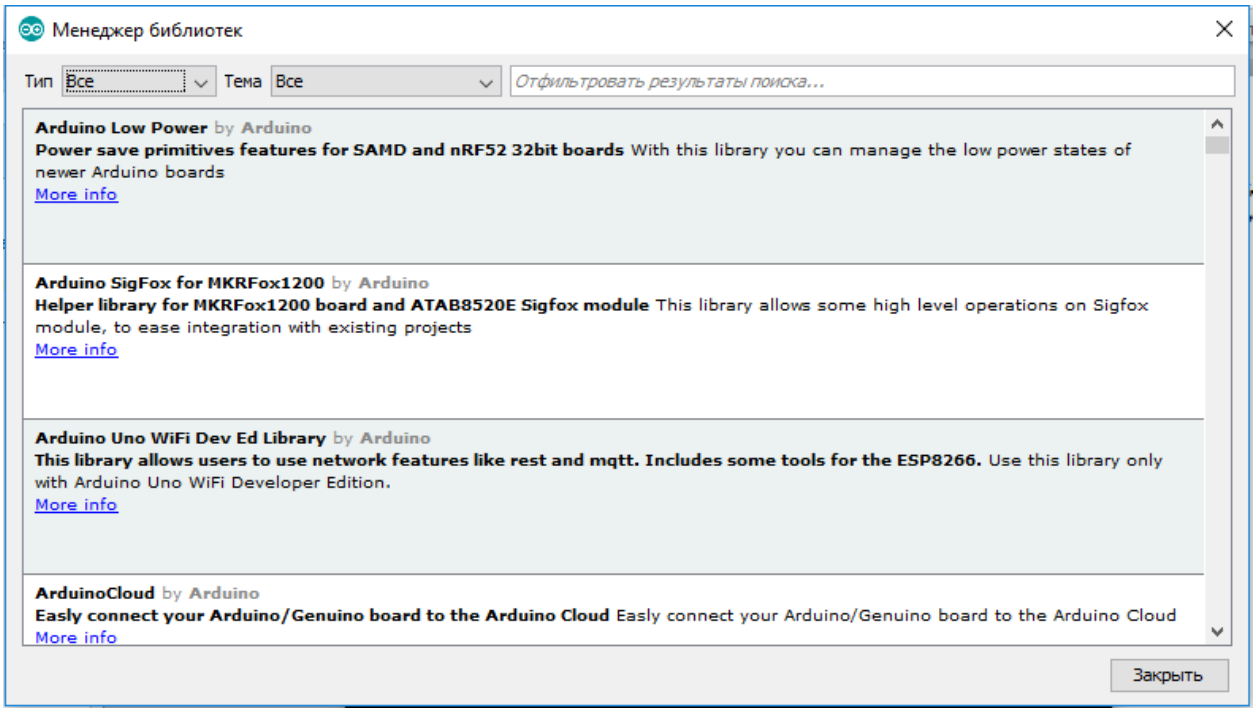


Рисунок 21 - Менеджер библиотек Arduino IDE

Найдите библиотеки **Ultrasonic** и **NewPing.** Проверьте, не установлены ли они? Если нет, то нужно скачать их и установить.

Подключите датчик, как показано на изображении.

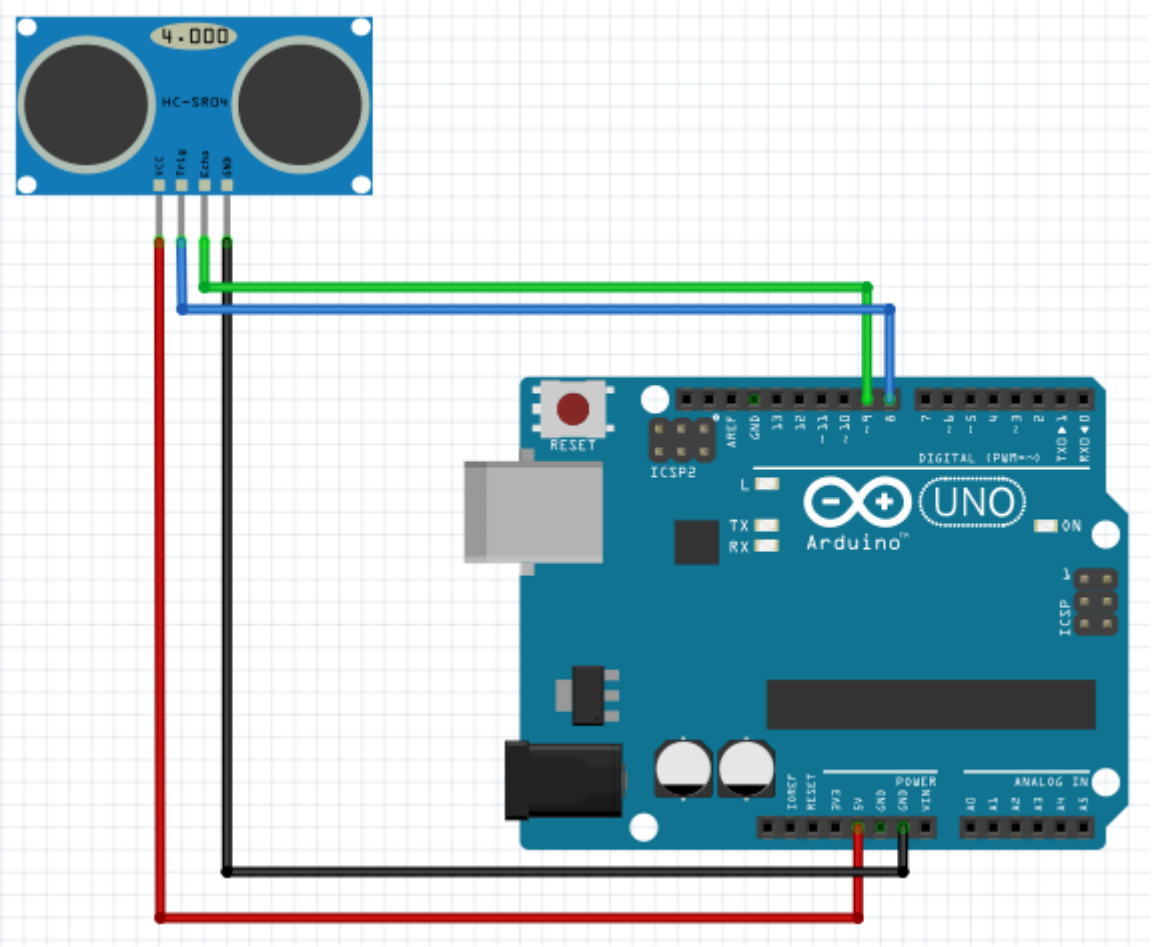


Рисунок 22 - Схема подключения ультразвукового датчика

Красный – 5V

Синий – Trig

Зеленый – Echo

Черный – GND

После этого напишите по очереди 3 программы для управления дальномером. Сравните, с какой точностью работает метод без использования библиотеки и с недавно установленными библиотеками.

После загрузки каждой программы запустите Монитор порта во вкладке Инструменты, чтобы увидеть результат измерения.

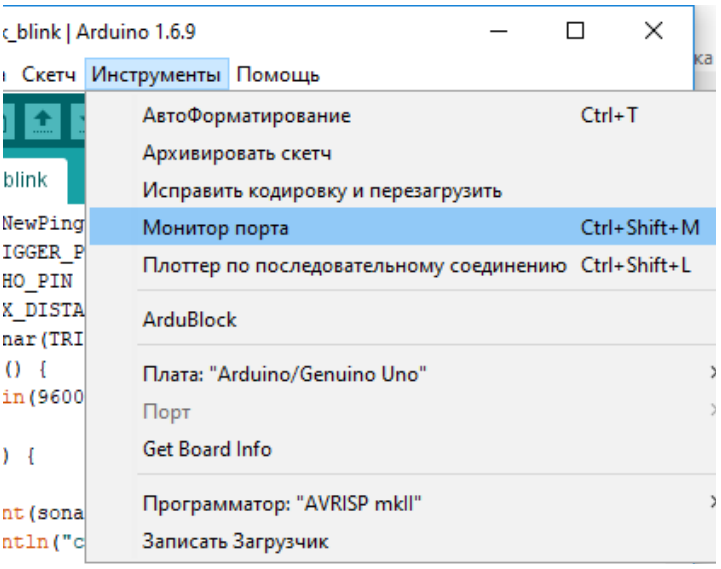


Рисунок 23 - Меню выбора Монитора порта

Программа без использования библиотеки:

```
// Объявляем пины
int echoPin = 9;
int trigPin = 8;
void setup() {
//Настройка монитора порта
Serial.begin (9600);
// Настройка портов подключения
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}
void loop() {
// Объявляем переменные, в которых будем хранить расстояние
int duration, cm;
// Используя порт trigger посылаем короткий ультразвуковой сигнал
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Принимаем сигнал через Echo
duration = pulseIn(echoPin, HIGH);
// Переводим полученные данные в сантиметры
cm = duration / 58;
// Выводим полученный результат на экран
Serial.print(cm);
Serial.println(" cm");
delay(100);
}
```

Рисунок 24 - Код управления ультразвуковым датчиком без использования сторонних библиотек

Программа с использованием библиотеки Ultrasonic

```
#include "Ultrasonic.h"
// Насторойка дальномера
// Trig - 8, Echo - 9
Ultrasonic ultrasonic(8, 9);
void setup()
{
  // Настройка монитора порта
  Serial.begin(9600);
}
void loop()
{
  // Получение расстояния
  float dist_cm = ultrasonic.distanceRead();
  // вывод на монитор порта
  Serial.print(dist_cm);
  Serial.println("cm");
  delay(100);
}
```

Рисунок 25 - Код управления ультразвуковым датчиком при помощи библиотеки Ultrasonic

Программа с использованием библиотеки NewPing

```
#include <NewPing.h>
#define TRIGGER_PIN 8
#define ECHO_PIN 9
#define MAX_DISTANCE 200
// Настройка пинов и максимального расстояния.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
void setup() {
  // Настройка монитора порта
Serial.begin(9600);
}
void loop() {
delay(50);
// Вывод сигнала на монитор опрта в санитметрах
Serial.print(sonar.ping_cm());
Serial.println("cm");
}
```

Рисунок 26 - Код управления ультразвуковым датчиком при помощи библиотеки NewPing

Какая программа вам больше понравилась и почему?

_____
_____
_____

Прочитайте программу без использования библиотеки, как можно подробнее опишите, как работает программу. В лучшем случае необходимо описать, какой пин отправляет сигнал, какой получает и как устройство понимает, что звук прошел это расстояние.

_____
_____
_____
_____
_____
_____
_____

Работа с инфракрасным датчиком

Инфракрасный датчик используется для получения данных с пульта. Сам пульт при нажатии клавиш отправляет луч света, который незаметен для человеческого глаза, датчик считывает эти данные и переводит из в числовые значения(кодирует), чтобы было проще работать.

Перед началом работы с библиотеками нужно их скачать. Для этого перейдите во вкладку Скетч > Подключить библиотеку > Управлять библиотеками.

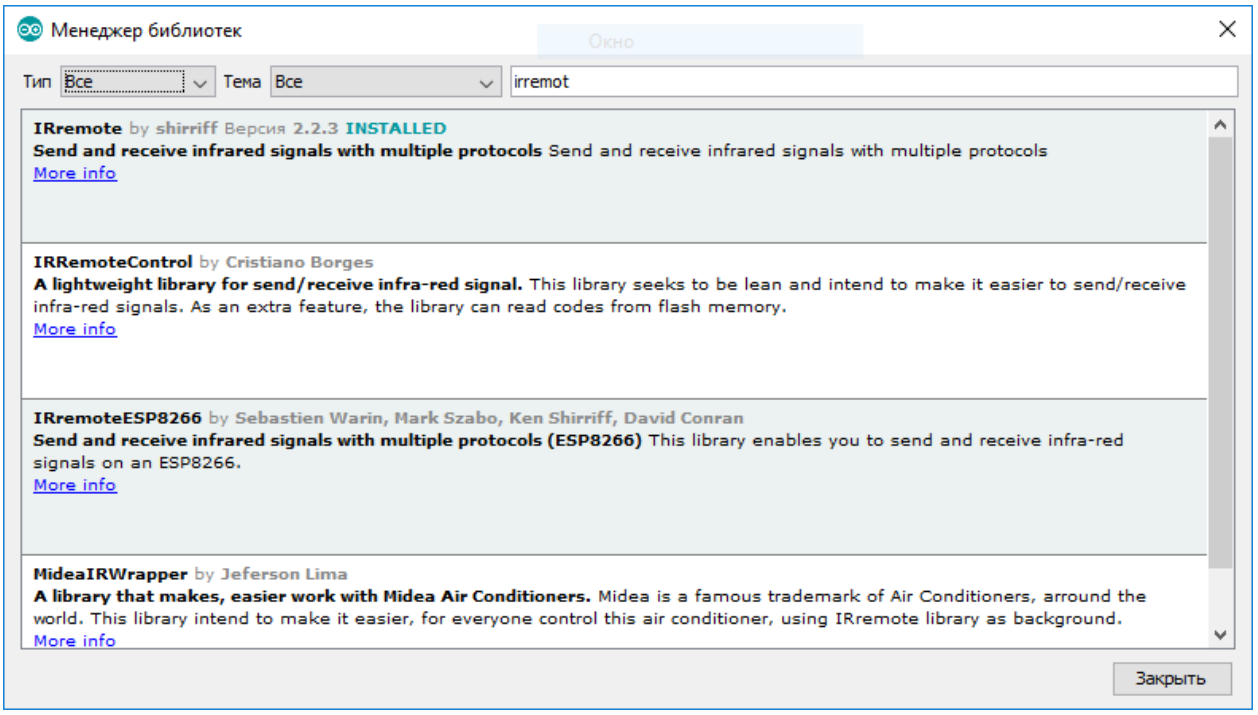
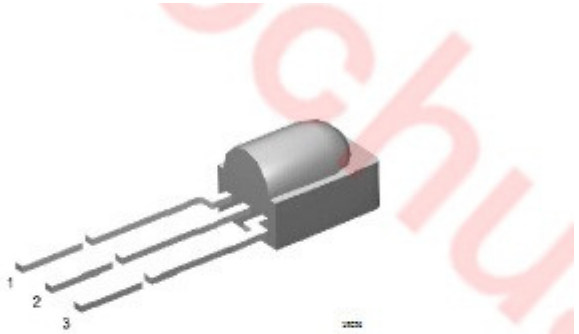


Рисунок 27 - Установка библиотеки для управления инфракрасным датчиком при помощи Менеджера библиотек

Найдите библиотеку **IRremote.** Проверьте, не установлена ли она? Если нет, то нужно скачать её и установить.

После этого подключите ик-приемник как показано на картинке.



MECHANICAL DATA

Pinning:

1 = OUT, 2 = GND, 3 = V

Рисунок 28 - Распиновка ножек инфракрасного датчика

1 ножка на пин(11), 2 – на GND, 3 – 5V

Программа:

```
#include <IRremote.h>

IRrecv recv(11);

decode_results results;

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

void loop() {
  if (recv.decode(&results)) {
    Serial.println(results.value);
    recv.resume();
  }
  delay(100);
}
```

Рисунок 29 - Код для считывания сигналов с инфракрасного датчика

Запишите в эту таблицу клавиши пульта, и их код для клавиш, которыми будете управлять машинкой укажите, что будете делать(например клавиша 2 имеет код XXXXXXXX и будет использовать для движения вперед)

Измените программу на представленную дальше.

Замените XXXXXXXX на код одной из клавиш.

Попробуйте запустить программу и зажечь светодиод.

```
#include <IRremote.h>
IRrecv recv(11);
decode_results results;

void setup()
{
  Serial.begin(9600);
  recv.enableIRIn();
  pinMode(13, OUTPUT);
}

void loop() {
  if (recv.decode(&results)) {
    Serial.println(results.value);
    if (results.value == XXXXXXXX)// Вместо XXXXXXXX запишите код одной из клавиш
    {
      digitalWrite(13, HIGH);
    }
    else
    {
      digitalWrite(13, LOW);
    }
    recv.resume();
  }
  delay(100);
}
```

Рисунок 30 - Код для управления светодиодом при помощи инфракрасного датчика

Учебное издание

Составители:

Ростислав Олегович **Фокин**

ОСНОВЫ РАБОТЫ С ARDUINO

Методические указания к лабораторной работе

Темплан 2020 г. (учебно-методическая литература). Поз. №

Подписано в печать 00.00.2020. Формат 60x84 1/16. Бумага офсетная.

Гарнитура Times. Печать офсетная. Усл. печ. л. 2,0.

Тираж 10 экз. Заказ

Волгоградский государственный технический университет.

400005, г. Волгоград, просп. В. И. Ленина, 28, корп. 1.

Отпечатано в типографии ИУНЛ ВолгГТУ.

400005, г. Волгоград, просп. В. И. Ленина, 28, корп. 7.