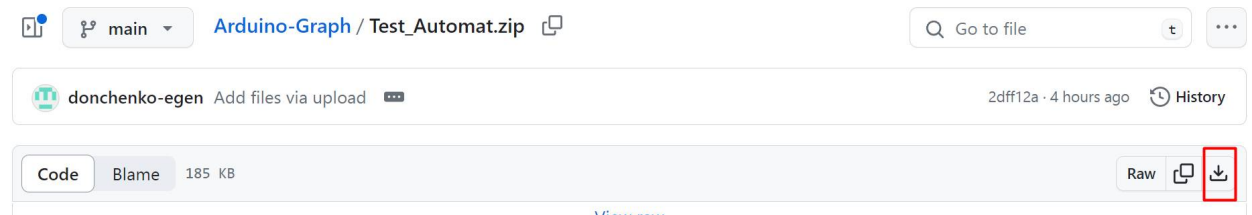
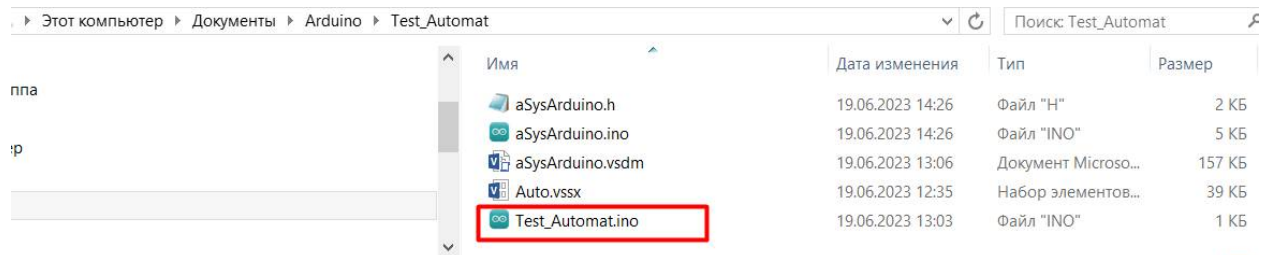


## Быстрый старт

1. Скачайте актуальный пример и распакуйте его.





2. Создайте свой проект Ардуино. Можно использовать готовый проект из примера:



Можно переименовать его. Главное условие – папка и проект должны иметь одинаковое имя.

Так же вы можете просто скопировать два файла

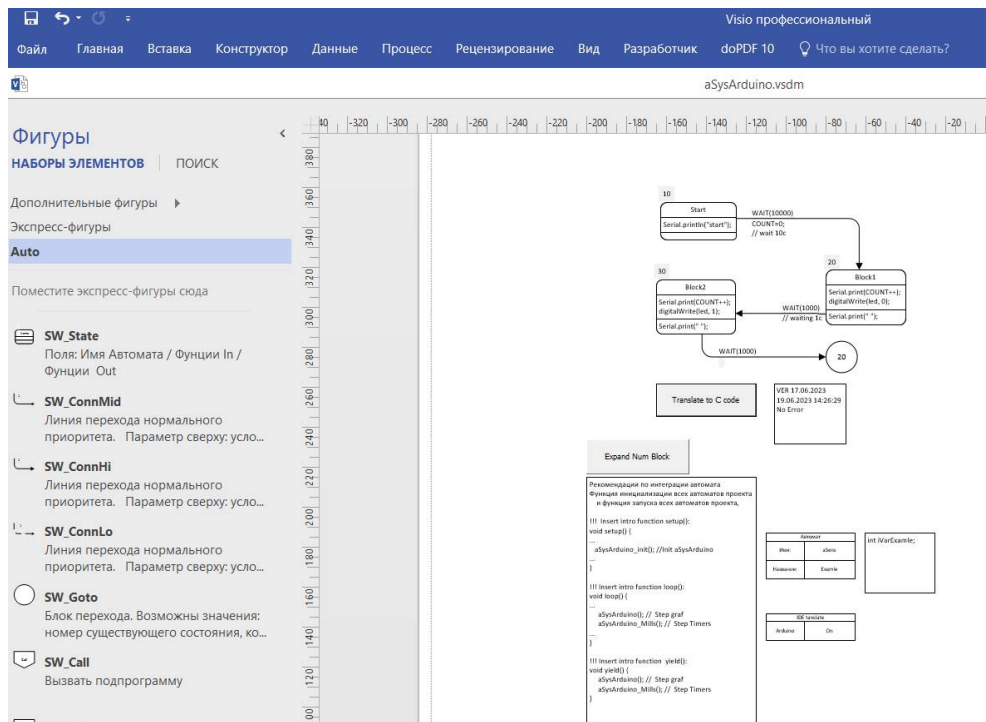
 aSysArduino.vsdm	19.06.2023 13:06	Документ Microso...	157 КБ
 Auto.vssx	19.06.2023 12:35	Набор элементов...	39 КБ

в папку своего проекта.

Файл aSysArduino.vsdm можно переименовать по собственному усмотрению.

3. В системе обязательно должно быть установлено Microsoft Visio версии 16 и выше.

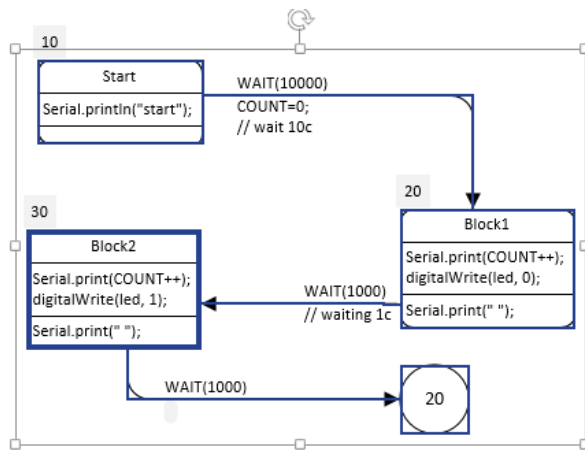
Тогда файл aSysArduino.vsdm будет открываться на редактирование.



Слева на панели размещен набор фигур из файла Auto.vssx.

#### 4. Удалим ненужные объекты.

Для этого выделите и удалите нумерованные объекты и соединяющие их дуги.



Остальные объекты остаются

Translate to C code

VER 17.06.2023  
 19.06.2023 14:26:29  
 No Error

Expand Num Block

Рекомендации по интеграции автомата  
 Функция инициализации всех автоматов проекта  
 и функция запуска всех автоматов проекта,

```

      !!! Insert intro function setup():
      void setup() {
      ...
      aSysArduino_init(); //Init aSysArduino
      ...
      }

      !!! Insert intro function loop():
      void loop() {
      ...
      aSysArduino(); // Step graf
      aSysArduino_Mills(); // Step Timers
      ...
      }

      !!! Insert intro function yield():
      void yield() {
      aSysArduino(); // Step graf
      aSysArduino_Mills(); // Step Timers
      ...
      }
        
```

Автомат	
Имя:	aSens
Название:	Examlle

int iVarExamlle;

IDE tanslate	
Arduino	On

На самом деле, их тоже можно удалить, кроме кнопок (в последствии придется их разместить с панели), но эта работа не имеет смысла.

5. Выполним настройку графа. Для этого изменим значения в таблице «Автомат»

Автомат		Автомат	
Имя:	aSens	Имя:	aSensor
Название:	Examlle	Название:	Тестовый автомат

Эта действие совершенно не обязательное, влияет на имя результирующего прораамного объекта и комментариев к нему в программном коде. Большинство пользователей туда не заглядывают.

6. Настроим (если нужно) блок переменных.

int iVarExamlle;

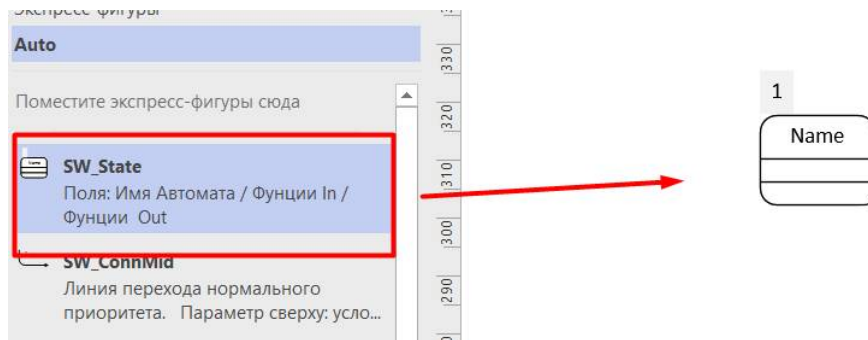
В этот текстовый блок мы можем размещать определения переменных, типов, массивов и т.д., которые будут использованы в вашем автомате. Блок сделан для удобства быстрого программирования, на самом деле вам доступны все переменные из центрального файла проекта.

7. Убедимся в наличии флага Ардуино.

IDE tanslate	
Arduino	On

Если вы его не удаляли, то он на месте. А вот если таблицу убрать, то результатами компиляции станут файлы проекта для среды проектирования Keil, впрочем они отличаются незначительно.

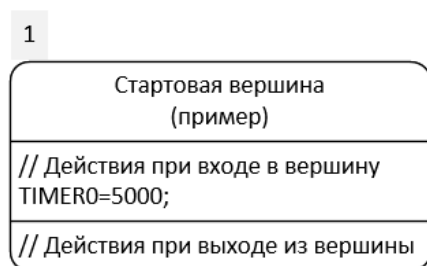
8. Начинаем создание программы. Разместим стартовую вершину.



Для этого перетащим с панели на рабочую область «вершину» (она же «состояние») графа.

Собственно, работа программного графа и заключается в том, что в любой момент времени он присутствует ТОЛЬКО в одной вершине, т.е. имеет только одно, строго определенное состояние. В этом заключается одно из крупнейших достоинств программирования методом «Графов» - вы всегда можете узнать, чем конкретно занимается ваша программа и как она к этому состоянию пришла.

9. Заполняем стартовую вершину.



Верхняя часть блока – Название. Играет роль комментария к состоянию, на работу программы влияние не оказывает.

Средняя часть блока - код IN, который выполняется однократно при ВХОДЕ в вершину (состояние) из вне. Для стартового блока – при запуске автомата. Если к вершине подключена дуга, которая возвращается обратно на саму же вершину, то код IN повторно не выполняется!

`TIMER0 = 5000;`

Забегая вперед, укажем на наличие макрокоманд, одной из которых является `TIMER0-9` (итого 10 программных таймеров). При выполнении программы, они автоматически уменьшаются на единицу каждую 1миллисекунду, т.е. за 1 секунду переменная

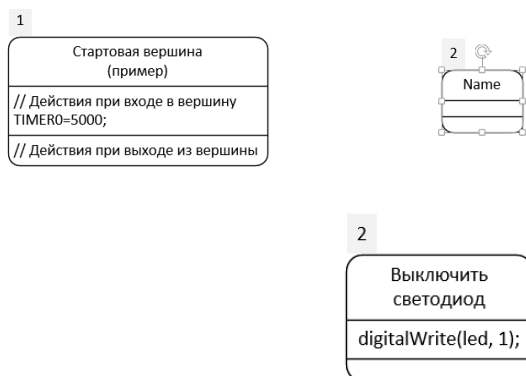
уменьшиться на 1000, естественно, если значение таймера больше чем 1000. В примере программный таймер 0 обнулиться через 5 секунд.

Нижняя часть блок – код OUT, который выполняется однократно при полном ВЫХОДЕ из вершины. Если к вершине подключена дуга, которая возвращается обратно на саму же вершину, то код OUT повторно не выполняется!

## 10. Заполняем вершину 2

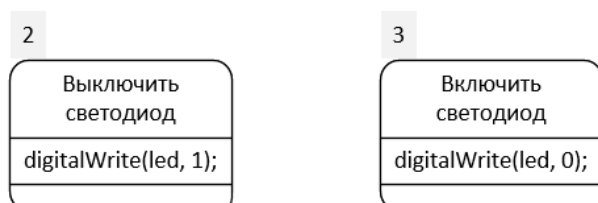
Тут уже начинается творчество. Правильно продумать алгоритм действий автомата – задача не самая простая. Но пользователю предоставлено достаточно средств, что бы максимально упростить ее.

Перетаскиваем с панели вершину 2.

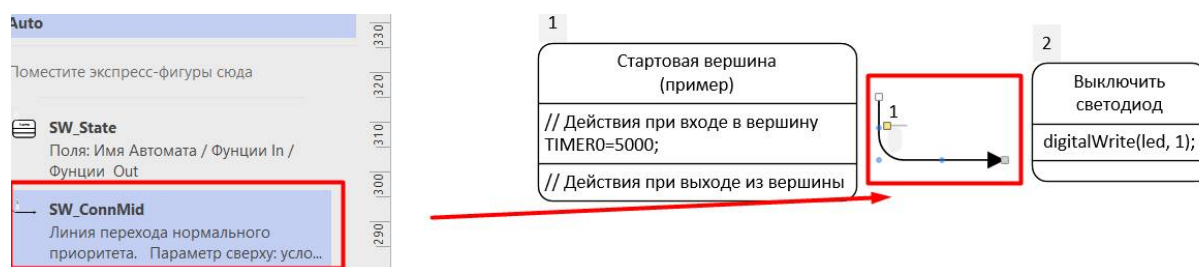


Заполняем имя вершины

## 11. Реализуем алгоритм мигания светодиода на двух вершинах, для чего добавляем вершину 3

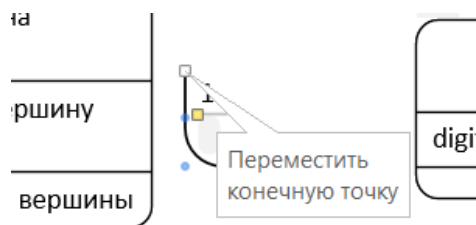


## 12. Соединим вершины (состояния) дугами переходов.

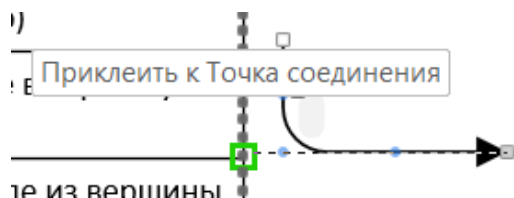


Перетаскиваем с панели дугу на свободное место. Рекомендуется использовать по умолчанию ConnMid, имеющий нормальный приоритет. (О приоритетах поговорим позднее).

Дуга имеет две точки – начало и конец (обозначен стрелкой). За эти точки дугу можно перемещать и подключать к блокам.

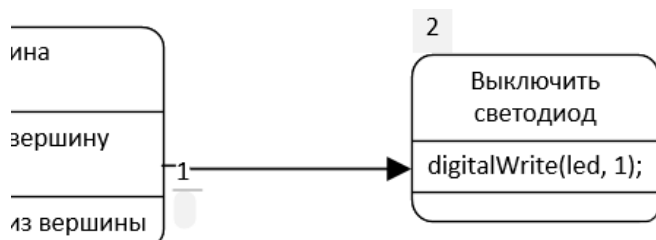


Зажав точку мышкой, перемещаем ее к блоку 1.



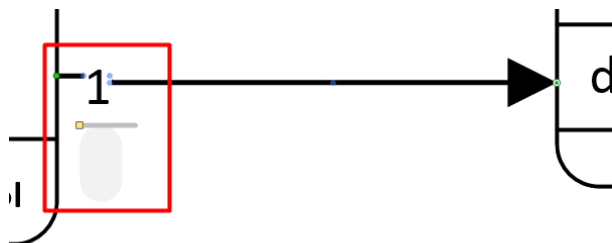
Присоединяем к удобной точке.

Повторяем операцию с концом дуги.



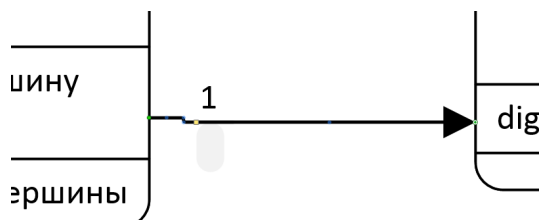
13 Перемещаем условие перехода между узлами

Для этого щелкните мышкой по дуге (в любом месте, кроме краев)



Условие перехода получит желтую точку привязки, за которую можно ухватиться мышкой.

И аккуратно переместить на линию дуги:



Во время операций подключения удобно изменять масштабирование документа – удерживая нажатой клавишу «ctrl» вращать ролик мыши. При этом соединение получается быстро и аккуратно.

14. Определяем условие перехода.

По умолчанию там стоит 1.

После трансляции дуга превратиться в команду на языке си:

```
if (1)
{
    SW_aSensor.uNewState = 2; // goto    Выключить светодиод
}
```

Понятно, что 1 в условии if приведет к безусловному выполнению перехода на состояние 2.

Вы можете ввести собственное условие. Это может быть константа, переменная или даже функция. Главное – что бы ее синтаксически верно можно было использовать с командой си:

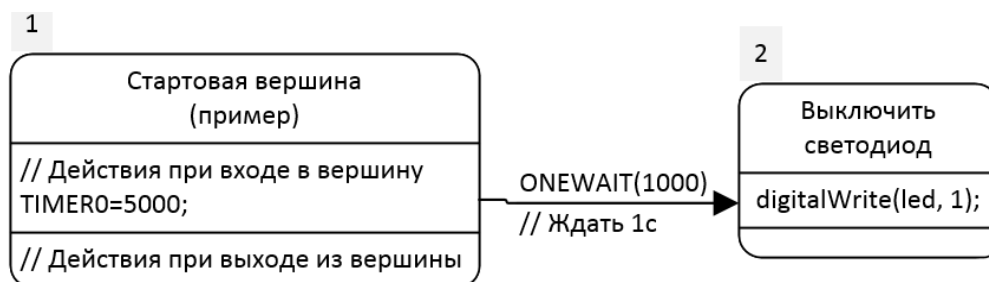
“if(условие)”.

## 15. Макрокоманды в условиях перехода.

Самая распространенная макрокоманда – WAIT(время), где время измеряется в тысячных долях секунды, т.е. 1000 будет соответствовать ожиданию в 1с. Использовать макрокоманду просто:



Альтернативно, можно применить команду ONEWAIT, действие которой при переходе на соседний блок аналогично.



Разница между этими командами будет наблюдаться только для дуг, которые не «покидают» вершину – WAIT выполняет дугу многократно, а ONEWAIT – только один раз.

Почему WAIT() считается макрокомандой? Рассмотрим результат трансляции:

```

case 1: // Стартовая вершина (пример)
    if (is_SW_aSensor_Wait(1))
    {
        SW_aSensor.uNewState = 2; // goto  Выключить светодиод
        // Ждать 1с
        set_SW_aSensor_Wait(1,1000); // Multi Wait Set 1000ms
    }
    break;
case 2: // Выключить светодиод

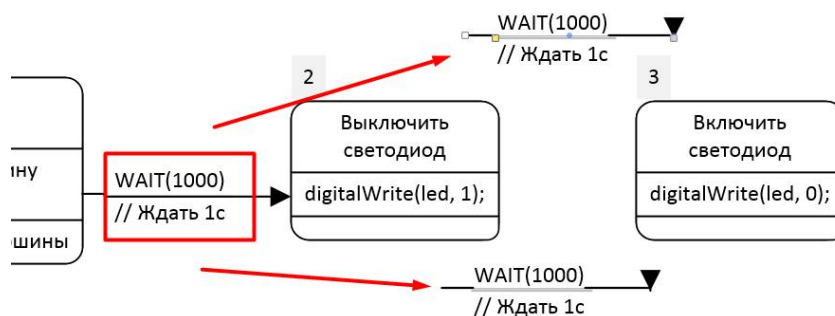
```

Как видно, на самом деле в операции if() проверяется значение функции состояния программного таймера, привязанного к текущему состоянию, а сам программный таймер, незримо для пользователя, уменьшает свое значение каждую тысячную секунды.

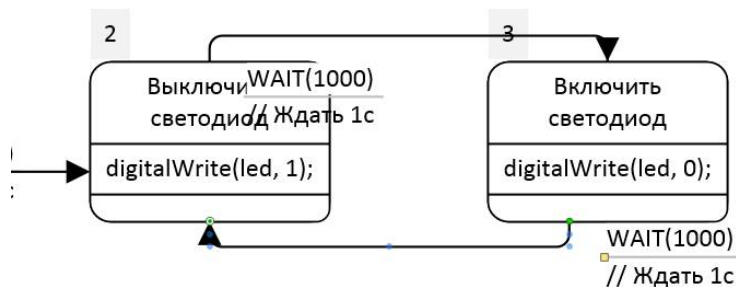
Все эти подробности скрыты, программисту достаточно указать WAIT(время) и требуемая задержка будет реализована. В отличие от delay(), задержка не прерывает выполнение программы, пауза распространяется исключительно на логику перехода между состояниями.

16. Реализуем паузы в состояниях светодиодов.

Можно создать дуги, перетаскивая их из панели и заново оформляя. А можно просто скопировать готовую дугу, наполнение которой проще редактировать.

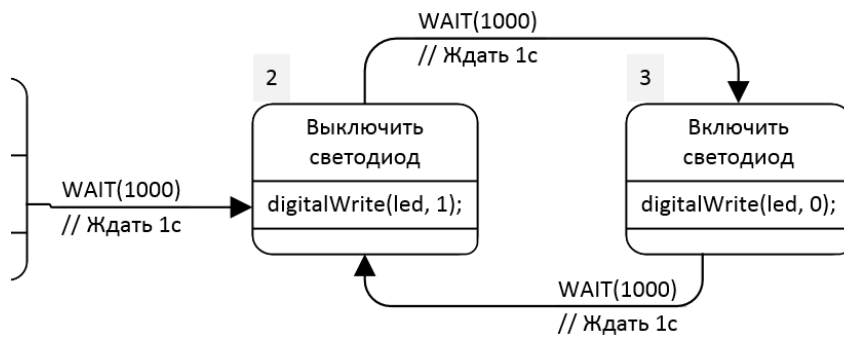


Удерживая “ctrl” перетаскиваю «образцовую» дугу на свободное место. И подключаем ее, не забывая выполнять масштабирование при тонких операциях с графикой.

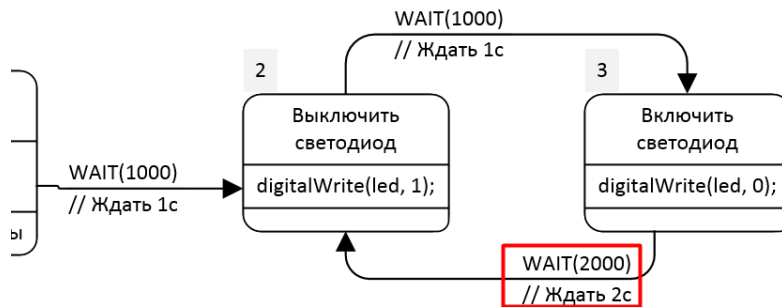


Получилось не очень красиво. Растянем дуги вверх/вниз и переместим условия переходов:





17. Откорректируем время свечения.



Пусть, например, светодиод будет гореть 2с, а в потушенном состоянии пребывает 1с.

Собственно говоря, программа уже готова! Осталось ее правильно интегрировать в основную программу ардуино.

18. Интеграция в ардуино.

Нажимаем на кнопку “Translate to C code”. Все необходимые файлы будут созданы и подключены в проект ардуино. Одновременно будет заполнено поле «Рекомендации по интеграции автомата».

```

Рекомендации по интеграции автомата
Функция инициализации всех автоматов проекта
и функция запуска всех автоматов проекта,

!!! Insert intro function setup():
void setup() {
...
  aSysArduino_init(); //Init aSysArduino
...
}

!!! Insert intro function loop():
void loop() {
...
  aSysArduino(); // Step graf
  aSysArduino_Mills(); // Step Timers
...
}

!!! Insert intro function yield():
void yield() {
  aSysArduino(); // Step graf
  aSysArduino_Mills(); // Step Timers
}
  
```

Оно уже содержит все необходимые команды, в привязке к заданному вами имени автомата. Их нужно скопировать и вставить в требуемые строки основного файла.

```

1  const int led = 16;
2
3  void setup(void) {
4      pinMode(led, OUTPUT);
5      digitalWrite(led, 0);
6      Serial.begin(115200);
7      Serial.println("");
8
9      aSysArduino_init(); //Init aSysArduino
10
11     Serial.println("aSysArduino_init");
12 }
13
14 void loop(void) {
15     aSysArduino(); // Step graf
16     aSysArduino_Mills(); // Step Timers
17 }
18
19 void yield() {
20     aSysArduino(); // Step graf
21     aSysArduino_Mills(); // Step Timers
22 }
23

```

Вывод    Монитор порта

Скетч использует 266208 байт (25%) памяти устройства. Всего доступно 1044464 байт.  
Глобальные переменные используют 27000 байт (32%) динамической памяти, оставляя 54926

Результат компиляции – положительный.

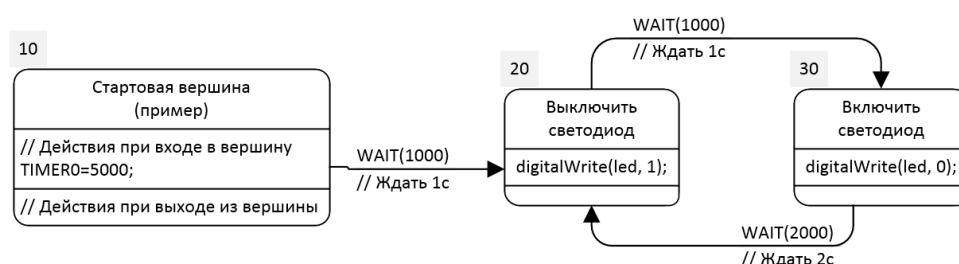
Как видно из примера, в основной файл проекта вставлено три функции:

- 1) инициализации, в setup()
- 2) шаг автомата (выполняет однократный прогон по автомату)
- 3) подсчет таймеров, на основе данных mills().

Так же рекомендовано использовать функцию yeild(), тело которой выполняется, когда в блоке loop стоит delay(), что нам позволит не потерять таймеры при длительных паузах.

## 19. Оптимизация номеров вершин.

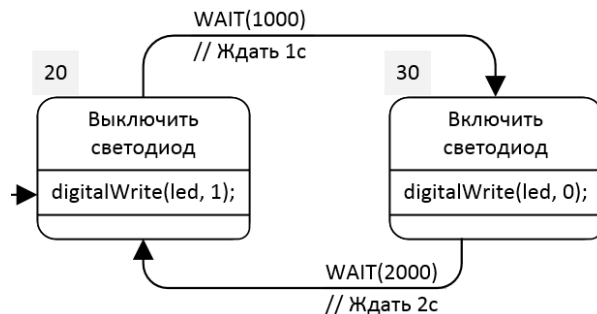
Как видно из рисунка, по умолчанию нам присвоили номера 1, 2 и 3 для созданных вершин. Но что делать, если в будущем нам понадобится изменить программу и добавить еще одну вершину, скажем между 1 и 2? Будет вершина 47? (Программы бывают очень большие). Используем кнопку “Expand Num Block”:



Все вершины были перенумерованы с шагом 10.

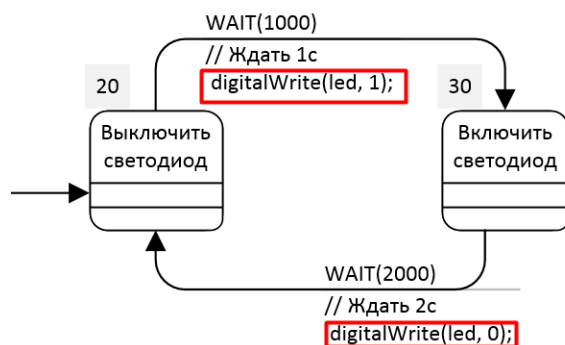
## 20. Автоматы Мили и Мура.

Теория графов, в приложении к созданию автоматов, упоминает автоматы Мили и автоматы Мура. Так, Автомат Мили:



Действия происходят только в состояниях (в примере – при входе).

Напротив, действия автомата Мура выполняются на переходах:

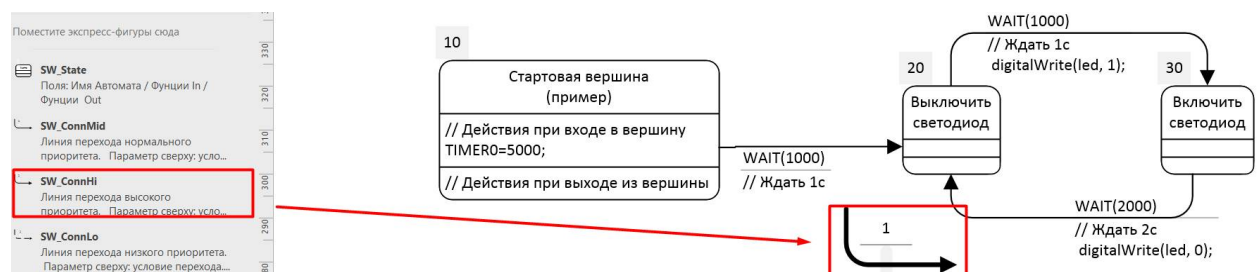


Результат тот же самый.

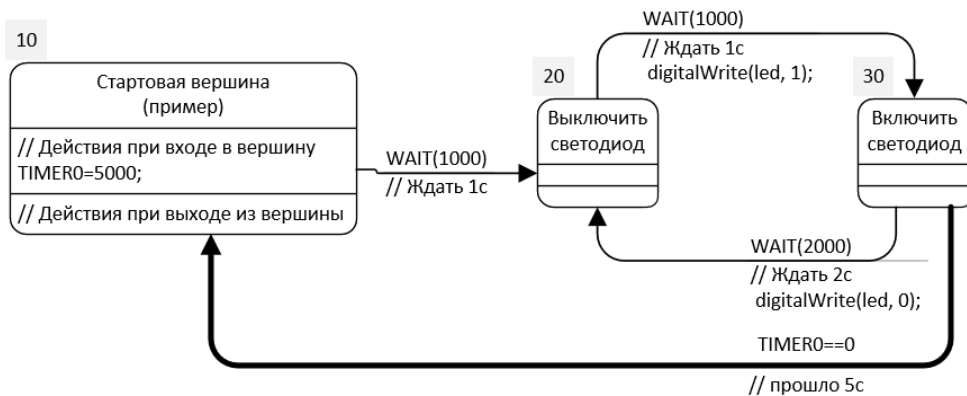
Никаких ограничений на использование любого из подходов к созданию автомата нет. Вы можете выполнять действия как при входе (выходе) в вершину, так и непосредственно на переходе. Единственно, в таблицу блока вершины писать код будет несколько проще, не понадобится потом сдвигать блок условия, чтобы соблюсти «красоту».

## 21. Приоритеты переходов.

Перетащим с панели на свободное место дугу ConnHi:



Подключаем линию перехода:



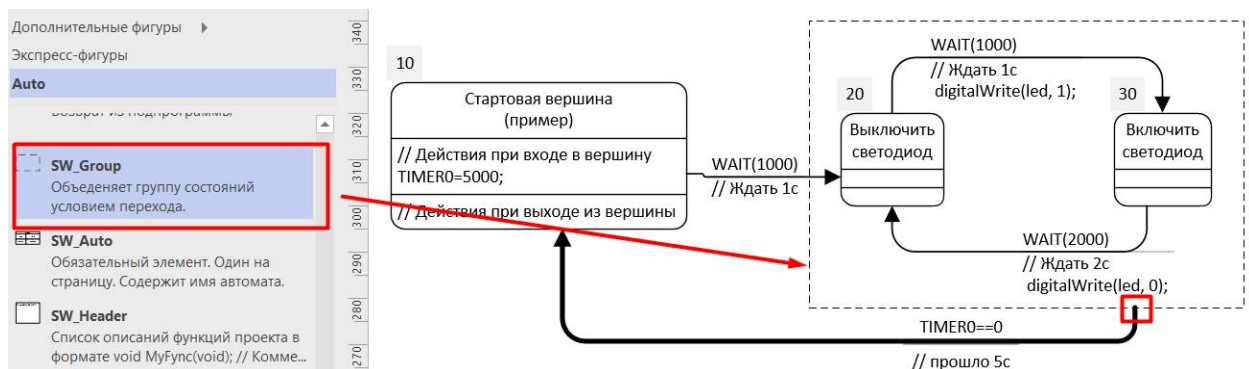
Что было сделано? Добавлен переход высшего приоритета, из вершины 30 в вершину 10. Условие перехода – обнуление программного таймера 0, который должен произойти спустя 5 секунд после первого входа в вершину 10.

Фактически, приоритет определяет очередность проверки условий на выход из блока 30. Сначала проверяется высший (толстая линия), затем нормальный (средняя линия), а в конце – низший приоритет (пунктирная тонкая дуга). Результат трансляции для нашего примера:

```
case 30: // Включить светодиод
    if (SW_aSensor.uTimer0==0)
    {
        SW_aSensor.uNewState = 10; // goto Стартовая вершина (пример)
        // прошло 5с
    }
    else
    if (is_SW_aSensor_Wait(1))
    {
        SW_aSensor.uNewState = 20; // goto Выключить светодиод
        // Ждать 2с
        digitalWrite(led, 0);
        set_SW_aSensor_Wait(1,2000); // Multi Wait Set 2000ms
    }
    break;
```

## 22. Групповые переходы.

Недостаток предыдущего решения – мы контролируем таймер только в вершине 30. Если обнуление таймера произойдет в вершине 20, мы это обнаружим только при переходе обратно на 30. Обычно это не критично. Но не всегда. Тогда нам на выручку приходят «групповые переходы»



Перетаскиваем пунктирный прямоугольник на рабочую область и изменяем ее размеры так, чтобы вершины 20 и 30 полностью разместились внутри. Переподключаем вход дуги в удобном месте группового прямоугольника (к свободной точке).

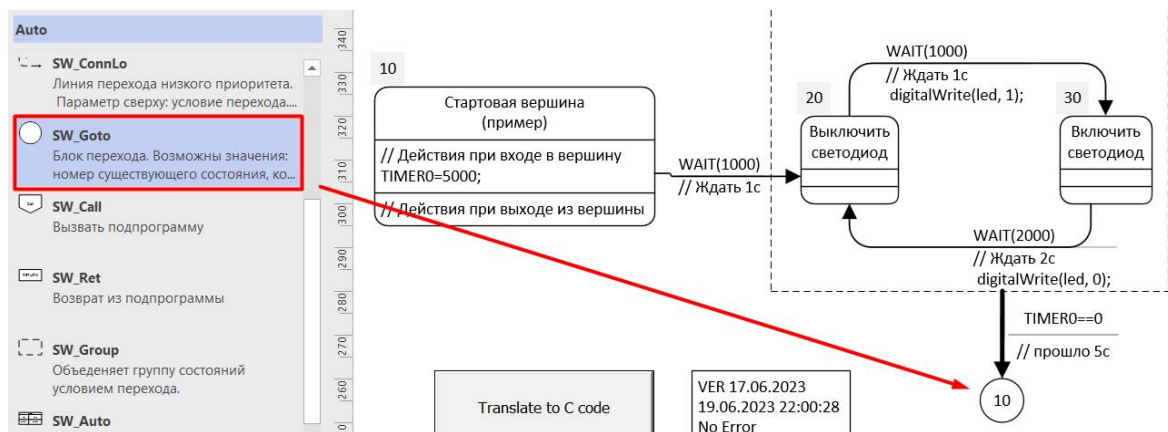
Размеры группового прямоугольника не должны быть чрезмерными – чем ближе к отслеживаемым блокам, тем лучше. Результат трансляции:

```
case 20: // Выключить светодиод
    if (SW_aSensor.uTimer0==0)
    {
        SW_aSensor.uNewState = 10; // goto Стартовая вершина (пример)
        // прошло 5с
    }
    else
    if (is_SW_aSensor_Wait(1))
    {
        SW_aSensor.uNewState = 30; // goto Включить светодиод
        // Ждать 1с
        digitalWrite(led, 1);
        set_SW_aSensor_Wait(1,1000); // Multi Wait Set 1000ms
    }
    break;
case 30: // Включить светодиод
    if (SW_aSensor.uTimer0==0)
    {
        SW_aSensor.uNewState = 10; // goto Стартовая вершина (пример)
        // прошло 5с
    }
    else
    {
        // ...
    }
}
```

Команды групповых переходов размещены в каждой отмеченных вершин, причем в начале проверки, что соответствует высшему приоритету.

### 23. Переход по метке

Иногда дуга получается огромной длинны. Или расположена неудобно, пересекает другие дуги и запутывает восприятие. На помощь приходят блоки GoTo.



Результат трансляции неизменный – использование метки позволяет перейти к вершине 10 без необходимости пересекать пол-экрана.