МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
"МИРЭА – РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"
РТУ МИРЭА

Детский Технопарк «Альтаир»

Техническое задание мастер класса по теме «Светофор»

Москва 2025

СОДЕРЖАНИЕ

1 НЕОБХОДИМЫЕ МАТЕРИАЛЫ

Для проведения данного мастер-класса можно использовать разные составляющие, однако при реализации данного варианта мастер-класса использовались:

1. Соединительные провода «папа-папа» (~4 штук)
2. Резистор (3 штуки)
3. Светодиоды (3 штуки: красный, зеленый, желтый)
4. Arduino UNO (или любая другая подобная плата с минимум 3-мя цифровыми выходами)
5. Провод для подключения Arduino к компьютеру
6. Среда написания кода и прошивки платы - *Arduino IDE*

2 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

2.1 Текстовое введение

Светофоры являются важной частью современной транспортной сети, обеспечивая безопасное движение автомобилей и пешеходов. В данном мастер-классе мы рассмотрим простой принципы работы светофора и соберем его простую модель на базе микроконтроллера и трех светодиодов.

Светофор работает по циклическому алгоритму, переключая сигналы в определенной последовательности: красный – остановка, желтый – предупреждение о смене сигнала, зеленый – разрешение движения. В реальных светофорах дополнительно могут использоваться таймеры, датчики движения и адаптивные алгоритмы управления. В нашем проекте мы создадим базовую модель, реализующую стандартный алгоритм смены сигналов.

Программа будет выполнять следующие шаги: включение красного светодиода на 10 секунд и мигание в течение 3 секунд, затем включение желтого на короткое время – около 5 секунд, затем включение зеленого аналогично красному: 10 секунд горит 3 секунды мигает, после чего цикл повторяется. Этот алгоритм будет написан на языке программирования и загружен в микроконтроллер.

2.2 Схема подключения

На данном примере (рисунок 1) использовалась плата Arduino UNO, но ничего не запрещает заменить эту модель на другую - аналогичную. Пример со сборкой на базе Arduino Nano на рисунке 2.

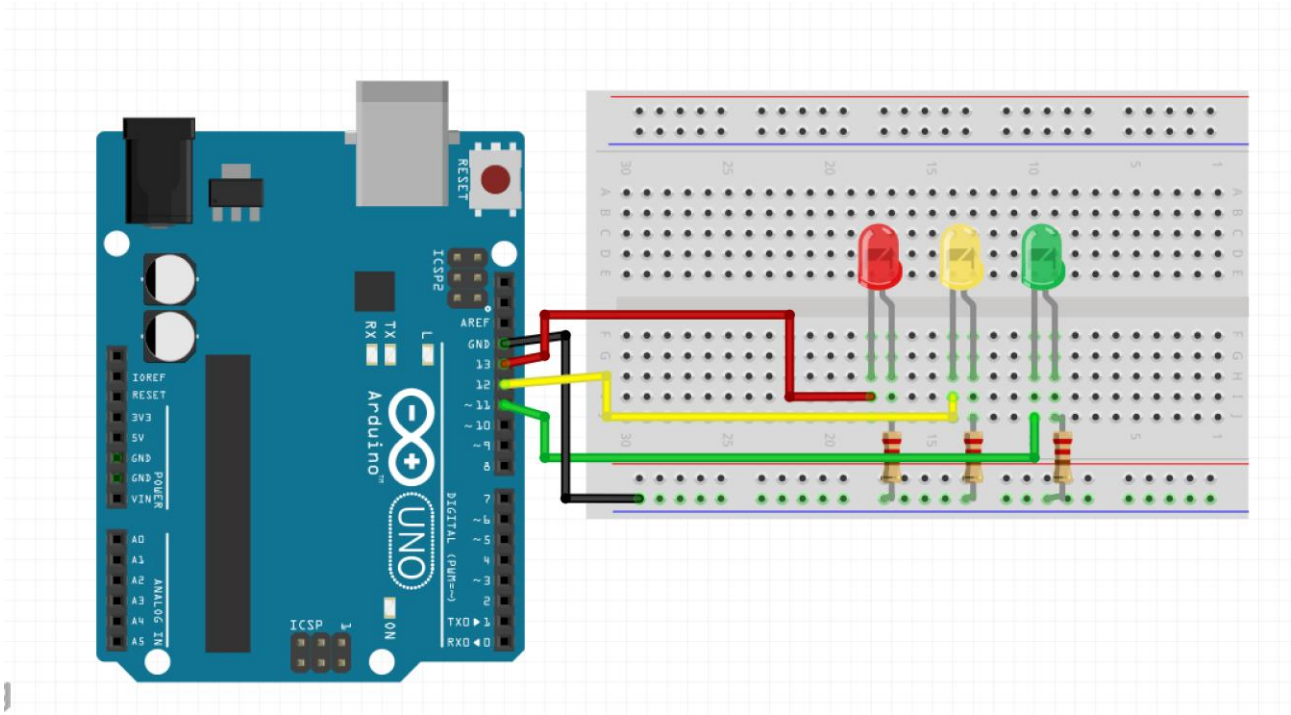

Рисунок 1 - пример сборки схемы на Arduino UNO

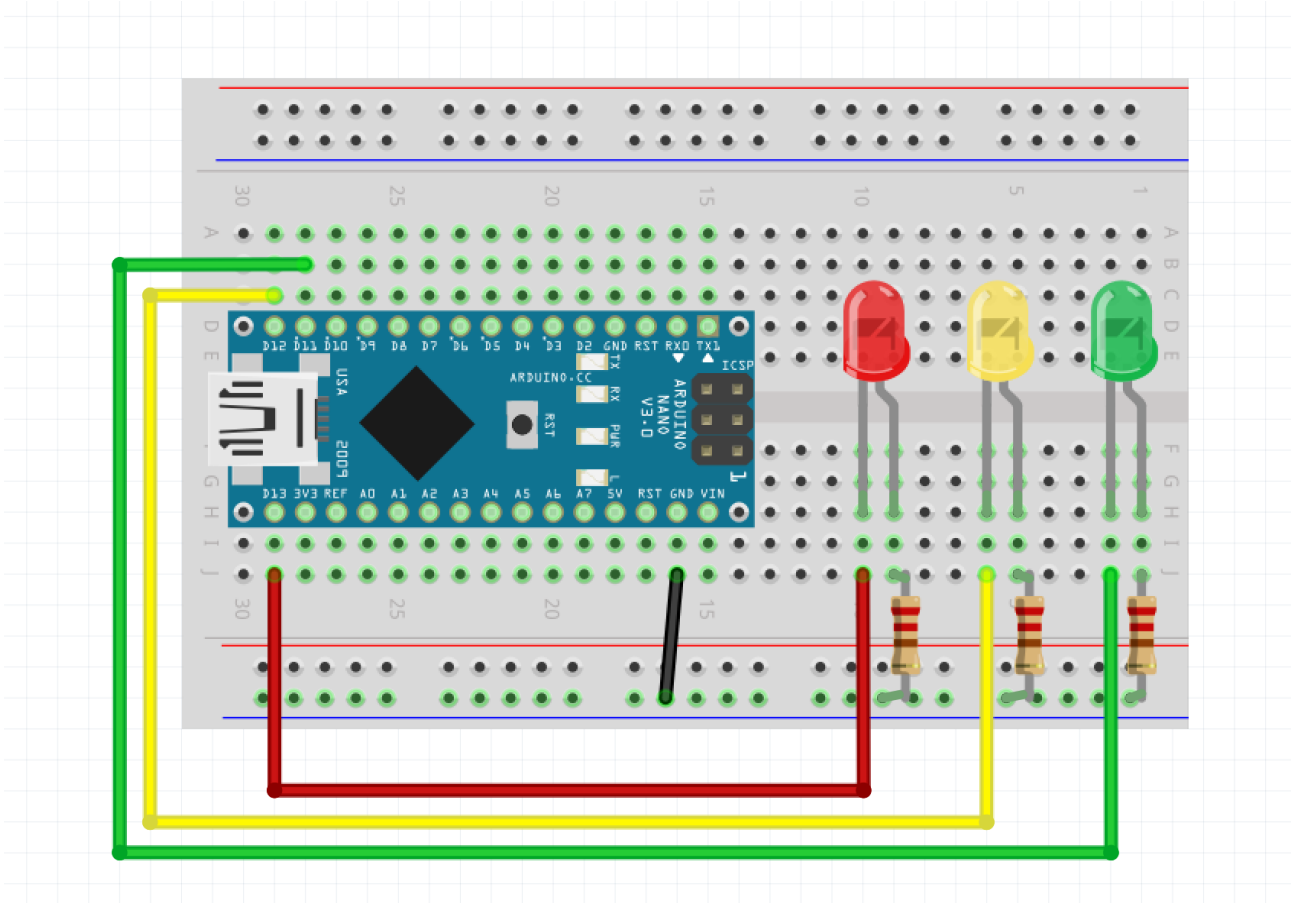

Рисунок 2 - пример сборки схемы на Arduino Nano

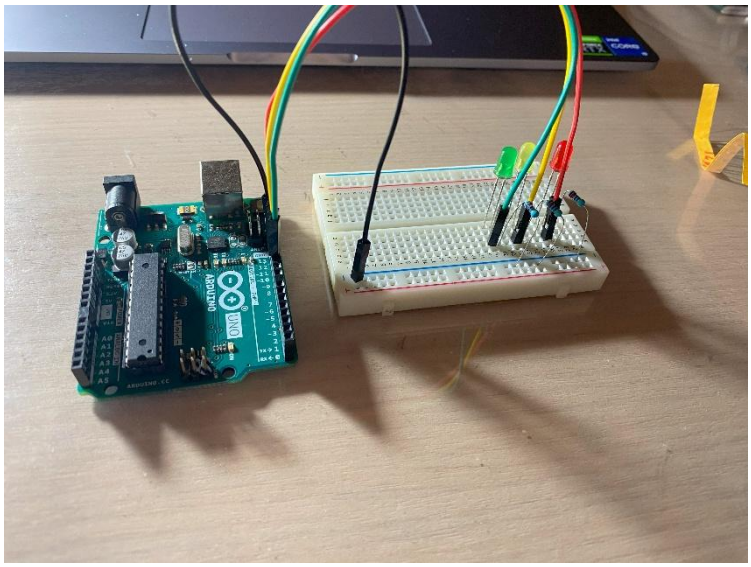

Рисунок 3.1 - пример сборки в жизни (Arduino UNO)

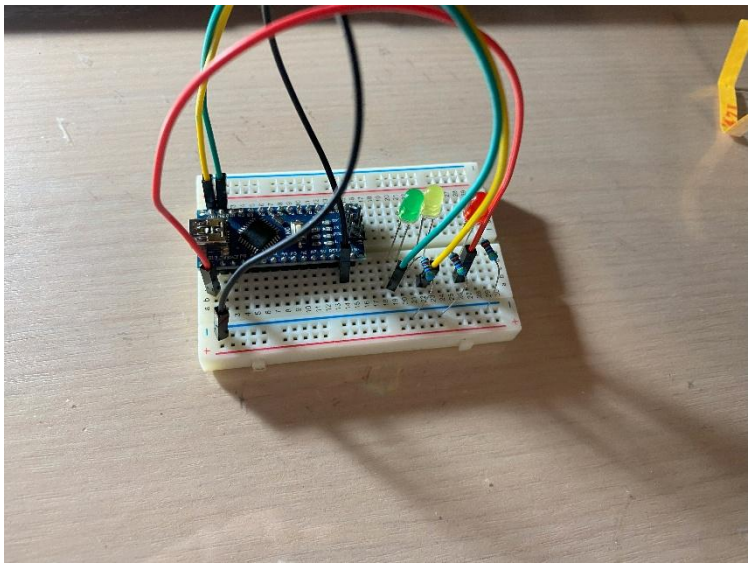

Рисунок 3.2 - пример сборки в жизни (Arduino Nano)

2.3 Исходный код

```
// Определяем пины для светодиодов
int RED = 13; // красный
int YELLOW = 12; // желтый
int GREEN = 11; // зеленый
void setup() { // Сработает только при включении
  // Режим работы пинов выставляем на ВЫХОД (ВЫВОД)
  pinMode(RED, OUTPUT);
  pinMode(YELLOW, OUTPUT);
  pinMode(GREEN, OUTPUT);
}
void loop() { // Повторяется всегда
  // включаем краный светодиод
  digitalWrite(RED, HIGH);
  // ждем 10 секунд (10000 миллисекунд)
  delay(10000);
  // Красный мигает 3 секунды (2-3 раза в секунду)
  for (int i = 0; i < 6; i++) {
    digitalWrite(RED, LOW);
    delay(250);
    digitalWrite(RED, HIGH);
    delay(250);
  }
  digitalWrite(RED, LOW);
  digitalWrite(YELLOW, HIGH);
```

```
  delay(5000);
  digitalWrite(YELLOW, LOW);
  digitalWrite(GREEN, HIGH);
  delay(10000);
  for (int i = 0; i < 6; i++) {
    digitalWrite(GREEN, LOW);
    delay(250);
    digitalWrite(GREEN, HIGH);
    delay(250);
  }
  digitalWrite(GREEN, LOW);
}
```

Это простая версия кода, где линейно повторяются несколько совершенно одинаковых операций выглядит так, как показано выше. Такой подход считается самым простым, но при этом количество полезных строк кода не самое оптимальное. Теперь хочется как-то сократить код так, чтобы было не так громоздко в плане количества строк кода. Для этого предлагается использовать собственные функции. К примеру можно отдельно вынести операцию мигания светодиода, выделив ее в отдельную функцию. Назовем ее *blink:*

```
// функция мигания светодиодом, подключенного к пину pin
void blink(int pin, int time=250){
  for (int i = 0; i < 6; i++) {
    digitalWrite(pin, LOW);
    delay(time);
    digitalWrite(pin, HIGH);
    delay(time);
  }
}
```

Уже с помощью этой функцией можно заменить все внутренние циклы в нашей программе.

2.4 3D модели

Рисунок 4 - 3Д модель светофора, напечатанная на 3Д принтере

3D модель была взята из открытых источников, а конкретнее с сайта *thingiverse.com;* для проведения мастер-класса наличие подобной конструкции необязательно, но развивает интерес и показывает практичность и полезность работы обучающегося. Пример того, как может выглядеть модель светофора, напечатанная на 3Д принтере на рисунке 4.

Ссылка на модель: https://www.thingiverse.com/thing:3988668.

2.5 Расширенная версия

Если задача одностороннего светофора обучающимся далась легко, то можно попробовать ее усложнить, добавив дополнительных условий. К примеру теперь поток машин у нас с двух сторон, следовательно необходимо продублировать информацию от светофора машинам с одной его стороны и с другой. Или же, сделать светофор на перекрестке, где уже необходимо

реализовать обратную логику – если красный свет горит на одной полосе, то боковая стоит, а если наоборот, то боковая едет, а основная (она же прямая) стоит и ждет всех.

Для решения достаточно впихнуть несколько условий в основной *loop* цикл программы. Расширенный код может выглядеть вот так:

```
// Определяем пины для светодиодов
int RED1 = 13; // красный (первая сторона)
int YELLOW1 = 12; // желтый (первая сторона)
int GREEN1 = 11; // зеленый (первая сторона)

int RED2 = 10; // красный (вторая сторона)
int YELLOW2 = 9; // желтый (вторая сторона)
int GREEN2 = 8; // зеленый (вторая сторона)

void setup() { // Сработает только при включении
  // Режим работы пинов выставляем на ВЫХОД (ВЫВОД)
  pinMode(RED1, OUTPUT);
  pinMode(YELLOW1, OUTPUT);
  pinMode(GREEN1, OUTPUT);
  pinMode(RED2, OUTPUT);
  pinMode(YELLOW2, OUTPUT);
  pinMode(GREEN2, OUTPUT);
}

void loop() { // Повторяется всегда
  // включаем красный светодиод на первой стороне, зеленый на
второй
  digitalWrite(RED1, HIGH);
  digitalWrite(GREEN2, HIGH);
  delay(10000);

  // Красный на первой стороне и зеленый на второй мигают 3
секунды
  blink(RED1, 250);
  blink(GREEN2, 250);
  digitalWrite(RED1, LOW);
  digitalWrite(GREEN2, LOW);

  // Желтый загорается на обеих сторонах
  digitalWrite(YELLOW1, HIGH);
  digitalWrite(YELLOW2, HIGH);
  delay(5000);
```

```
  digitalWrite(YELLOW1, LOW);
  digitalWrite(YELLOW2, LOW);

  // Зеленый на первой стороне, красный на второй
  digitalWrite(GREEN1, HIGH);
  digitalWrite(RED2, HIGH);
  delay(10000);

  // Зеленый на первой стороне и красный на второй мигают 3
секунды
  blink(GREEN1, 250);
  blink(RED2, 250);
  digitalWrite(GREEN1, LOW);
  digitalWrite(RED2, LOW);
}

// функция мигания светодиодом, подключенного к пину pin
void blink(int pin, int time=250){
  for (int i = 0; i < 6; i++) {
    digitalWrite(pin, LOW);
    delay(time);
    digitalWrite(pin, HIGH);
    delay(time);
  }
}
```

В этом примере код работает для светофора, который стоит по середине дороги, как показано на рисунке 5.

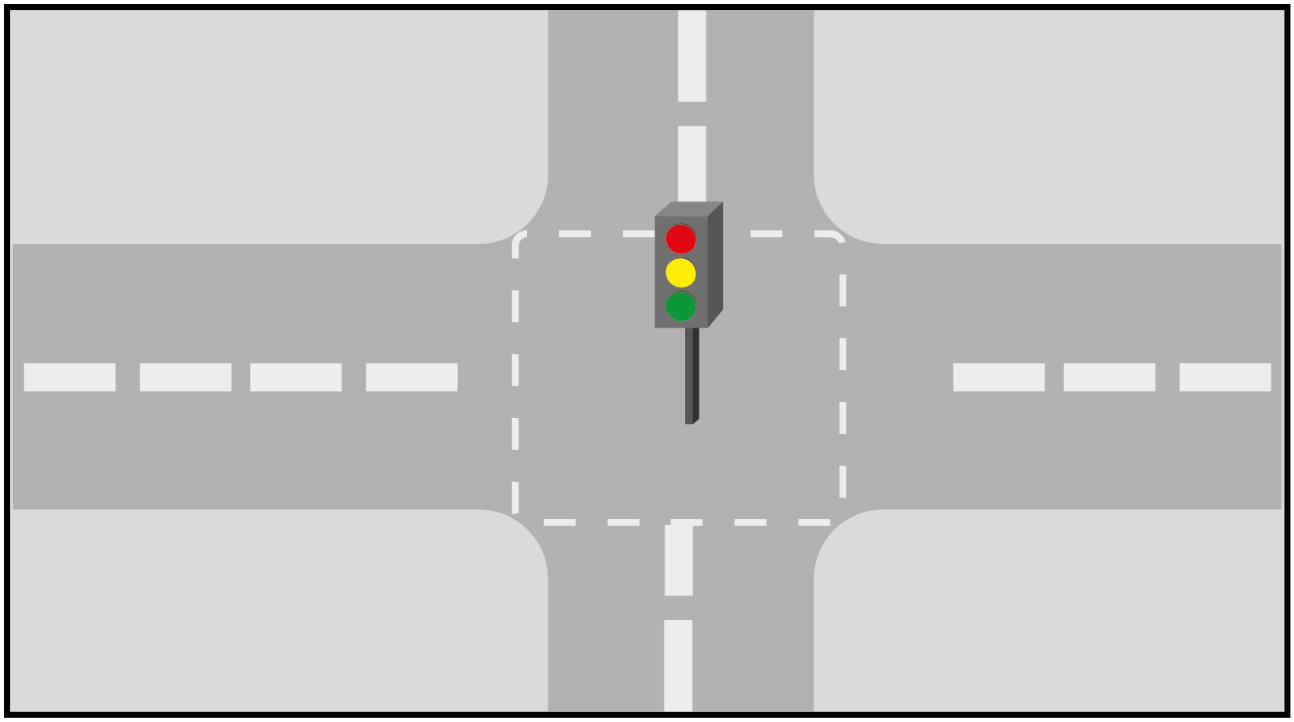

Рисунок 5 - пример светофора на перекрестке

3 СБОРКА И ПРОВЕДЕНИЕ МАСТЕР-КЛАССА

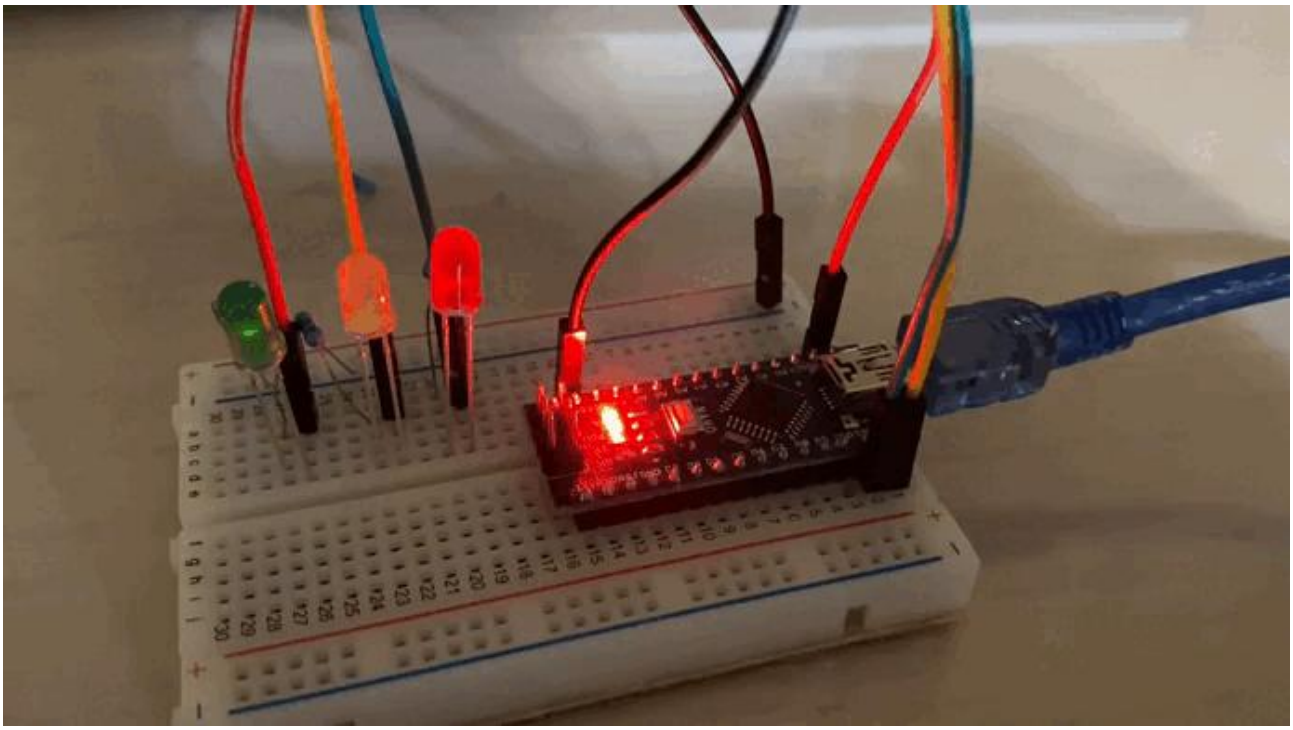

Рисунок 6 – GIF презентация работы схемы

3.1 Основная часть

Сборку схемы следует начать с поиска необходимых компонентов. На рисунке 6 использовалось всего 3 светодиода, 3 резистора и плата с микроконтроллером Arduino Nano. В данном примере легко можно сократить количество используемых резисторов до 1, подключив его к общему выводу земли GND, а светодиоды подключить напрямую в общую линию GND и подключить ко второй ноге каждого отдельно провод питания. Наглядная схемы присутствует на рисунке 7.

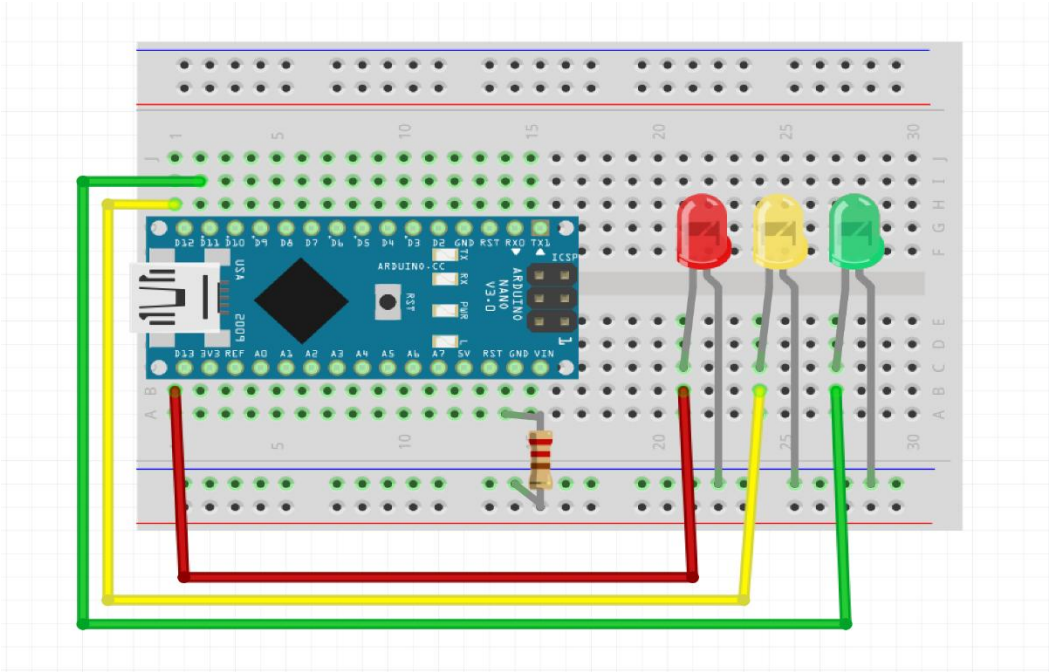

Рисунок 7 - пример сокращенной схемы в плане компонентов

Стоит сказать, что не имеет значение к какой «ноге» светодиода подключать резистор, главное то, что по итогу мы уменьшаем силу тока, проходящую через электрическую цепь. С другой стороны, важно соблюдать полярность светодиода, т.е. подключать анод к плюсу питания, а катод к минусу.

При программировании необходимо учесть, что у разных сбоок могут использоваться разные пины!

Что же означает каждая строчка кода?

Для начала объявляем переменные. *Переменная простыми словами — это хранилище данных. Сюда можно положить какое-то значение (например, число, строку или другой тип данных).* В переменных мы храним номер пинов, к котором подключены светодиоды. Это не обязательно делать, но сильно упрощает дальнейшее написание и поддержку кода, что важно для масштабных проектов. *Лучше научиться правильно делать сейчас, чем переучиваться потом.*

В базовом представлении Arduino кода присутствуют 2 базовые функции, которые отвечают за основную работу микроконтроллера: *void setup* и *void loop*. Функция **setup** выполниться один раз, при запуске платы, а функция **loop** будет выполняться до тез пор, пока есть питание.

В setup мы прописываем логику работы пинов: вход/выход или считывание/выдача сигнала. Так как нам нужно не считывать, а наоборот генерировать сигнал на пинах, то используем режим ***pinMode(pin, OUTPUT).***

Далее пишем код для основной логики работы. Для начала включим красный светодиод, для этого необходимо подать *цифровой сигнал* на необходимый пин. *Цифровой сигнал простыми словами — это последовательность цифровых значений или байтов информации — 0 и 1. За определённый период времени информация представляется только в определённом значении.* Для генерации цифрового сигнала используется метод ***digitalWrite(pin, HIGH),*** а для прекращения генерации сигнала и соответственно выключения светодиода используется запись ***digitalWrite(pin, LOW).***

Из-за того, что код выполняется очень быстро, мы вынуждены ненадолго останавливать всю программу. Для этого используется метод ***delay(time).*** Важно, что значение time подается в миллисекундах, а не в секундах, как это обычно принято. Другими словами, если мы хотим приостановить выполнение цикла loop на 1 секунду необходима запись: delay(1000).

Для реализации функции «мигания светодиода» можно использовать линейную структуру, когда все действия и задержки прописываются друг за другом, а можно использовать более оптимальный способ – *использовать цикл for*.

Цикл for простыми словами — это цикл, который выполняется заданное количество раз. Его используют, когда количество итераций (повторов) известно заранее.

После написания всех строк кода модно записать и прошить плату, нажав на соответствующие кнопки в Arduino IDE.

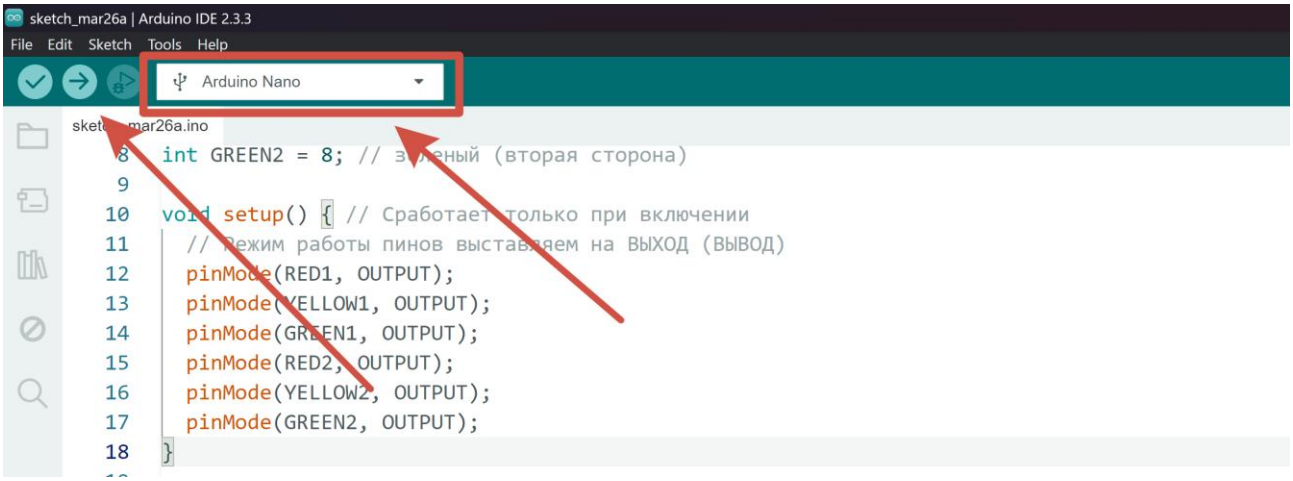

Рисунок 8 - нахождение кнопок прошивки и выбора платы в Arduino IDE

После успешной прошивки и правильной сборке должно получиться что-то похожее на то, что представлено на рисунке 6 (см. выше).

3.2 Возможные ошибки

При сборке и программировании Arduino могут возникнуть некоторые ошибки.

Самой распространенной считается ошибка при сборке схемы – перепутаны полярности у светодиодов. Для исправления достаточно попробовать поменять местами ноги у компонента светодиода местами.

Еще одна распространенная ошибка возникает при попытке прошивки платы микроконтроллера. Если в меню Arduino IDE выбран неправильный тип платы или неправильный порт прошивки. Для исправления достаточно перепроверить правильность значений или переворкнуть провод из компьютера.