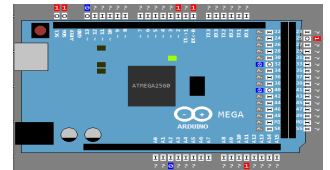


# UnoArduSimV2.8.2 Полная Помощь



## Оглавление

### [обзор](#)

### [Панель с кодом, Настройки и Изменить/Просмотреть](#)

#### [Панель с кодом](#)

#### [Настройки](#)

#### [Изменить/Просмотреть](#)

### [Панель с переменными и Изменить/Отслеживать Переменная окно](#)

### [Лабораторная панель](#)

#### ['Uno' или 'Mega'](#)

#### ['I/O' Устройства](#)

##### [Serial Монитор \('SERIAL'\)](#)

##### [чередовать Серийный \('ALTSER'\)](#)

##### [Карта памяти SD \('SD\\_DRV'\)](#)

##### [TFT Дисплей \('TFT'\)](#)

##### [Конфигурируемый SPI Ведомый \('SPISLV'\)](#)

##### [Двухпроводная I2C Ведомый \('I2CSLV'\)](#)

##### [Текстовый LCD I2C \('LCDI2C'\)](#)

##### [Текстовый LCD SPI \('LCDSPI'\)](#)

##### [Текстовый LCD D4 \('LCD\\_D4'\)](#)

##### [Мультиплексор LED I2C \('MUXI2C'\)](#)

##### [Мультиплексор LED SPI \('MUXSPI'\)](#)

##### [Порт Расширения SPI \('EXPSPi'\)](#)

##### [Порт Расширения I2C \('EXPI2C'\)](#)

##### ['1-Wire' Ведомый \('OWIISLV'\)](#)

##### [Регистр Сдвига Ведомый \('SRSLV'\)](#)

##### [Программируемый 'I/O' Устройство \('PROGIO'\)](#)

##### [Один-Сигнал \('1SHOT'\)](#)

##### [Цифровой Генератор Импульсов \('PULSER'\)](#)

##### [Аналоговый Генератор Функций \('FUNCGEN'\)](#)

##### [Шаговый двигатель \('STEPR'\)](#)

##### [Пулсирующий Шаговый двигатель \('PSTEPR'\)](#)

##### [Двигатель постоянного напряжения \('MOTOR'\)](#)

##### [Серводвигатель \('SERVO'\)](#)

##### [Пьезоэлектрический Динамик \('PIEZO'\)](#)

##### [Слайд резистор \('R=1K'\)](#)

##### [Кнопка Тактовая \('PUSH'\)](#)

##### [Цветной LED \('LED'\)](#)

##### [4-LED ряд \('LED4'\)](#)

##### [7-сегментный LED Цифра \('7SEG'\)](#)

##### [Аналоговый Слайдер](#)

##### [Pin Перемычка \('JUMP'\)](#)

### [меню](#)

#### [Файл:](#)

##### [Загрузить INO или PDE Prog \(Ctrl-L\)](#)

##### [Изменить/Просмотреть \(Ctrl-E\)](#)

##### [Сохранить](#)

##### [Сохранить как следующий \('#include'\)](#)

##### [предыдущий](#)

##### [Выход](#)

## Найти:

Ascend Call Stack  
Стек вызовов Descend  
Установить текст Искать (Ctrl-F)  
Найти Следующий текст  
Найти Предыдущий текст

## Выполнить:

Шаг с заходом (F4)  
Шаг с обходом (F5)  
Шаг с выходом (F6)  
Выполнить До (F7)  
Выполнить Пока (F8)  
Выполнить (F9)  
Приостановить (F10)  
Сброс  
Анимация  
Замедленное движение

## Опции:

Шаг с обходом Structors/ Операторы  
Регистр-Allocation  
Ошибка при неинициализированном  
добавленной 'loop()' задержка  
Разрешить вложенные прерывания

## Конфигурировать:

'I/O' Устройства  
Настройки

## ПеремОбновить:

Разрешить авто (-) Сокращаться  
минимальная  
Основной момент изменения

## Окна:

Serial Монитор  
Восстановить все  
Pin Цифровые Осциллограммы  
Pin Аналоговый Осциллограммо

## Помощь:

Быстрый Помощь Файл  
Полный Помощь Файл  
Исправления Ошибка  
Изменение / Улучшение  
Около

'Uno' или 'Mega' Плата и 'I/O' Устройства

Синхронизация

'I/O' Устройство Синхронизация

Звуки

## Ограничения и неподдерживаемые элементы

Включено Файлы

Динамическое распределение памяти и оперативная память

'Flash' Распределение памяти

'String' Переменные

Библиотеки Arduino

указатели

'class' а также 'struct' Объектов

Сфера

Отборочные 'unsigned', 'const', 'volatile', 'static'

Директивы Компилятор

[Ардуино-языковые элементы](#)

[C / C ++ - языковые элементы](#)

[Функциональный модуль Шаблоны](#)

[Эмуляция в реальном времени](#)

[Примечания к выпуску - начиная с версии V2.5](#)

[Исправления Ошибка](#)

[V2.8.2- сентябрь 2020](#)

[V2.8.1- июнь 2020](#)

[V2.8.0- июнь 2020](#)

[V2.7- март 2020](#)

[V2.6.0- | январь 2020](#)

[V2.5.0- октябрь 2019](#)

[Чанг ES / Улучшения](#)

[V2.8.2- сентябрь 2020](#)

[V2.8.0- июнь 2020](#)

[V2.7 марта 2020 г.](#)

[V2.6.0 январь 2019](#)

[V2.5.0 Oct 2019](#)

## обзор

UnoArduSim является бесплатной **реальное время** (видеть для Синхронизация **ограничения**) инструмент симулятор, который я разработал для студента и энтузиаста Arduino. Он предназначен для того, чтобы вы могли экспериментировать и легко отлаживать Arduino программы **без необходимости какого-либо фактического оборудования**, Он нацелен на **Arduino 'Uno' или 'Mega'** плата, и позволяет выбирать из набора виртуальных 'I/O' Устройства, а также настраивать и подключать эти Устройства к вашему виртуальному 'Uno' или 'Mega' в **Лабораторная панель**, - вам не нужно беспокоиться об ошибках проводки, обрыве / разрыве соединений или неисправности Устройства, которые мешают разработке и тестированию программа.

UnoArduSim предоставляет простые сообщения об ошибках для любых ошибок разобрать или выполнение, с которыми он сталкивается, и позволяет выполнять отладку с помощью **Сброс**, **Выполнить**, **Выполнить До**, **Выполнить Пока**, **Приостановить** и гибкий **Шаг** операции в **Панель с кодом** с одновременным просмотром всех глобальных и в настоящее время активных локальных переменные, массивы и объектов в **Панель с переменными**, Предусмотрена проверка границ массив во время Выполнить, и будет обнаружено переполнение памяти АТмега (и строка программа виновника выделена!). Любые электрические конфликтует с, прикрепленные к 'I/O' Устройства, помечаются и сообщаются по мере их появления.

Когда INO или PDE программа файл открыт, он загружается в программа **Панель с кодом**, программа затем дается Анализировать, чтобы преобразовать его в исполняемый файл, который затем готов к **смоделированный выполнение** (в отличие от Arduino.exe, автономный исполняемый файл двоичном не создано) Любая ошибка разобрать обнаружена и помечена путем выделения строки, которая не прошла разобрать, и сообщения об ошибке на **Статус бар** в самом низу приложения UnoArduSim окно.

**Изменить/Просмотреть** окно можно открыть, чтобы вы могли просматривать и редактировать выделенную синтаксисом версию вашего пользователя программа. Ошибки во время смоделированного выполнения (такие как несовпадающий скорость передачи) сообщаются в строке состояния и через всплывающее окно сообщения.

UnoArduSim V2.7 является практически полной реализацией **Язык программирования Arduino V1.8.8 как документально подтверждено на [arduino.cc](http://arduino.cc)**, Веб-страница со справочником по языку, а также с дополнениями, указанными на странице версии Загрузка Примечания к выпуску. Хотя UnoArduSim не поддерживает полную реализацию C ++, которую поддерживает Arduino.exe, лежащий в основе GNU компилятор, вполне вероятно, что только самые продвинутые программисты обнаружат, что какой-то элемент C / C ++, который они хотят использовать, отсутствует (и, конечно, всегда есть простые кодирование обходных путей для таких отсутствующих функций). В общем, я поддерживал только то, что, по моему мнению, является наиболее полезным C / C ++ для любителей и студентов Arduino - например, 'enum' а также '#define' поддерживаются, но указатели функциональный модуль - нет. Даже если пользовательский объектов ('class' а также 'struct') и (большинство) перегрузок операторов поддерживаются, **множественное наследование не**,

Поскольку UnoArduSim является языковым симулятором высокого уровня, **поддерживаются только операторы C / C ++**, **заявления на ассемблере не**, Точно так же, потому что это не машинная симуляция низкого уровня, **Регистры АТмега328 недоступны для вашей программа** для чтения или записи, хотя распределение, передача и возврат регистров эмулируются (если вы выбираете это в меню **Опции**).

Начиная с версии 2.6 UnoArduSim имеет автоматическую поддержку встроенный для ограниченного подмножества библиотек, предоставляемых Arduino, а именно: 'Stepper.h', 'Servo.h', 'SoftwareSerial.h', 'SPI.h', 'Wire.h', 'OneWire.h', 'SD.h', 'TFT.h' и 'EEPROM.h' (версия 2). V2.6 представляет механизм для 3<sup>й</sup> поддержка партийной библиотеки через файлы, предоставленную в 'include\_3rdParty' папка это можно найти в каталоге установки UnoArduSim. Для любой '#include' других (т.е. пользовательских) библиотек, UnoArduSim будет **не** найдите обычную структуру каталогов установки Arduino, чтобы найти библиотеку; вместо тебя **нужно** скопировать соответствующий заголовок (".h") и исходный (".cpp") файл в тот же каталог, что и программа файл, над которым вы работаете (при условии, конечно, ограничения, что содержимое любого '#include' файл должен быть полностью понятным для UnoArduSim синтаксический анализатор).

Я разработал UnoArduSimV2.x в QtCreator с поддержкой нескольких языков, и в настоящее время он доступен только для Окна™, Портинг на Linux или MacOS, это проект на будущее! UnoArduSim выросла из симуляторов, которые я разрабатывал на протяжении многих лет для курсов, которые я преподавал в

Университете Королевы, и он был протестирован достаточно обширно, но там наверняка будет несколько ошибки, которые все еще прячутся там. Если вы хотите сообщить о ошибка, пожалуйста, опишите его (кратко) по электронной почте [unoArduSim@gmail.com](mailto:unoArduSim@gmail.com) а также **не забудьте приложить полный исходный код ошибка-Arduino, вызывающий ошибка** так что я могу скопировать ошибка и исправить его. Я не буду отвечать на отдельные отчеты ошибка, и у меня нет гарантированных сроков исправлений в следующем выпуске (помните, что обходные пути почти всегда есть!).

Ура,

Стэн Симмонс, PhD, P.Eng.

Доцент (в отставке)

Кафедра электротехники и вычислительной техники

Университет королевы, Кингстон, Онтарио, Канада



## Панель с кодом, Настройки и Изменить/Просмотреть


(В сторону: образец окна, показанный ниже, все под выбранной пользователем Окна-OS цветовая тема, которая имеет темно-синий цвет фона окна).

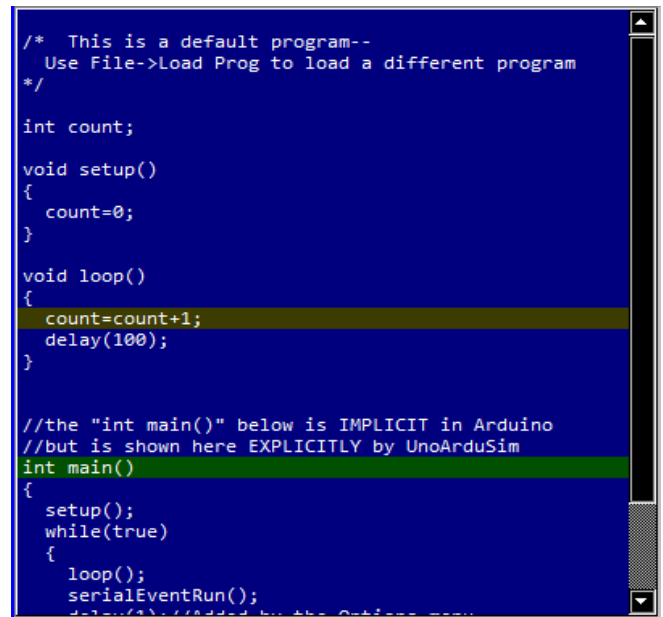
### Панель с кодом

**Панель с кодом** отображает вашего пользователя программа, и **зеленый** Подсветка треков выполнение. (или основные моменты **красный** за ошибку)

После того, как загруженный программа имеет успешный Анализировать, первая строка в 'main()' выделен, и программа готов к выполнение. Обратите внимание, что 'main()' неявно добавляется Arduino (и UnoArduSim), и вы делаете **не** включите его как часть вашего пользователя программа файл. Выполнение находится под контролем меню **Выполнить** и связанные с ним **Инструмент-Var** кнопки и комбинации клавиш функциональный модуль.

После перехода выполнение по одной (или более) инструкции (вы можете использовать **Инструмент-Var** кнопки **, , или** ), строка программа, которая будет следующей выполнил, затем будет выделена зеленым цветом - зеленая линия всегда будет следующей строкой **готов к выполнил** ,

Если программа выполнение в настоящее время остановлен, и вы нажимаете в **Панель с кодом** окно, линия, которую вы только что щелкнули, будет выделена темно-оливковым цветом (как показано на рисунке) - следующая за выполнил линия всегда будет выделена зеленым цветом (начиная с V2.7). Но вы можете вызвать выполнение *прогрессировать до* линия, по которой вы только что нажали затем нажав на **Выполнить До**  **Инструмент-Var** кнопка. Эта функция позволяет вам быстро и легко достичь определенных линий в программа, чтобы впоследствии вы могли шаг за шагом проходить интересующую вас часть программа.





```
/* This is a default program--
   Use File->Load Prog to load a different program
*/






int count;

void setup()
{
  count=0;
}

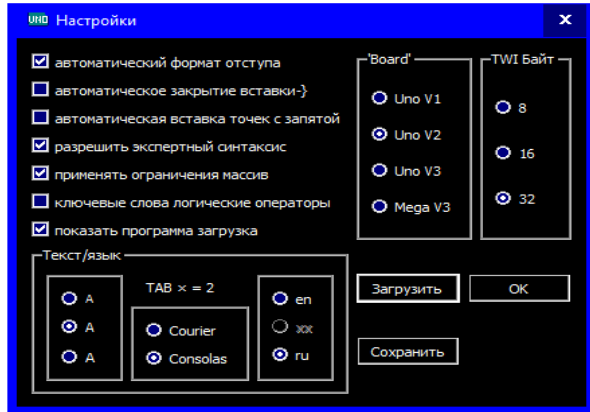
void loop()
{
  count=count+1;
  delay(100);
}

//the "int main()" below is IMPLICIT in Arduino
//but is shown here EXPLICITLY by UnoArduSim
int main()
{
  setup();
  while(true)
  {
    loop();
    serialEventRun();
  }
}
```

Если ваш загруженный программа имеет какие-либо '#include' файлы, вы можете перемещаться между ними с помощью **Файл | предыдущий** а также **Файл | следующий** (с **Инструмент-Var** кнопки  а также  ). Последняя строка, выбранная пользователем в каждом из этих модулей, остается выделенной и определяет возможную строку точка останова, к которой будет выполняться, но только точка останова в *текущий отображаемый модуль* активен на следующем **Выполнить До** ,

**Найти** действия в меню позволяют **ИЛИ** найти текст в **Панель с кодом** или **Панель с переменными** ( **Инструмент-Var** кнопки  а также  или сочетания клавиш **стрелка вверх** а также **стрелка вниз** ) *после первого использования* **Найти | Установить текст Искать** или **Инструмент-Var**  ), **ИЛИ АЛЬТЕРНАТИВНО** в *перемещаться по стек вызовов* в **Панель с кодом** ( **Инструмент-Var** кнопки  а также  или сочетания клавиш **стрелка вверх** а также **стрелка вниз** ). Ключи **вниз на страницу** а также **вверх на страницу** перейти к выбору следующего / предыдущего функциональный модуль ..

## Настройки



**Конфигурировать | Настройки** позволяет пользователям установить программу и предпочтения просмотра (которые пользователь обычно желает принять на он следующий сеанс). Поэтому их можно сохранить и загрузить из 'myArduPrefs.txt' файл, который находится в том же каталоге, что и загруженный 'Uno' или 'Mega' программа ( 'myArduPrefs.txt' автоматически загружается, если он существует).

Это диалоговое окно позволяет выбирать между двумя моноширинными шрифтами и тремя размерами шрифта, а также другими разными предпочтениями. Начиная с версии 2.0, выбор языка теперь включен. - это всегда включают английский (**ан**), плюс один или два других

языка локали пользователя (где они существуют), и одно переопределение на основе двухбуквенного кода языка ISO-639 на самой первой линии из 'myArduPrefs.txt' файл (если таковой имеется). Выбор появляется только если перевод ".qm" файл существует в папке переводов (внутри домашний каталог UnoArduSim.exe).

## Изменить/Просмотреть

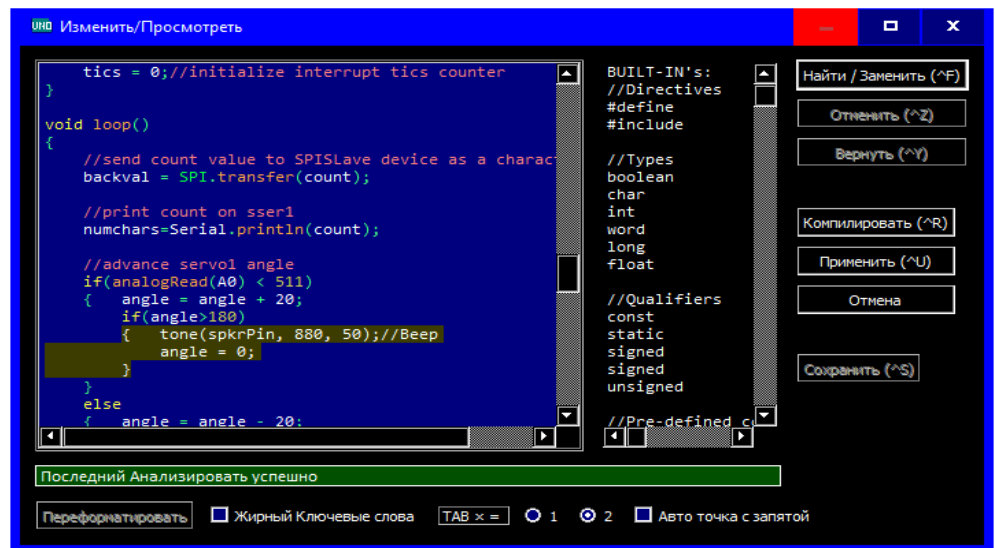
Двойным щелчком по любой строке в **Панель с кодом** (или используя меню **Файл** ), **Изменить/Просмотреть** окно открывается для внесения изменений в ваш программа файл - он открывается с **выбранная строка** в **Панель с кодом** подсвечен.

Этот окно имеет возможность полного редактирования с динамической подсветкой синтаксиса (различные цвета основной момент используются для ключевых слов C ++, комментариев и т. Д.).

Существует дополнительная подсветка синтаксиса жирный и автоматическое форматирование на уровне отступа (при условии, что вы выбрали это с помощью **Конфигурировать | Настройки** ). Вы также можете удобно выбрать встроенный функциональный модуль звонки (или встроенный '#define' константы) для

добавления в ваш программа из предоставленного списка - просто дважды щелкните по нужному элементу списка, чтобы добавить его к вашему программа в текущей позиции каретки (функциональный модуль - вызов переменной *типы* только для информации и лишены возможности оставлять фиктивные заполнители при добавлении в ваш программа).

окно имеет **Найти** (использование **Ctrl-F**) а также **Найти / Заменить** возможность (использовать **Ctrl-H** ), **Изменить/Просмотреть** окно имеет **Отменить** ( **Ctrl-Z** ), а также **Вернуть** ( **Ctrl-Y**) кнопки (которые появляются автоматически).





**Используйте ALT-стрелка вправо** запросить варианты автозаполнения для встроенный **глобальный переменные**, и для **член переменные и функциональные модули**.

Отказаться **все изменения** Вы сделали, так как вы впервые открыли программа для редактирования, нажмите **Отмена** кнопка. Принять текущее состояние, нажмите **Применить** Кнопка и программа автоматически получает другой Анализировать (и загружается в 'Uno' или 'Mega', если ошибок не обнаружено), и в главном UnoArduSim окно появляется новый статус **Статус бар**,

**Компилировать ( Ctrl-R )** кнопка (плюс связанный **Анализировать Статус** Окно сообщения, как показано на рисунке выше) было добавлено, чтобы позволить тестирование изменений без необходимости сначала закрывать окно. **Сохранить ( Ctrl-S )** также была добавлена в качестве ярлыка (эквивалент **Применить** плюс позже отдельный **Сохранить** от основной окно).

Либо на **Отмена** или **Применить** без внесенных изменений **Панель с кодом** текущая строка меняется, чтобы стать **последняя позиция каретки Изменить/Просмотреть** и вы можете использовать эту функцию, чтобы перейти **Панель с кодом** к определенной линии (возможно, чтобы подготовиться к **Выполнить До**), Вы также можете использовать **Ctrl-PgDn** а также **Ctrl-PgUp** чтобы перейти к следующему (или предыдущему) разрыву пустой строки в программа - это полезно для быстрой навигации вверх или вниз по значимым местам (например, пустые строки между функциональные модули). Вы также можете использовать **Ctrl-Home** а также **Ctrl-End** перейти к началу и концу программа, соответственно.

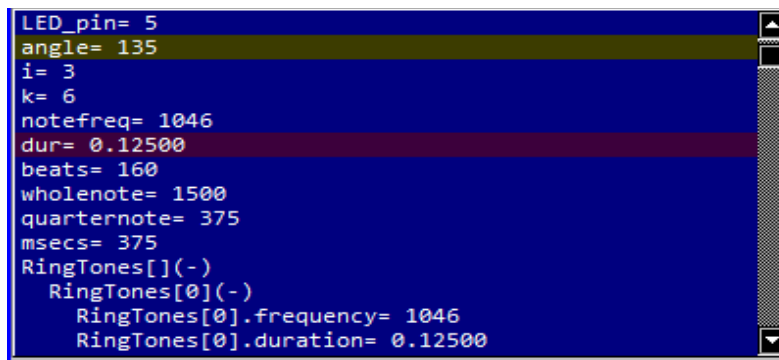
**Автоматическое форматирование отступа на уровне 'Tab'** выполняется при открытии окно, если эта опция была установлена в **Конфигурировать | Настройки**, Вы можете повторить это форматирование в любое время, нажав кнопку "Переформатировать" (она включается, только если вы ранее выбрали **Автоматическая установка отступов**). Вы также можете добавлять или удалять вкладки в группу предварительно выбранных последовательных строк с помощью клавиатуры **правая стрелка** или **стрелка влево** ключи - но **предпочтение должно быть отключено** чтобы не потерять свои собственные уровни вкладок.

когда **Авто точка с запятой** проверяется нажатием **Войти** для завершения строки автоматически вставляется точка с запятой.

И чтобы помочь вам лучше отслеживать ваши контексты и изогнутые скобки, нажав на ' { ' или ' } ' изогнутая скобка **выделяет весь текст между этим изогнутой скобкой и его соответствующим партнером**,



## **Панель с переменными и Изменить/Отслеживать Переменная окно**

**Панель с переменными** расположен чуть ниже **Панель с кодом**, Он показывает текущие значения для каждого пользователя глобального и активного (in-сфера) локального переменная / массив / объект в загруженной программа. Когда ваш программа выполнение перемещается между функциональные модули, **содержимое изменяется, чтобы отразить только те локальные переменные, которые доступны для текущего функциональный модуль / сфера**, плюс любые объявленные пользователем глобальные переменные, Любой переменные объявлен как 'const' или как 'PROGMEM' (выделено для 'Flash' память) имеют значения, которые не могут быть изменены, и поэтому для экономии места они **не отображаются**, 'Servo' а также 'SoftwareSerial' Экземпляры объект не содержат полезных значений, так же как и не отображаются



```
LED_pin= 5
angle= 135
i= 3
k= 6
notefreq= 1046
dur= 0.12500
beats= 160
wholenote= 1500
quarternote= 375
msecs= 375
RingTones[0](-)
  RingTones[0](-)
    RingTones[0].frequency= 1046
    RingTones[0].duration= 0.12500
```

Вы можете **найти** указанный **текст** с **Найти**

команды текстового поиска меню (с **Инструмент-Var** кнопки  **text** а также  **text** или сочетания клавиш **стрелка вверх** а также **стрелка вниз**), после первого использования **Найти | Комплект Искать** текст или ,

**Массивы** а также **объектов** показаны либо в **ун-расширенный** или **расширенный** формат, либо с завершающим плюсом ' (+ ) ' или минус ' ( - ) ' подписать соответственно. Символ для массив **Икс** показывает как ' **x** [ ] ' , Чтобы расширить это показать все элементы массив, просто нажмите один раз на

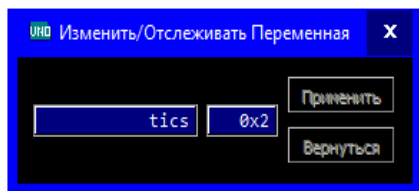


'x[] (+)' в **Панель с переменными**, Чтобы сокращаться вернуться к представлению un-расширенный, нажмите на 'x[] (-)', По умолчанию un-расширенный для объект 'p1' показывает как 'p1 (+)' К расширять это показать всем членам этого 'class' или 'struct' Например, нажмите один раз на 'p1 (+)' в **Панель с переменными**, Чтобы сокращаться вернуться к представлению un-расширенный, нажмите один раз на 'p1 (-)',

если ты **одним щелчком мыши на любой строке основной момент в темно-оливковом цвете** (это может быть простой переменная или совокупный '(+)' или '(-)' линия массив или объект, или один элемент массив или член объект), затем делать **Выполнить Пока** заставит выполнение возобновить и заморозить на следующем **записи доступа** где-нибудь внутри этого выбранного агрегата, или к тому выбранному единственному местоположению переменная.

Когда используешь **Шаг** или **Выполнить** Обновление отображаемых значений переменная производится в соответствии с настройками пользователя, выполненными в меню. **ПеремОбновить** - это обеспечивает полный диапазон поведения от минимальных периодических обновлений до полных немедленных обновлений. Уменьшенные или минимальные обновления полезны для снижения нагрузки на процессор и могут быть необходимы, чтобы выполнение не отставал в реальном времени от того, что в противном случае было бы чрезмерным **Панель с переменными** Обновление окно загружает. когда **Анимация** действует или если **Основной момент Изменения** опция меню, изменяется на значение переменная во время **Выполнить** приведет к обновлению отображаемого значения **немедленно** и он будет выделен фиолетовым цветом - это вызовет **Панель с переменными** прокрутить (при необходимости) до строки, которая содержит, что переменная, и выполнение больше не будет в режиме реального времени !.

**Когда выполнение замерзает** после **Шаг**, **Выполнить До**, **Выполнить Пока**, или **Выполнить** -тогда-**Приостановить**, **Панель с переменными** Основные моменты в **фиолетовом** переменная, соответствующий **адрес (а) адреса, которые были изменены** (если есть) **самая последняя инструкция** в течение этого выполнение (включая инициализацию объявления переменная). Если эта инструкция **полностью** заполнены **объект или массив**, **родительская (+) или (-) строка** для этого агрегата становится выделенным. Если вместо этого инструкция изменила расположение то, что в данный момент видно, затем становится подсвеченным. Но если измененное местоположение (я) в настоящее время скрывается внутри UN-расширенный массив или объект, которые в совокупности **родительская линия** получает **выделение курсивом** как визуальный сигнал, что что-то внутри этого было записано - щелкнув по расширять, оно затем вызовет его **прошлой** измененный элемент или член, чтобы стать выделенным.



**Изменить/Отслеживать** окно дает вам **способность следовать любому значению переменная в течение выполнение** или **изменить его значение в середине (остановлено) программа выполнение** (так что вы можете проверить, каков будет эффект продолжения работы с этим новым значением). **Приостановить** Сначала выполнение, потом **левый двойной щелчок** на переменная, значение которого вы хотите отслеживать или изменить. Чтобы просто контролировать значение во время программа

выполнение, **оставить диалоговое окно открытым** а затем один из **Выполнить** или **Шаг** команды - его значение будет обновлено в **Изменить/Отслеживать** в соответствии с теми же правилами, которые регулируют обновления в **Панель с переменными**, **Чтобы изменить значение переменная**, заполните значение поля ввода и **Применить**, Продолжайте выполнение (используя любой из **Шаг** или **Выполнить** команды), чтобы использовать это новое значение с этого момента вперед (или вы можете **Вернуться** к предыдущему значению).

**На программа Загрузить или Сброс** обратите внимание, что все **неинициализированное значение** переменные сбрасывается в значение 0, а все **неинициализированный указатель** переменные сбрасывается в 0x0000.

## Лабораторная панель

Лабораторная панель показывает 5-вольтовый 'Uno' или 'Mega' плата, который окружен набором 'I/O' Устройства, который вы можете выбрать / настроить и подключить к желаемому 'Uno' или 'Mega' пинах.

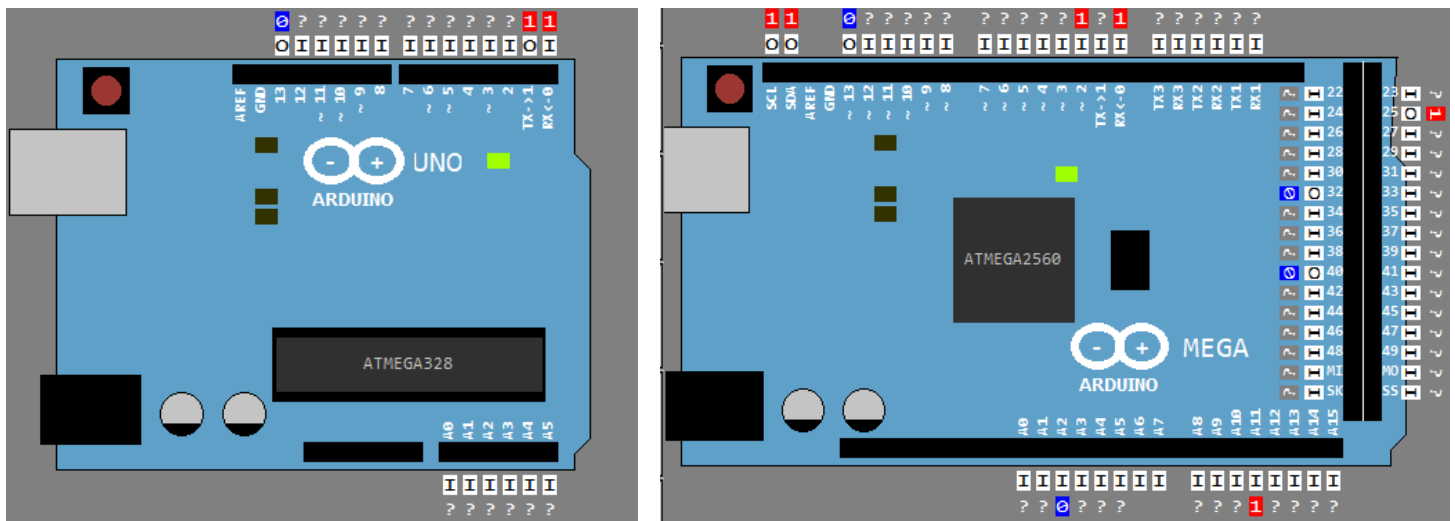
### 'Uno' или 'Mega'

Это изображение 'Uno' или 'Mega' плата и его встроенных светодиодов. Когда вы загружаете новый программа

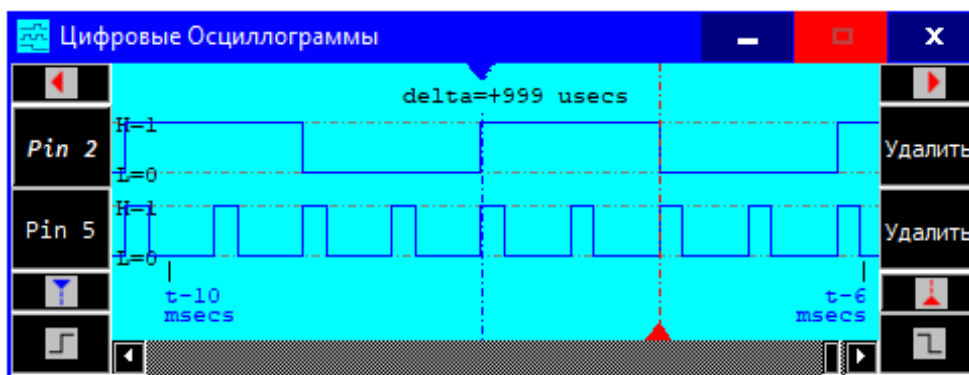
в UnoArduSim, если он успешно анализирует, он подвергается "моделированию загрузки" в плата, который имитирует способ фактического плата ведет себя - вы увидите, что серийные RX и TX LED мигают (наряду с активностью на пинах 1 и 0, которые *встроенный для последовательной связи с главным компьютером*). За ним сразу следует вспышка пин 13 LED, которая обозначает сброс плата и (и автоматическую остановку UnoArduSim в) начало загруженной программа выполнение. Вы можете избежать этого отображения и связанной с ним задержки загрузки, отменив выбор **Показать Загрузка** от **Конфигурировать | Настройки**,

окно позволяет визуализировать логические уровни цифровой на всех 20 'Uno' пинах или на всех 70 'Mega' пинах ( '1' на красный для 'HIGH' , '0' на синем для 'LOW' , а также '?' на сером для неопределенного неопределенного напряжения) и направления запрограммированный ( 'I' за 'INPUT' , или 'O' за 'OUTPUT' ). Для пинах, которые пульсируют с использованием ШИМ через 'analogWrite()' или 'tone()' или 'Servo.write()' , цвет меняется на фиолетовый и отображаемый символ становится '^' ,







Обратите внимание, что **Цифровой пинах 0 и 1 подключены через резисторы 1 кОм к микросхеме USB для последовательная связь с хост-компьютером.**





**Левой кнопкой мыши** на любом 'Uno' или 'Mega' пин откроет **Pin Цифровые Осциллограммы** окно, который отображает прошлое **стоимость одной секунды** из **Активность уровня цифровой** на этом пин. Вы можете нажать на другой пинах, чтобы добавить их на дисплей Pin Цифровые Осциллограммы (до 4 сигналов одновременно).



Нажмите для просмотра страницы влево или вправо или используйте клавиши Home, PgUp, PgDn, End

Один из отображаемых сигналов будет **активный пин** осциллограмма обозначается кнопкой "Пин", обозначаемой как нажата (как на снимке экрана выше Pin Цифровые Осциллограммы). Вы можете выбрать осциллограмма, нажав на его цифровую кнопку Пин, а затем выбрать интересующую полярность края, нажав соответствующую кнопку выбора полярности восходящего / падающего края,  , или  или с помощью сочетаний клавиш **стрелка вверх** а также **стрелка вниз** , Вы можете тогда **Прыгать** активный курсор (синие или красные линии курсора с указанным их дельта-временем) назад или вперед к краю выбранной полярности цифровой **этого активного пин** осциллограмма с помощью кнопок курсора,  ,  или  ,  (в зависимости

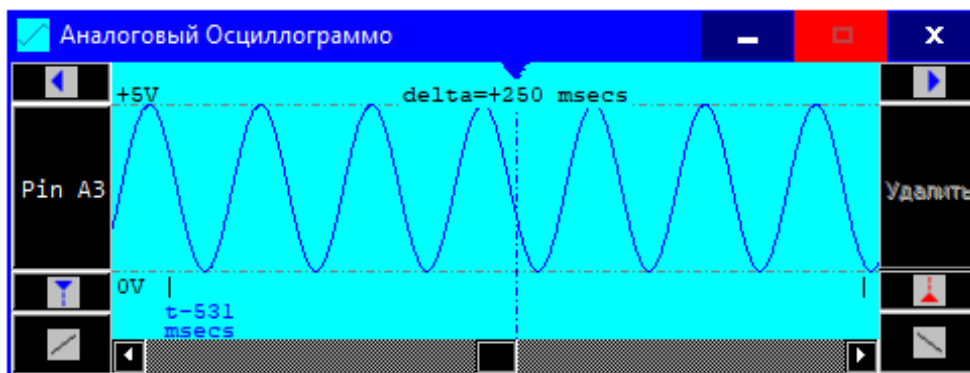
от того, какой курсор был активирован ранее с  или ), или просто используйте клавиши клавиатуры ← а также → ,

Чтобы активировать курсор, нажмите его цветную кнопку активации (  или  показано выше) - *это также прокручивает представление к текущему местоположению этот курсор* , Кроме того, вы можете быстро чередовать активацию между курсорами (с их соответственно центрированными видами), используя ярлык 'Tab' ключ.







Вы можете **Прыгать** текущий активированный курсор **щелкнув левой кнопкой мыши в любом месте** в области просмотра осциллограмма на экране. Кроме того, вы можете выбрать красную или синюю линию курсора, щелкнув справа вверху (чтобы активировать), затем **перетащите его в новое место** и отпустите. Когда нужный курсор находится где-то вне экрана, вы можете **щелкните правой кнопкой мыши в любом месте** чтобы перейти к этому новому местоположению на экране. Если оба курсора уже находятся на экране, щелчок правой кнопкой мыши просто чередует активированный курсор.

**Чтобы ZOOM IN и ZOOM OUT (масштаб всегда центрирован на АКТИВНОМ курсоре), используйте колесико мыши или сочетания клавиш CTRL-стрелка вверх и CTRL-стрелка вниз,**

Делать вместо **щелкните правой кнопкой мыши на любом 'Uno' или 'Mega' пин** открывает Pin Аналоговый Осциллограммо окно, который отображает **за одну секунду стоит** из **Активность на уровне аналоговый** на этом пин. В отличие от Pin Цифровые Осциллограммы окно, вы можете отображать активность аналоговый только на одном пин одновременно.



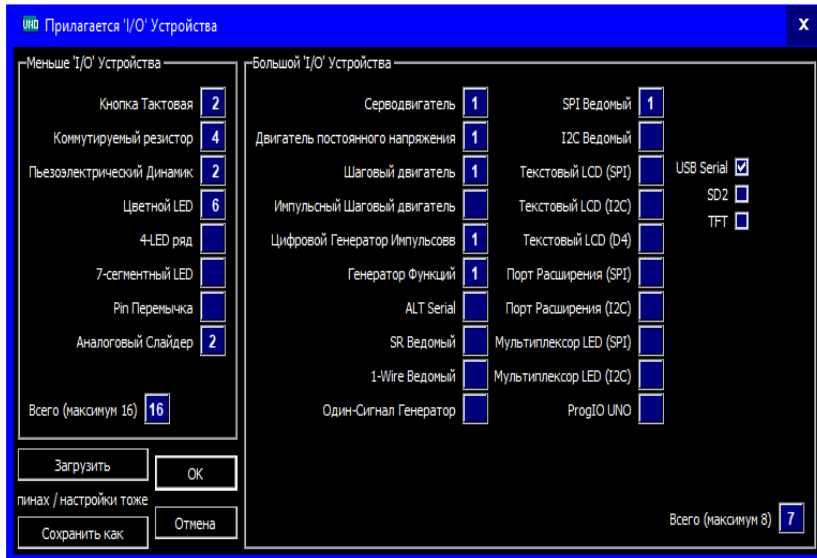
Нажмите для просмотра страницы влево или вправо или используйте клавиши Home, PgUp, PgDn, End

Вы можете **Прыгать** синие или красные линии курсора к следующей поднимающейся или опускающейся "точке наклона" с помощью кнопок со стрелками вперед или назад (  ,  или  ,  , снова в зависимости от активированного курсора, или используйте ← а также → клавиш) в сочетании с кнопками выбора подъема / спада  ,  ("Точка наклона" возникает, когда напряжение аналоговый проходит через высокий порог логического уровня Kmega пин ATmega). Кроме того, вы можете снова щелкнуть, чтобы прыгать, или перетащить эти линии курсора, аналогичные их поведению в Pin Цифровые Осциллограммы окно

прессование 'Ctrl-S' внутри **любой окно позволяет сохранить осциллограмма (X, Y) данные** на текст файл по вашему выбору, где **Икс** в микросекундах с левой стороны, и **Y** в болтах.

## 'I/O' Устройства

Ряд различных Устройства окружают 'Uno' или 'Mega' плата по периметру **Лабораторная панель**, "Малый" 'I/O' Устройства (из которых вам разрешено до 16) расположены вдоль левой и правой сторон Панель. "Большой" 'I/O' Устройства (вам разрешено столько, сколько поместится) имеет "активные" элементы и располагается вдоль верхней и нижней части **Лабораторная панель**, Желаемое количество каждого типа доступных 'I/O' устройство можно установить с помощью меню **Конфигурировать | 'I/O' Устройства**,



Каждый 'I/O' устройство имеет одно или несколько вложений пин, отображаемых как **два-цифра** Номер пин (00, 01, 02,... 10,11,12, 13 и либо A0-A5, или 14-19, после этого) в соответствующем поле ввода. Для номеров пин со 2 по 9 вы можете просто ввести один цифра - ведущие 0 будут предоставлены автоматически, но для пинах 0 и 1 вы должны сначала ввести ведущие 0. Входы *обычно* на левой стороне 'I/O' устройство, и выходы *обычно* справа (если позволяют). Все 'I/O' Устройства будут реагировать непосредственно на уровни пин и изменения уровня пин, поэтому будут реагировать либо на библиотеку функциональные модули, нацеленную на подключенный к ней пинах, либо на запрограммированный

'digitalWrite()' (для "битовой" операции).

Вы можете подключить несколько Устройства к одному ATmega пин, если это не создает **электрический конфликт**, Такой конфликт может быть создан или 'Uno' или 'Mega' пин как **output** вождение с подключением устройство с сильной проводимостью (низким сопротивлением) (например, с выходом 'PUSH' или 'MOTOR' **Enc** выход) или двумя подключенными Устройства, конкурирующими друг с другом (например, и кнопка 'PULSER' и 'PUSH' -, прикрепленная к одному и тому же пин). Любой такой конфликт будет иметь катастрофические последствия в реальной аппаратной реализации и поэтому будет запрещен, и будет отмечен для пользователя через всплывающее окно сообщения).

Диалоговое окно может использоваться, чтобы позволить пользователю выбирать типы и номера желаемого 'I/O' Устройства. Из этого диалогового окна вы также можете **Сохранить** 'I/O' Устройства для текста файл и / или **Загрузить** 'I/O' Устройства из ранее сохраненного (или отредактированного) текста файл (**включая все соединения пин, интерактивные настройки и любые введенные значения в поле ввода**).

**Обратите внимание, что значения в полях редактирования периода, задержки и ширины импульса в соответствующем IO Устройства могут иметь суффикс 'S' (или 's')**. Это указывает на то, что они должны быть **масштабируются** в соответствии с позицией глобального **'IO \_\_\_\_S'** ползунков, который появляется в главном окне **Инструмент-Bar**, С этим ползунком вправо масштабный коэффициент равен 1,0 (единица), и с ползунком влево масштабный коэффициент равен 0,0 (с учетом минимальных значений, установленных каждым конкретным 'I/O' устройство). Вы можете масштабировать более одного значения поля ввода **одновременно** используя этот слайдер. Эта функция позволяет перетаскивать ползунок во время выполнения легко эмулировать изменение длительности импульсов, периодов и задержек для подключенных 'I/O' Устройства.

В оставшейся части этого раздела приведены описания для каждого типа устройство.

Несколько из этих Устройства **поддержка масштабирования их введенных значений** с помощью ползунка на основной окно **Инструмент-Bar**, Если значение устройство имеет букву 'S' в качестве суффикса, его значение будет умножено на коэффициент масштабирования (от 0,0 до 1,0), который определяется положением ползунка при условии ограничения минимального значения устройство (1.0 полностью вправо, 0.0 полностью влево) - смотрите **'IO \_\_\_\_S'** под каждым шлангом Устройства подробно описан ниже.



## Serial Монитор ('SERIAL')

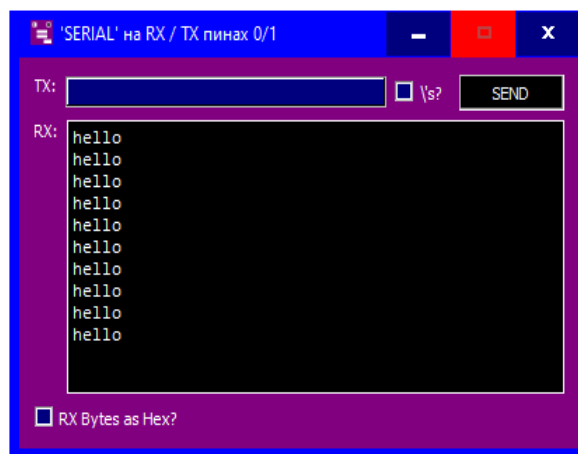


Этот 'I/O' устройство допускает аппаратно-опосредованный последовательный ввод и вывод ATmega (через USB-чип 'Uno' или 'Mega') на пинах 0 и 1. скорость передачи устанавливается с помощью раскрывающегося списка внизу - выбранный скорость передачи **должен соответствовать** значению, которое ваш программа передает 'Serial.begin()' функциональный модуль для правильной передачи / приема.

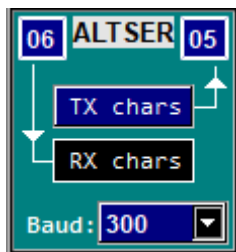
Последовательная связь фиксируется на 8 битах данных, 1 стоповом бите и без битов четности. Вам разрешено **Отключить** (Пусто) **но не заменить** TX пин 00, но не RX пин 01.

Чтобы отправить ввод с клавиатуры на программа, введите один или несколько символов в верхнем (символы TX), отредактируйте окно, а затем ударь '**Enter**' клавиша на клавиатуре, (символы выделяются курсивом, чтобы указать, что передачи начались) - или, если они уже выполняются, добавленные печатные символы будут выделены курсивом. Затем вы можете использовать 'Serial.available()' а также 'Serial.read()' функциональные модули для чтения символов в порядке их поступления в буфер пин 0 (крайний левый набранный символ будет отправлен первым). Отформатированные текстовые и числовые распечатки или неформатированные байтовые значения могут быть отправлены на нижний выход консоли (RX-символы) окно, вызвав Arduino. 'print()', 'println()', или 'write()' функциональные модули.

Дополнительно, **окно большего размера для установки / просмотра символов TX и RX можно открыть, дважды щелкнув (или щелкнув правой кнопкой мыши) на этом 'SERIAL' устройство**, Этот новый окно имеет увеличенное поле редактирования символов TX и отдельную кнопку 'Send', которую можно щелкнуть, чтобы отправить символы TX на 'Uno' или 'Mega' (на пин 0). Существует также опция флажка для повторной интерпретации последовательностей символов с обратной косой чертой, таких как '\n' или '\t' для необработанного отображения.



## чередовать Серийный ('ALTSER')

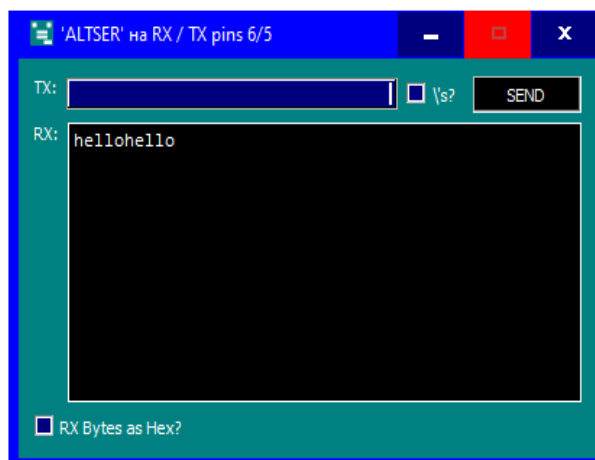


Этот 'I/O' устройство допускает библиотеку или, альтернативно, пользовательский "битовый разряд", последовательный ввод и вывод на любую пару 'Uno' или 'Mega' пинах, которую вы выбираете заполнить (**кроме** пинах 0 и 1, которые предназначены для аппаратного обеспечения 'Serial' связи). Ваш программа должен иметь '#include <SoftwareSerial.h>' строка рядом с верхом, если вы хотите использовать функциональные возможности этой библиотеки. Как и в случае 'SERIAL' устройство, скорость передачи для 'ALTSER' устанавливается с помощью раскрывающегося списка внизу - выбранный скорость передачи должен соответствовать значению, которое программа передает в 'begin()' функциональный модуль для

правильной передачи / приема. Последовательная связь фиксируется на 8 битах данных, 1 стоповом бите и без битов четности.

Также как с 'SERIAL', **окно для настройки и просмотра TX и RX можно открыть, дважды щелкнув (или щелкнув правой кнопкой мыши) на ALTSER устройство**,

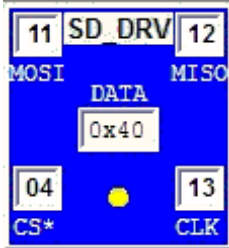
Обратите внимание, что в отличие от аппаратной реализации 'Serial', в 'SoftwareSerial' нет предоставленного TX-буфер поддерживается внутренними операциями прерывания ATmega (только RX-буфер), поэтому который 'write()' (или 'print') вызовы блокируются (т. е. ваш программа не будет продолжаться, пока они не будут завершены).





## Карта памяти SD ('SD\_DRV')

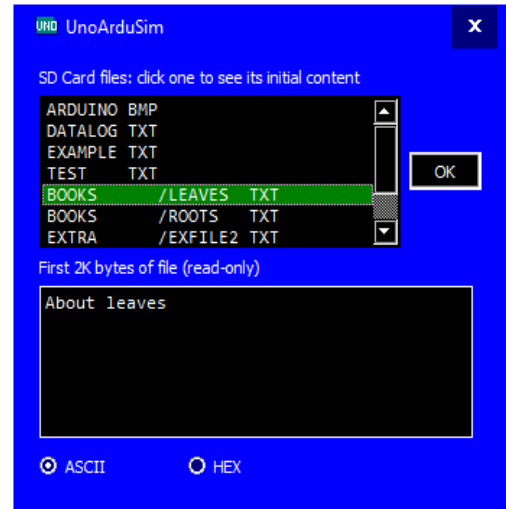
Этот 'I/O' устройство позволяет использовать библиотеку с программным обеспечением (но **не** "битовый разряд") Операции ввода и вывода файл на 'Uno' или 'Mega' **SPI** пинах (вы можете выбрать, какой **CS \*** пин вы будете использовать). Ваш программа может просто `#include <SD.h>` линия около вершины, и вы можете использовать `<SD.h>` функциональные модули ИЛИ позвоните напрямую `SdFile` функциональные модули самостоятельно.



*окно большего размера с каталогами и файлы (и содержимым) можно открыть, дважды щелкнув (или щелкнув правой кнопкой мыши) на 'SD\_DRV' устройство. , Все содержимое диска **загружен из SD** подкаталог в загруженном каталоге программа (если он существует) в `SdVolume::init()` , **и отражается в** то же самое **SD** подкаталог*

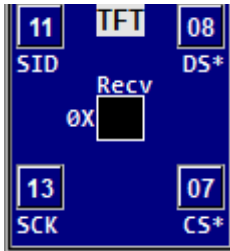
на файл `'close()'` , `'remove()'` и на `'makeDir()'` а также `'rmDir()'` ,

Желтый LED мигает во время передачи SPI, и 'DATA' показывает последний 'SD\_DRV' **ответ** байт. Все сигналы SPI точны и могут быть просмотрены в **Осциллограмма окно**,



## TFT Дисплей ('TFT')

Этот 'I/O' устройство подражает Adafruit™ TFT-дисплей размером 1280–160 пикселей (при его собственном повороте = 0, но при использовании `'TFT.h'` библиотека, `'TFT.begin()'` наборы инициализации для поворота = 1, что дает "пейзажное" представление размером 160 на 128 пикселей). Вы можете толкнуть это устройство, позвонив в функциональные модули `'TFT.h'` библиотека (которая сначала требует `#include <TFT.h>`), или вы можете использовать систему SPI для отправки собственной последовательности байтов на нее толкнул.

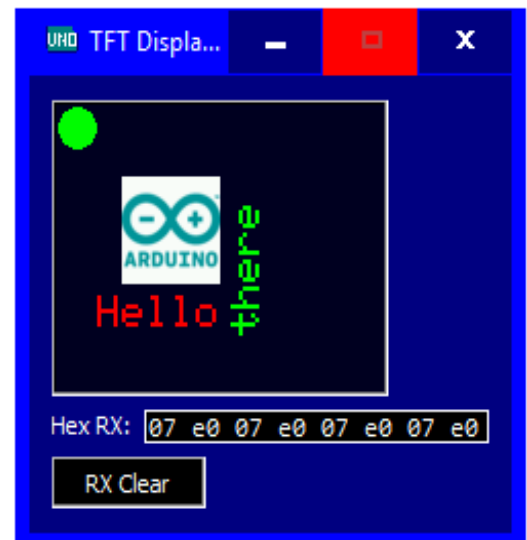


TFT всегда подключен 'SPI' пинах 'MOSI' (для 'SID') и 'SCK' (для 'SCK') - их нельзя изменить. 'DS\*' пин предназначен для выбор данных / команды ('LOW' выбирает режим данных) и 'CS\*' пин активный-низкий выбор чипа

Сброс пин не предоставляется, поэтому вы не можете выполнить аппаратный сброс это устройство, ведя низкий пин (как `'TFT::begin()'` функциональный модуль пытается сделать, когда вы передали действительный номер 'reset' пин в качестве третьего параметра к `'TFT(int cs, int ds, int rst)'` конструктор). устройство, однако,

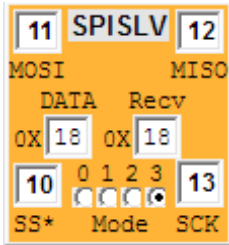
имеет скрытое соединение с системной линией Сброс, поэтому он сбрасывает его; f каждый раз, когда вы щелкаете по основному UnoArduSim Значок панели инструментов Сброс или кнопка сброса KL77 'Uno' или 'Mega'.

По **двойной щелчок** (или **щелкнув правой кнопкой мыши**) на этом устройстве открывается более крупный окно, чтобы показать полный ЖК-дисплей размером 160 на 128 пикселей вместе с последними полученными 8 байтами (как показано ниже)



## Конфигурируемый SPI Ведомый ('SPISLV')

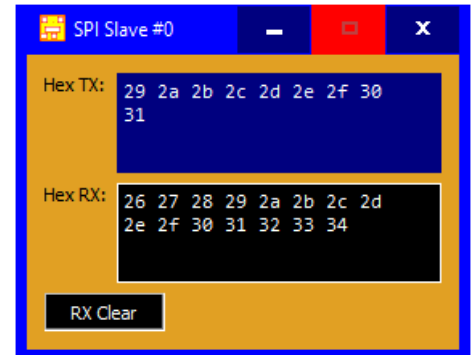
Этот 'I/O' устройство эмулирует ведомый SPI выбранного режима с активным низким **SS \*** ("выбор ведомого") пин управляет **MISO** выход пин (когда **SS \*** в приоритете, **MISO** это не толкнул). Ваш программа должен иметь `'#include <SPI.h>'` линии, если вы хотите использовать функциональность встроенный SPI Arduino объект и библиотеки. В качестве альтернативы, вы можете создать свой собственный "битовый удар" **MOSI** а также **SCK** сигналы к толкнул это устройство.



устройство ощущает краевые переходы на своем **CLK** ввод в соответствии с выбранным режимом ('MODE0', 'MODE1', 'MODE2', или 'MODE3'), который должен быть выбран в соответствии с режимом SPI запрограммированный вашего программа.

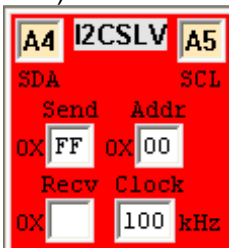
**Двойной щелчок (или щелчок правой кнопкой мыши) на устройство позволяет открыть**

**более крупный спутник окно** что вместо позволяет вам Вы должны заполнить 32-байтовый максимальный буфер (чтобы эмулировать SPI Устройства, который автоматически возвращает свои данные), и увидеть последние 32 полученных байта (все как шестнадцатеричные пары). Обратите внимание, что следующий байт буфера TX автоматически отправляется только на 'DATA' после полный '`SPI.transfer()`' завершено!



## Двухпроводная I2C Ведомый ('I2CSLV')

Этот 'I/O' устройство только эмулирует *ведомый режим* устройство. устройство может быть назначен адрес шины I2C, используя запись в два шестнадцатеричных цифра в поле ввода 'Addr' (он будет отвечать только на I2C автобусные операции с использованием назначенного ему адреса). устройство отправляет и получает данные на своем открытом канале (только для раскрывающегося списка) **SDA** пин, и отвечает на сигнал тактовой частоты шины на своем открытом канале (только для раскрывающегося списка) **SCL** пин. Хотя 'Uno' или 'Mega' будут хозяином шины, ответственным за генерацию **SCL** сигнал, этот раб устройство также будет тянуть **SCL** низкий во время своей низкой фазы, чтобы увеличить (если это необходимо) низкое время шины до значения, соответствующего ее внутренней скорости (которую можно установить в поле редактирования 'Clock').

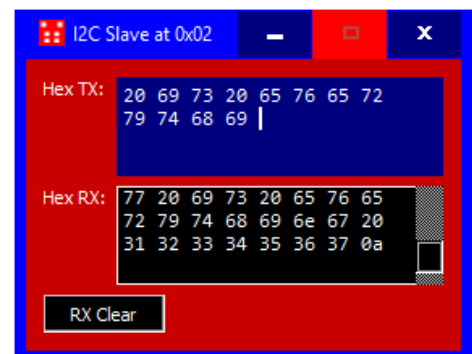


Ваш программа должен иметь `'#include <Wire.h>'` линии, если вы хотите использовать функциональность '`TwoWire`' библиотека для взаимодействия с этим устройство. В качестве альтернативы, вы можете создать свои собственные битовые данные и тактовые сигналы для толкнул этого ведомого устройство.

Один байт для передачи обратно на ведущее устройство 'Uno' или 'Mega' может быть установлен в поле редактирования 'Send', а один (последний принятый) байт может быть просмотрен в его (только для чтения) окне редактирования 'Recv'. Обратите внимание, что значение поля редактирования 'Send' всегда отражает следующий байт

для передачи из этого внутреннего буфера данных устройство.

**Двойной щелчок (или щелчок правой кнопкой мыши) на устройство позволяет открыть более крупный спутник окно** вместо этого это позволяет вам заполнить буфер FIFO с максимальным размером 32 байта (чтобы эмулировать TWI Устройства с такой функциональностью) и просматривать (максимум до 32) байтов самых последних полученных данных (как два hex-цифра отображает 8 байтов на строку). Количество строк в этих двух полях редактирования соответствует выбранному размеру буфера TWI (который можно выбрать с помощью **Конфигурировать | Настройки**). Это было добавлено в качестве опции, так как Arduino '`Wire.h`' библиотека использует **пять** такие буферы ОЗУ в своем коде реализации, который является дорогой оперативной памяти. Редактируя установку Arduino '`Wire.h`' файл для изменения определенной константы '`BUFFER_LENGTH`' (а также редактирование компаньона '`utility/twi.h`' файл для изменения длины буфера TWI) оба вместо 16 или 8, пользователь *мог* значительно сократить объем оперативной памяти 'Uno' или 'Mega' в целевых **аппаратная реализация** - поэтому UnoArduSim отражает эту реальную возможность через **Конфигурировать | Настройки**,

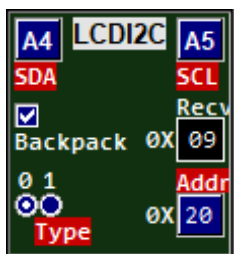




## Текстовый LCD I2C ('LCDI2C')

Это 'I/O' устройство эмулирует 1, 4, 4 строки Символьный ЖК-дисплей, в одном из трех режимов:  
а) рюкзак тип 0 (расширитель портов в стиле Adafruit с оборудованием, имеющим адрес шины I2C 0x20-0x27)  
б) рюкзак типа 1 (расширитель порта в стиле DFRobot с Адрес шины I2C 0x20-0x27)  
с) нет рюкзака (встроенный интерфейс I2C в основном режиме с адресом шины I2C 0x3C-0x3F)

Поддержка кода библиотеки для каждого режима устройство была предоставлена в 'include\_3rdParty' папка вашего установочного каталога UnoArduSIm: 'Adafruit\_LiquidCrystal.h', 'DFRobot\_LiquidCrystal.h', а также 'Native\_LiquidCrystal.h' соответственно.



устройство может быть назначен любой адрес шины I2C, используя запись в два шестнадцатеричных цифра в поле ввода 'Addr' (он будет отвечать только на I2C автобусные операции с использованием назначенного ему адреса). устройство получает адрес шины и данные (и отвечает ACK = 0 или NAK = 1) на открытом канале (только для выпадающего меню) SDA пин. Вы можете писать только команды LCD и данные DDRAM - вы **не можешь** читать данные из

записанных мест DDRAM ..



**Двойной клик** или **щелкните правой кнопкой мыши** открыть ЖК-монитор монитора окно, с которого вы также можете установить размер экрана и набор символов.

## Текстовый LCD SPI ('LCSPI')

Этот 'I/O' устройство эмулирует 1, 4, 4 линии Символьный ЖК-дисплей, в одном из двух режимов:  
а) рюкзак (Расширитель порта SPI в стиле Adafruit)  
б) нет рюкзака (встроенный режим SPI интерфейс - как shown below)

Поддержка кода библиотеки для каждого режима устройство была предоставлена в 'include\_3rdParty' папка вашего установочного каталога UnoArduSIm: 'Adafruit\_LiquidCrystal.h', а также 'Native\_LiquidCrystal.h' соответственно.

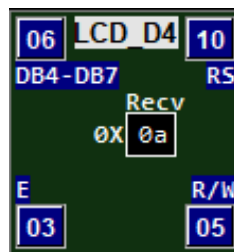


Пин 'SID' - это последовательные данные, 'SS\*' - активный низкий-устройство-выбор, 'SCK' - тактовый сигнал пин, а 'RS' - данные / команда пин. Вы можете писать только команды LCD и данные DDRAM (все транзакции SPI являются записью) - вы **не можешь** читать данные из записанных мест DDRAM.

**Двойной клик** или **щелкните правой кнопкой мыши** открыть ЖК-монитор монитора окно, с которого вы также можете установить размер экрана и набор символов.



## Текстовый LCD D4 ('LCD\_D4')



Этот 'I/O' устройство эмулирует 1, 4, 4 линии Символьный ЖК-дисплей с интерфейсом 4-битной параллельной шины. Байты данных записываются / читаются в **две половины** на его 4 данных пинах 'DB4-DB7' (где поле редактирования содержит **наименьший из 4 последовательных номеров пин**), - данные синхронизируются по задним фронтам на 'E' (активировать) пин, направление данных контролируется 'R/W' пин, а режим данных / команд на ЖК-дисплее - 'RS' пин.

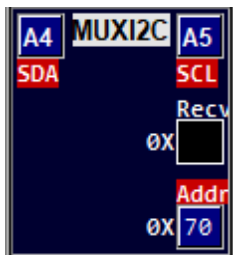
Поддерживающий библиотечный код был предоставлен внутри 'include\_3rdParty' папка вашего установочного каталога UnoArduSIm: 'Adafruit\_LiquidCrystal.h', а также 'Native\_LiquidCrystal.h' обе работают.

**Двойной клик** или **щелкните правой кнопкой мыши** открыть ЖК-монитор монитора окно, с которого вы также можете установить размер экрана и набор символов.



## Мультиплексор LED I2C ('MUXI2C')

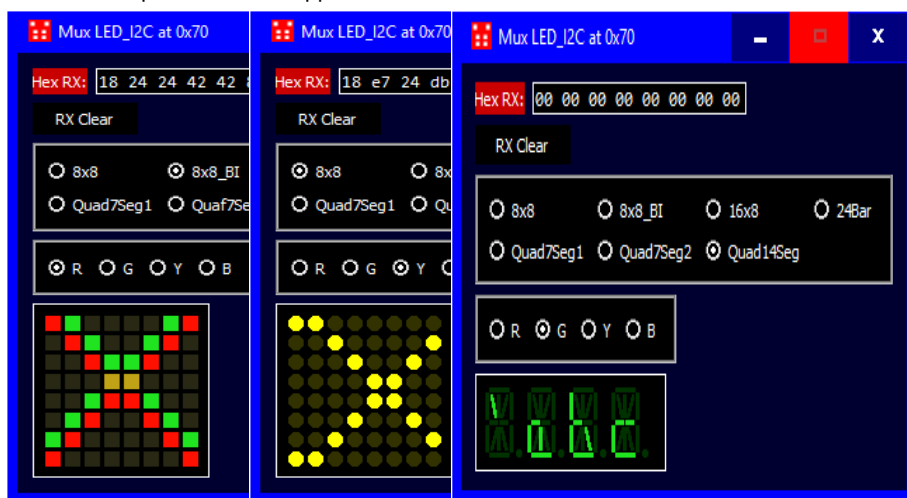
Этот 'I/O' устройство эмулирует контроллер HT16K33 с интерфейсом I2C (having I2C адрес шины 0x70-0x77) которому можно подключить один из нескольких типов мультиплексированных дисплеев LED



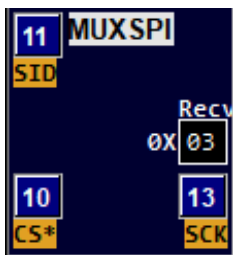
- а) 8x8 или 16x8, LED массив
- б) 8x8 двухцветный LED массив
- в) 24-цветная полоса LED
- д) два стиля 7-сегментных дисплеев 4-цифра
- е) один 4-цифра 14-сегментный буквенно-цифровой дисплей

Все поддерживаются 'Adafruit\_LEDBackpack.h' код предоставляется внутри 'include\_3rdParty' папка:

Двойной клик (или щелкните правой кнопкой мыши) открыть большой окно выбирать и просматривать один из нескольких цветных LED дисплеи.

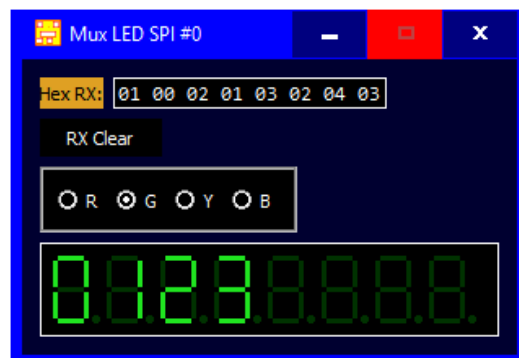


## Мультиплексор LED SPI ('MUXSPI')

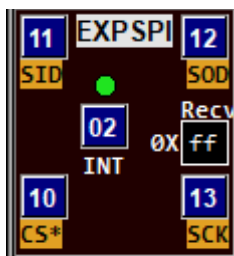


Мультиплексный контроллер LED на основе MAX6219, с поддержкой 'MAX7219.h' код предоставляется внутри 'include\_3rdParty' папка к толкнул до восьми 7-сегментных цифр.

Двойной клик (или щелкните правой кнопкой мыши) открыть большой окно т вид цветной 8-цифра 7-сегментный дисплей.



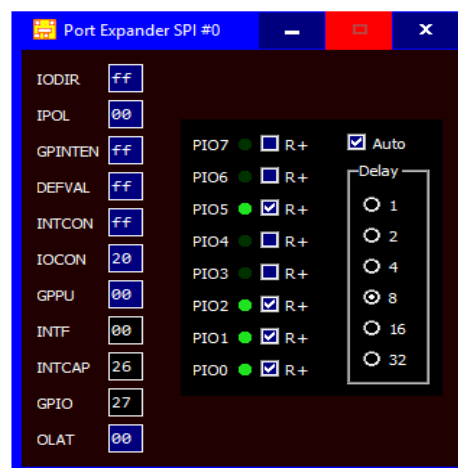
## Порт Расширения SPI ('EXPSPi')



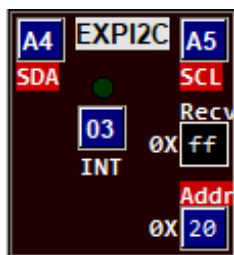
8-битный расширитель порта на основе MCP23008, с поддержкой 'MCP23008.h' код предусмотрено внутри 'include\_3rdParty' папки. Вы можете записать в регистры MCP23008 и прочитать обратно GPIO пин уровни. Прерывания могут быть включены при каждом изменении GPIO пин - сработавшее прерывание будет толкнул 'INT' пин.

Двойной клик (или щелкните правой кнопкой мыши) открыть **больше окно увидеть** 8 линий портов GPIO и подключенные подтягивающие резисторы. Вы можете изменить подтягивания

вручную, нажав или прикрепив счетчик, который будет периодически изменять их в порядке увеличения. Скорость, с которой увеличивается отсчет, определяется задержкой уменьшения коэффициент, выбранный пользователем (коэффициент 1x соответствует одному приращению приблизительно каждые 30 миллисекунд; более высокие коэффициенты задержки дают более медленную скорость увеличения)

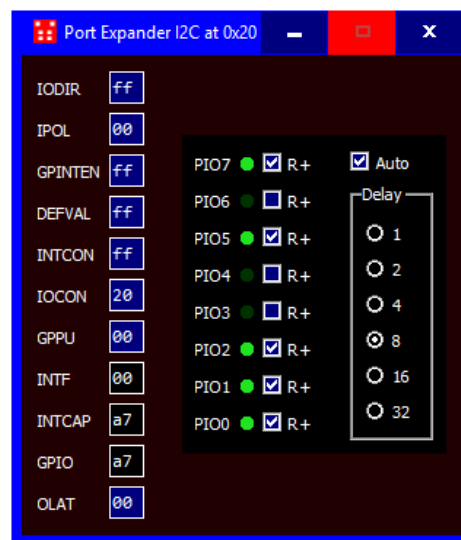


## Порт Расширения I2C ('EXPI2C')



8-битный расширитель порта на основе MCP23008, с поддержкой 'MCP23008.h' код предоставляется внутри 'include\_3rdParty' папки. Возможности соответствуют 'EXPSPi' устройство.

Двойной клик (или щелкните правой кнопкой мыши) открыть **большой окно** как на 'EXPSPi' устройство.



## '1-Wire' Ведомый ('OWISLV')

Этот 'I/O' устройство эмулирует один из небольшого набора шины '1-Wire' Устройства, подключенного к пин OWIO. Вы можете создать шину '1-Wire' (с одним или несколькими из этих ведомых '1-Wire' Устройства) на 'Uno' или 'Mega' пин по вашему выбору. Эту кабину устройство можно использовать, позвонив 'OneWire.h' библиотека функциональные модули после размещения '#include <OneWire.h>' линия в верхней части вашего программа. В качестве альтернативы вы также можете использовать побитовые сигналы на OWIO для этого устройство (хотя это очень сложно сделать правильно, не вызывая электрический конфликт - такой конфликт все еще возможен даже при использовании 'OneWire.h' функциональные модули, но такие конфликты сообщаются в UnoArduSim).



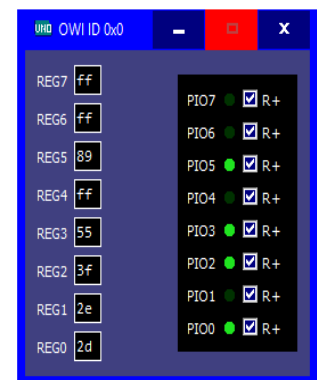
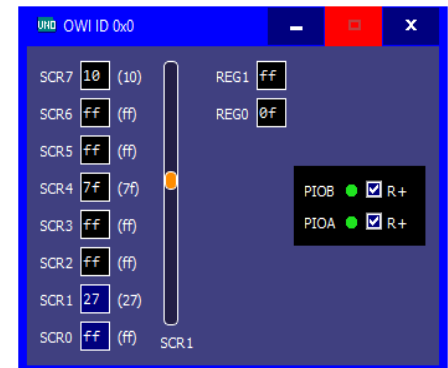
Каждый реальный OWISLV устройство должен иметь уникальный внутренний 8-байтовый i (64-битный!) Внутренний серийный номер - в UnoArduSim это упрощается тем, что пользователь предоставляет короткий 1-байтовый шестнадцатеричный 'ID' значение (которое назначается последовательно по умолчанию при загрузке / добавлении устройство), плюс 'Fam' Семейный код для этого устройство. UnoArduSim распознает небольшой набор кодов Семейства с V2.3 (0x28, 0x29, 0x3A, 0x42), охватывающий датчик температуры, и параллельный ввод-вывод (PIO) Устройство (нераспознанный код Семейства) превращает устройство в универсальный 8-байтовый блокнот устройство с

универсальным датчиком.

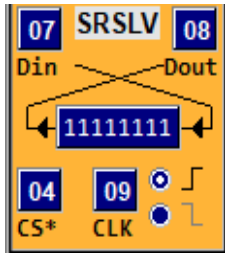
Если у семейства устройство нет регистров PIO, регистры **D0** а также **D1** представлять первые два байта блокнота, иначе они представляют PIO Регистр "status" (фактические уровни пин) и регистр данных фиксатора PIO пин, соответственно.

По **двойной щелчок** (или **щелкнув правой кнопкой мыши**) на устройство, больше **OWIMonitor** окно открыт. Из этого большего окна вы можете проверить все регистры устройство, изменить местоположения блокнота SCR0 и SCR1, используя правки и ползунок (опять же, SCR0 и SCR1 соответствуют только **D0** а также **D1** если PIO отсутствует), или установите внешние подтягивания пин PIO. Когда SCR0 и SCR1 редактируются, UnoArduSim запоминает эти отредактированные значения как пользовательское "предпочтение", представляющее начальное (начиная с Сброс) значение, представляющее значение со знаком, выводимое из устройство; датчик s - ползунок сбрасывается на 100% (масштабный коэффициент 1,0) во время редактирования. Когда ползунок впоследствии перемещается, 'signed' значение в SCR1 уменьшается в соответствии с положением ползунка (масштабный коэффициент от 1,0 до 0,0) - его функция позволяет вам легко проверить реакцию вашего программа на плавно изменяющиеся значения датчика. ,

Для устройство с PIO пинах, когда вы устанавливаете флажки уровня пин, UnoArduSim запоминает эти проверенные значения как текущие подтягивания применяются к пинах. Эти внешние значения подтягивания затем используются вместе с данными защелки пин (регистр **D1**), чтобы определить окончательные фактические уровни пин, и зажечь или погасить зеленый LED, прикрепленный к PIO пин (только пин 'HIGH' если применяется внешнее подтягивание, **а также** соответствующий **D1** защелка немного '1').



## Регистр Сдвига Ведомый ('SRSLV')



Этот 'I/O' устройство эмулирует простой регистр сдвига устройство с активным низким **SS \*** ("выбор ведомого") пин управляет **'Dout'** выход пин (когда **SS \*** в приоритете, **'Dout'** это не толкнул). Ваш программа может использовать функциональность встроенный SPI Arduino объект и библиотеки. В качестве альтернативы, вы можете создать свой собственный "битовый удар" **'Din'** а также **CLK** сигналы к толкнул это устройство.

устройство ощущает краевые переходы на своем **CLK** вход, который запускает сдвиг своего регистра - полярность воспринимается **CLK** край может быть выбран с помощью радио-кнопки управления. На каждом **CLK** край (воспринимаемой полярности), регистр захватывает его **шум** уровень в позицию младшего значащего бита (LSB) регистра сдвига, поскольку оставшиеся биты одновременно сдвигаются влево на одну позицию в направлении позиции MSB. Всякий раз, когда **SS \*** низкое, текущее значение в позиции MSB регистра сдвига толкнул на **'Dout'**,

## Программируемый 'I/O' Устройство ('PROGIO')



Этот 'I/O' устройство на самом деле является чистым 'Uno' плата, который вы можете программа (с отдельным программа), чтобы эмулировать 'I/O' устройство чье поведение вы можете полностью определить. Вы можете выбрать до четырех пинах (IO1, IO2, IO3 и IO4), которые этот подчиненный 'Uno' будет совместно использовать с ведущим (main 'Uno' или 'Mega'), который отображается в середине вашего **Лабораторная панель**. Как и в случае с другим Устройства, любой электрический конфликт между этим ведомым 'Uno' и ведущим плата будет обнаружен и помечен. Обратите внимание, что все соединения **непосредственно** проводная, **кроме пин 13** (где между двумя пинах

предполагается последовательное сопротивление R-1K, чтобы предотвратить электрический конфликт на Сброс). Начиная с V2.8, соединение между ведущим и подчиненным пинах **сопоставляются**: если ведущим также является 'Uno', по умолчанию для отображения идентификатор (за исключением того, что пин 1 отображается на пин 0 и наоборот, чтобы разрешить связь 'Serial'); если ведущим является 'Mega', пинах 1 и 0 снова переключаются, **и все SPI и TWI пинах являются mapped** для прямого соединения между соответствующей главной и подчиненной подсистемами. Это сопоставление по умолчанию можно изменить, указав **vlODEvs.txt** файл - явный список из 4 основных номеров пин, который следует за именем PROGIO программа файл - если любое из этих значений равно -1, это соответствует отображению по умолчанию. Для этого устройство набранные пин цифры **те из рабов 'Uno'**. На рисунке слева показаны 4 подчиненных пинах, указанных для его системы SPI пинах (**SS \***, **MISO**, **MOSI**, **SCK**) - независимо от того, был ли он ведущим, 'Uno' или 'Mega', это позволило бы программа этому ведомому в качестве универсальный подчиненный SPI (или ведущий), поведение которого вы можете определить программно.

По **двойной щелчок** (или **щелкнув правой кнопкой мыши**) на этом устройство открывается более крупный окно, чтобы показать, что у этого ведомого 'Uno' есть свой **Панель с кодом** а также связанный **Панель с переменными** так же, как мастер. Он также имеет свой собственный **Инструмент-бар**, который вы можете использовать для **нагрузка** а также **управление выполнение** подчиненного программа - действия пиктограмм имеют те же сочетания клавиш, что и в главном окне. (**Загрузить** является **Ctrl-L**, **Сохранить** является **Ctrl-S** так далее.). После загрузки вы можете Выполнить из **или** Основной UnoArduSim окно или изнутри этого Монитора Ведомый окно - в любом случае Main программа и Ведомый 'Uno' программа остаются заблокированными в синхронизации с прохождением в реальном времени, так как их исполнения продвигаются вперед. **Чтобы выбрать Панель с кодом, который будет иметь загрузка, поиск и фокус выполнение**, щелчок на его **родительская строка заголовка окно - нефокусированный Панель с кодом имеет свою панель инструментов действия затенены**,

Некоторые возможные идеи для ведомого Устройства, который может быть запрограммированный в этот 'PROGIO' устройство, перечислены ниже. Для последовательной эмуляции I2C или SPI устройство вы можете использовать соответствующую кодировку программа с массивы для буферов отправки и получения по порядку. подражать сложному поведению устройство, **такие как GPS Устройства, серийный EEPROM Устройства и т. д.**

а) Ведущий или ведомый SPI устройство. UnoArduSimV2.4. Продлил **'spi.h'** библиотека, чтобы разрешить

режим Slave S {PI через необязательный 'mode' параметр в 'SPI.begin(int mode = SPI\_MASTR)', Явно передать 'SPI\_SLV' выбрать подчиненный режим (вместо использования основного режима по умолчанию). Теперь вы также можете определить пользовательское прерывание функциональный модуль. (давайте назовем это 'onSPI') в ведущем или подчиненном программа для передачи байтов путем вызова другое добавленное расширение 'SPI.attachInterrupt(user\_onSPI)', **Сейчас же** с Alling 'rxbyte=SPI.transfer(tx\_byte)' изнутри вашего 'user\_onSPI' функциональный модуль очистит флаг прерывания и **возвращение немедленно** с только что полученным байтом в вашей переменной 'rxbyte', Кроме того, вы можете избежать прикрепления прерывание SPI и вместо этого просто позвоните 'rxbyte=SPI.transfer(tx\_byte)' изнутри вашего основного программа - этот вызов будет **блок выполнение** пока байт SPI не был передан, и будет затем **возвращение** с недавно полученным байтом внутри 'rxbyte',

б) Универсальный **серийный I/O** устройство. Вы можете общаться с Master плата, используя либо 'Serial' или 'SoftwareSerial' определяется внутри вашего раба программа— для 'SoftwareSerial' Вы должны определить 'txpin' а также 'rxpin' противоположно к тем из Master, так что раб получает на пин, на котором мастер передает (и наоборот), но для **только 'Serial'**, 1 и 0 пинах уже перевернуты для вас.

в) Общий ведущий или подчиненный 'I2C' устройство. Операция Ведомый была добавлена для завершения UnoArduSim; реализация 'Wire.h' библиотека (функциональные модули 'begin(address)', 'onReceive()' а также 'onRequest' теперь были реализованы для поддержки операций в подчиненном режиме).

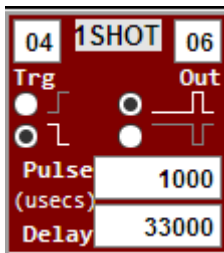
г) универсальный цифровой **пульсатор**, С помощью 'delayMicroseconds()' а также 'digitalWrite()' звонки внутри 'loop()' в вашем 'PROGIO' программа, вы можете определить 'HIGH' а также 'LOW' интервалы пульса. Добавляя отдельный 'delay()' позвоните в свой 'setup()' функциональный модуль, вы можете отложить запуск этой последовательности импульсов. Вы даже можете изменить ширину импульса как время прогрессирует с помощью счетчика переменная. Вы также можете использовать отдельный 'IOx' пин в качестве триггера для запуска синхронизация эмулируемого '1Shot' (или двойного, тройного и т. Д.) устройство, и вы можете управлять шириной производимого импульса, чтобы она изменялась любым желаемым вам способом с течением времени.

д) случайный сигнализатор. Это вариация на цифровой **пульсатор** который также использует звонки 'random()' а также 'delayMicroseconds()' генерировать случайные моменты времени, в которые 'digitalWrite()' сигнал на любой выбранный пин поделился с мастером. Используя все четыре 'IOx' пинах допускает четыре одновременных (и уникальных) сигнала.



## Один-Сигнал ('1SHOT')

Этот 'I/O' устройство эмулирует однократный выстрел цифровой, который может генерировать импульс выбранной полярности и ширины импульса на своем 'Out' пин, возникающий после указанной задержки от фронта запуска, полученного на его **Trg** (триггерный) вход пин. Как только заданный фронт запуска получен, начинается синхронизация, и тогда новый импульс запуска не будет распознан, пока импульс 'Out' не будет сгенерирован (и полностью не завершен).

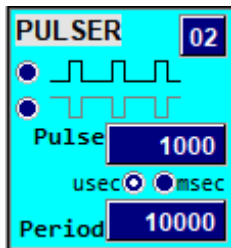


Одним из возможных применений этого устройство является моделирование датчики измерения дальности ультразвука, которые генерируют импульс дальности в ответ на импульс запуска. Его также можно использовать там, где вы хотите сгенерировать входной сигнал пин, синхронизированный (после выбранной вами задержки) с выходным сигналом пин, созданным вашим программой. '**Pulse**' и значения '**Delay**' можно масштабировать из основного окна **Инструмент-Bar** Управление ползунком 'I/O \_\_\_\_ S' с добавлением суффикса 'S' (или 's') или один (или оба).

Другое использование этого устройство - тестирование программа, использующего прерывания, и вы хотели бы посмотреть, что произойдет, если **специальная инструкция программа** прерывается. Временно отключите 'I/O' Устройство, который вы подключили к пин 2 (или пин 3), и замените его на '1SHOT' устройство, к которому 'Out' пин подключен к пин 2 (или пин3, соответственно), затем вы можете запустить его вход 'Trg' (при условии установки чувствительности переднего фронта), вставив пара инструкций { '**digitalWrite(LOW)** ', '**digitalWrite(HIGH)** ' } **только до** инструкции, внутри которой вы хотите, чтобы прерывание произошло. Установите 1SHOT; с '**Delay**' рассчитывать время импульса, генерируемого на 'Out', внутри инструкции программа, которая следует за этой парой инструкций запуска. Обратите внимание, что некоторые инструкции маскируют прерывания (например, '**SoftwareSerial.write(byte)** ', а так не может быть прервано.

## Цифровой Генератор Импульсов ('PULSER')

Этот 'I/O' устройство эмулирует простой генератор импульсов цифровой осциллограмма, который генерирует периодический сигнал, который может быть применен к любому выбранному 'Uno' или 'Mega' пин.



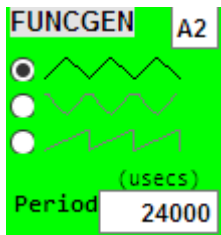
Период и длительность импульса (в микросекундах) можно установить с помощью полей редактирования - минимальный допустимый период составляет 50 микросекунд, а минимальная ширина импульса составляет 10 микросекунд. Вы можете выбирать между значениями синхронизация в микросекундах ('usec') и миллисекундах ('msec'), и этот выбор будет сохранен вместе с другими значениями, когда вы 'Save' из **Конфигурировать I / O Устройства**,

Также можно выбрать полярность: либо положительные импульсы переднего фронта (от 0 до 5 В), либо отрицательные импульсы переднего фронта (от 5 В до 0 В).

Значения '**Pulse**' и '**Period**' можно масштабировать из основного **Инструмент-Bar** Управление ползунком масштабного коэффициента 'I/O \_\_\_\_ S' путем добавления суффикса 'S' (или 's') к одному (или к обоим). Это позволяет вам затем изменить значение '**Pulse**' или '**Period**' **динамично** во время выполнения.

## Аналоговый Генератор Функций ('FUNCGEN')

Этот 'I/O' устройство эмулирует простой генератор аналоговый осциллограмма, который генерирует периодический сигнал, который может быть применен к любому выбранному 'Uno' или 'Mega' пин.



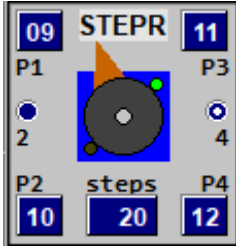
Период (в микросекундах) можно установить с помощью поля редактирования - минимально допустимый период составляет 100 микросекунд. осциллограмма, который он создает, может быть выбран синусоидальным, треугольным или пилообразным (для создания прямоугольной волны используйте вместо этого 'PULSER'). В меньшие периоды для моделирования произведенного осциллограмма используется меньшее количество образцов за цикл (только 4 образца за цикл в период = 100 микросекунд).

'Period' значение может быть масштабировано от основного окна **Инструмент-Bar** Ползунок масштабного коэффициента 'I/O \_\_\_\_ S' можно контролировать, добавив в качестве суффикса букву 'S' (или 's'). Это позволяет вам затем изменить Значение '**Period**' **динамично** во время выполнения.



## Шаговый двигатель ('STEPR')

Этот 'I/O' устройство эмулирует 6V биполярный или униполярный Шаговый двигатель со встроенным контроллером толкатель толкнул от **либо два** (на **P1** , **P2** ) **или четыре** (на **P1** , **P2** , **P3** , **P4** ) контрольные сигналы. Количество шагов на оборот также может быть установлено. Вы можете использовать '**Stepper.h**' функциональные модули '**setSpeed()**' а также '**step()**' до толкнул 'STEPR'. Кроме того, 'STEPR' будет *также ответить* к себе '**digitalWrite()**' " побитовые"сигналы толкнул.



Мотор точно смоделирован как механически, так и электрически. Мотор-толкатель падения напряжения и изменения реактивного сопротивления и индуктивности моделируются вместе с реалистичным моментом инерции относительно удерживающего момента. Обмотка ротора двигателя имеет смоделированное сопротивление  $R = 6$  Ом и индуктивность  $L = 6$  милли-Генри, которая создает электрическую постоянную времени 1,0 миллисекунды. Из-за реалистичного моделирования вы заметите, что очень узкие импульсы управления пин *не получают* шаг двигателя - как из-за конечного времени нарастания тока, так и из-за влияния

инерции ротора. Это согласуется с тем, что наблюдается при управлении реальным шаговым двигателем от 'Uno' или 'Mega' с, конечно, **и требуется** ) Двигатель толкатель чип между проводами двигателя и 'Uno' или 'Mega'!

Несчастный ошибка в Ардуино '**Stepper.h**' Код библиотеки означает, что при сбросе шаговый двигатель не будет находиться в положении Шаг 1 (из четырех шагов). Чтобы преодолеть это, пользователь должен использовать '**digitalWrite()**' в его / ее '**setup()**' рутина для инициализации контрольных уровней пин до '**step(1)**' уровни, соответствующие 2-пин (0,1) или 4-пин (1,0,1,0), и позволяют двигателю 40-100 миллисекунд, чтобы переместиться в исходное заданное положение двигателя на 12 часов.

Обратите внимание, что **редуктор не поддерживается напрямую** из-за недостатка места, но вы можете эмулировать его в своем программа, внедрив счетчик по модулю N переменная и только вызывая '**step()**' когда этот счетчик достигает 0 (для уменьшения передачи с коэффициентом N).

Начиная с версии 2.6 этот устройство теперь включает в себя 'sync' LED (зеленый для синхронизации или красный при отключении на один или несколько шагов). Кроме того, в дополнение к количеству шагов на оборот, два дополнительных (скрытых) значения могут опционально указывается в IODevs.txt файл для указания механической нагрузки - например, значения 20, 50, 30 определяют 20 шагов на оборот, момент инерции нагрузки, в 50 раз превышающий момент самого ротора двигателя, и момент нагрузки 30 процентов полного удерживающего момента двигателя.

## Пульсирующий Шаговый двигатель ('PSTEPR')

Этот 'I/O' устройство эмулирует 6V **микро-шаговый** биполярный Шаговый двигатель со встроенным контроллером толкатель толкнул по **импульсный 'Step'** пин, активный-низкий '**EN\***' (включить) пин и '**DIR**' (направление) пин , Количество полных шагов на оборот также может быть установлено напрямую, наряду с количеством микрогрупп на полный шаг (1,2,4,8 или 16). В дополнение к этим настройкам, два дополнительных (скрытых) значения могут опционально указывается в IODevs, txt файл, чтобы указать механическую нагрузку - например, значения 20, 4, 50, 30 определяют 20 шагов на оборот, 4 микрошага на полный шаг, момент инерции нагрузки в 50 раз больше, чем у двигателя сам ротор, и крутящий момент нагрузки 30 процентов от полного удерживающего момента двигателя.

Вы должны написать код толкнул контроля пинах соответственно.

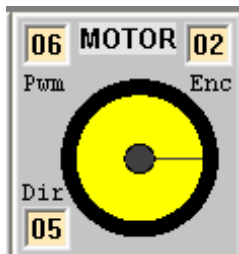


Мотор точно смоделирован как механически, так и электрически. Мотор-толкатель падения напряжения и изменения реактивного сопротивления и индуктивности моделируются вместе с реалистичным моментом инерции относительно удерживающего момента. Обмотка ротора двигателя имеет смоделированное сопротивление  $R = 6$  Ом и индуктивность  $L = 6$  милли-Генри, которая создает электрическую постоянную времени 1,0 миллисекунды.

Это устройство включает в себя желтую активность 'STEP' LED и 'sync' LED (ЗЕЛЕНЫЙ для синхронизации или КРАСНЫЙ при выключении одним или несколькими шагами).

## Двигатель постоянного напряжения ('MOTOR')

Этот 'I/O' устройство эмулирует 6-вольтовый двигатель постоянного тока 100: 1 с встроенным контроллером толкатель толкнул сигналом широтно-импульсной модуляции (на его **широтнo-импульснaя модуляция** вход), и сигнал управления направлением (на его **Dir** вход). Двигатель также имеет выход энкодера колеса, который толкает на его **Enc** выход пин. Ты можешь использовать '**analogWrite()**' к толкнул **широтнo-импульснaя модуляция** пин с 490 Гц (на пинах 3,9,10,11) или 980 Гц (на пинах 5,6) с широтно-импульсной модуляцией осциллограмма с рабочим циклом от 0,0 до 1,0 ( '**analogWrite()**' значения от 0 до 255). Кроме того, 'MOTOR' будет *также ответить* к себе '**digitalWrite()**' " побитовые" сигналы толкнул.



Двигатель точно смоделирован как механически, так и электрически. Учет (низкого) падения напряжения на МОП-транзисторе-драйвере и реалистичного крутящего момента редуктора без нагрузки дает полную скорость почти 3 об / с и крутящий момент чуть менее 10 кг-см (происходящий при устойчивом рабочем цикле ШИМ 1.0) с механической постоянной времени приблизительно 40 миллисекунд (которая увеличивается за счет инерции любой заданной нагрузки). В пользовательском файле «IODevS.txt» можно указать три (необязательных) значения параметра - «F» или «B» (для режима выбега на выбеге или режима торможения, когда Pwm НИЗКОЕ), а затем постоянный момент нагрузки в процентах от опрокидывания. крутящий момент, затем момент инерции

нагрузки как целое число, кратное внутренней инерции ротора двигателя (эти значения по умолчанию равны «F», 0 и 1, если ничего не указано). Кроме того, имеется встроенный постоянный противодействующий крутящий момент коробки передач, равный 10% крутящего момента при остановке (который увеличивает крутящий момент нагрузки).

Обмотка ротора двигателя имеет смоделированное сопротивление  $R = 2$  Ом и индуктивность  $L = 300$  мкГн, что создает электрическую постоянную времени 150 микросекунд. Из-за реалистичного моделирования вы заметите, что очень узкие импульсы ШИМ *не получают* вращение двигателя - как из-за конечного времени нарастания тока, так и из-за значительного времени простоя после каждого узкого импульса. Они объединяются, чтобы вызвать недостаточный импульс ротора, чтобы преодолеть пружинную защелку редуктора при статическом трении. Следствие при использовании '**analogWrite()**' рабочий цикл ниже примерно 0,125 не заставит двигатель сдвинуться - это согласуется с тем, что наблюдается при движении реального редукторного двигателя от 'Uno' или 'Mega' с, конечно, соответствующим ( ***и требуется*** ) Модуль двигателя толкатель между двигателем и 'Uno' или 'Mega'!

Датчик эмуляции двигателя представляет собой датчик оптического прерывания, установленный на валу, который выдает 50% -ный рабочий цикл осциллограмма, имеющий 8 полных периодов высокого-низкого за один оборот колеса (поэтому ваш программа может воспринимать изменения вращения колеса с разрешением 22,5 градуса).

## Серводвигатель ('SERVO')

Этот 'I/O' устройство имитирует 6-вольтовый сервопривод постоянного тока PWM-толкнул с управлением положением. Механические и электрические параметры моделирования для работы сервопривода будут точно соответствовать параметрам стандартного сервопривода HS-422. Сервопривод имеет максимальную скорость вращения около 60 градусов за 180 миллисекунд, Если левый нижний флажок установлен, сервопривод становится **непрерывное вращение** сервопривод с той же максимальной скоростью, но теперь ширина импульса ШИМ устанавливает **скорость** а не угол



Ваш программа должен иметь '**#include <Servo.h>**' линия, прежде чем объявить '**Servo**' экземпляр (ы) *если вы решите использовать функциональность библиотеки 'Servo.h'* например, '**Servo.write()**', '**Servo.writeMicroseconds()**' Кроме того, 'SERVO' также реагирует на '**digitalWrite()**' "Битовые" сигналы. Из-за внутренней реализации UnoArduSim, вы ограничены 6 'SERVO' Устройства.

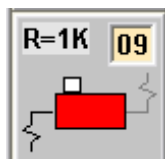
## Пьезоэлектрический Динамик ('PIEZO')



Этот устройство позволяет вам "прослушивать" сигналы на любом выбранном 'Uno' или 'Mega' пин и может быть полезным дополнением к светодиодам для отладки вашей работы программа. Вы также можете повеселиться, играя соответствующие мелодии 'tone()' а также 'delay()' звонки (хотя нет прямоугольной фильтрации осциллограмма, поэтому вы не услышите "чистых" заметок).

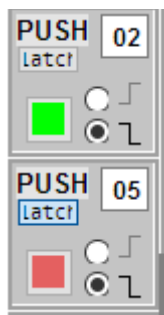
Вы также можете прослушивать подключенный 'PULSER' или 'FUNCEN' устройство, подключив 'PIEZO' к пин, который устройство толкает на.

## Слайд резистор ('R=1K')



Этот устройство позволяет пользователю подключаться к 'Uno' или 'Mega' пин либо с повышающим сопротивлением на 1 кОм до + 5 В, либо с понижающим сопротивлением на 1 кОм на землю. Это позволяет вам моделировать электрические нагрузки, добавленные к реальному оборудованию устройство. Щелкнув левой кнопкой мыши по ползунку **тело** Вы можете переключать желаемый выбор подтягивания или выпадающего. Использование одного или нескольких из этих Устройства позволит вам установить один (или несколько) -битный "код" для вашего программа для чтения и ответа.

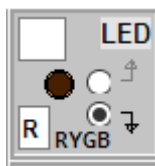
## Кнопка Тактовая ('PUSH')



Этот 'I/O' устройство эмулирует нормально открытый **мгновенное ИЛИ с фиксацией** однополюсная, однополюсная (SPST) кнопка с повышающим (или понижающим) сопротивлением 10 кОм. Если для устройство выбран переходной режим с передним фронтом, кнопочные контакты будут подключены между устройство и пин и + 5 В с опусканием 10 кОм на землю. Если для устройство выбран переход с падающей кромкой, кнопочные контакты будут подключены между устройство и пин и землей с напряжением 10 кОм до + 5В.

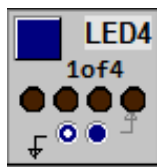
Щелкнув левой кнопкой мыши по кнопке или нажав любую клавишу, вы закрываете контакт кнопки. В **моментальный** режим, он остается закрытым, пока вы удерживаете кнопку мыши или клавишу, и в **защелка** режим (включается нажатием на 'latch' кнопка) он остается закрытым (и другого цвета), пока вы не нажмете кнопку еще раз. Отскок контактов (в течение 1 миллисекунды) будет производиться каждый раз, когда вы использовать **клавиша для интервалов** нажать на кнопку.

## Цветной LED ('LED')



Вы можете подключить LED между выбранным 'Uno' или 'Mega' пин (через токоограничивающий резистор 1 кОм скрытой серии встроенный) к заземлению или к + 5 В - это дает вам возможность включить подсветку LED при подключенном 'Uno' или 'Mega' пин является 'HIGH' или вместо когда он является 'LOW', Цвет LED может быть выбран как красный ('R'), желтый ('Y'), зеленый ('G') или синий ('B'), используя его поле редактирования.

## 4-LED ряд ('LED4')



Вы можете подключить этот ряд из 4 цветных светодиодов между выбранным набором 'Uno' или 'Mega' пинах (каждый из которых имеет токоограничивающий резистор 1 кОм скрытой серии встроенный) к заземлению или к + 5 В - это дает вам возможность выбора Светодиоды загораются, когда подключенный 'Uno' или 'Mega' пин 'HIGH' или вместо когда он является 'LOW',

'1of4' Поле ввода пин принимает одно число пин, которое будет означать **первый из четырех подряд** 'Uno' или 'Mega' пинах, которые будут подключаться к 4 светодиодам.

Цвет LED ('R', 'Y', 'G' или 'B') является **скрытая опция** это может быть **быть выбранным только редактирование IODevices.txt файл** (который Вы можете создать с помощью **Сохранить** от **Конфигурировать | I/O** Устройства диалоговое окно).

## 7-сегментный LED Цифра ('7SEG')



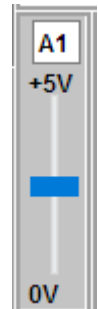
Вы можете подключить этот 7-сегментный дисплей Цифра LED к выбранному набору **четыре последовательных 'Uno' или 'Mega' пинах, которые дают код шестнадцатеричный** для требуемого отображаемого цифра (от '0' до 'F') и включите или выключите этот цифра с помощью CS \* пин (активный-НИЗКИЙ для ВКЛ).

Это устройство включает в себя декодер встроенный, который использует **активный ВЫСОКИЙ** уровни на четырех последовательных **'1of4'** пинах для определения запрашиваемого шестнадцатеричный цифра для отображения. Уровень Те на самом низком номере пин (тот, который отображается в **'1of4'** поле редактирования) представляет младший бит 4-битного кода шестнадцатеричный.

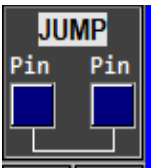
Цвет сегментов LED ('R', 'Y', 'G' или 'B') является **скрытая опция** это может быть **быть выбранным только редактирование IODevices.txt файл** Вы можете создать с помощью **Сохранить** от **Конфигурировать | I/O** Устройства диалоговое окно.

## Аналоговый Слайдер

Ползунковый потенциометр 0-5 В может быть подключен к любому выбранному 'Uno' или 'Mega' пин для получения статического (или медленно меняющегося) уровня напряжения аналоговый, который будет считываться **'analogRead()'** в качестве значения от 0 до 1023. С помощью мыши перетащите ползунок аналоговый или щелкните, чтобы перейти к нему.



## Pin Перемычка ('JUMP')



Вы можете подключить два 'Uno' или 'Mega' пинах вместе, используя это устройство (если какой-либо электрический конфликт обнаружен при заполнении второго номера пин, выбранное соединение не разрешается и пин отключается).

Эта переключатель устройство имеет ограниченную полезность и наиболее полезна в сочетании с прерываниями для тестирования программа, экспериментов и учебные цели. **Начиная с UnoArduSim V2.4, вы можете обнаружить, что использование 'PROGIO' устройство**

**обеспечивает большую гибкость, чем методы прерывания толкнул, приведенные ниже.**

Три возможных использования этого устройство следующие:

- 1) Вы можете **создать вход цифровой для тестирования вашего программа** который имеет более сложной синхронизация, чем может быть произведен с использованием любого из набора Стандарт 'I/O' Устройства, а именно:

Определить прерывание функциональный модуль (назовем его **'myIntr'**) и делать **'attachInterrupt(0, myIntr, RISING)'** внутри вашего **'setup()'**. Подключите **пульсатор** От устройство до пин2 - сейчас **'myIntr()'** будет Выполнить каждый раз **пульсатор** возникает нарастающий фронт. Ваш **'myIntr()'** функциональный модуль может быть алгоритм у вас есть запрограммированный (используя глобальный счетчик переменные и, возможно, даже **'random()'**) изготовить осциллограмма вашего собственного дизайна на любом доступном **'OUTPUT'** пин (допустим, это пин 9). Сейчас же **ПРЫГАТЬ** пин 9 по вашему желанию 'Uno' или 'Mega' 'INPUT'пин, чтобы применить сгенерированный цифровой осциллограмма к этому входу пин (чтобы проверить ваш программа; ответ этого конкретного осциллограмма). , Вы можете генерировать последовательность импульсов, или последовательные символы, или просто граничные переходы, любой произвольной сложности и с различными интервалами. Обратите внимание, что если ваш основной программа звонит **'micros()'** (или вызывает любой

функциональный модуль, который полагается на него), его `'return'` ценность **будет увеличено** по времени, проведенному внутри вашего `'myIntr()'` функциональный модуль каждый раз, когда прерывание срабатывает. Вы можете произвести быстрый всплеск точно рассчитанных граней, используя вызовы `'delayMicroseconds()'` от внутри `'myIntr()'` (возможно, чтобы создать целый **байт** из высокая передача скорость передачи), или просто сгенерировать один переход за прерывание (возможно, для генерации **один бит** передачи с низким уровнем скорость передачи) с **пульсатор** устройство **'Period'** выбран в соответствии с вашими потребностями синхронизация **пульсатор** ограничивает его минимум **'Period'** до 50 микросекунд).

## 2) Вы можете **Эксперимент с подсистемой обратной связи:**

Например, отключите **'SERIAL'** I/O устройство TX **'00'** пин (отредактируйте его пустым), а затем **ПРЫГАТЬ** **'Uno'** или **'Mega'** пин **'01'** вернуться к **'Uno'** или **'Mega'** пин **'00'** эмулировать аппаратную петлю ATmega **'Serial'** подсистема. Теперь в вашем тесте программа, внутри `'setup()'` сделать **Один** `'Serial.print()'` из слово или символ, и внутри вашего `'loop()'` вернуть все полученные символы (когда `'Serial.available()'`) сделав `'Serial.read()'` с последующим `'Serial.write()'`, а затем посмотреть, что происходит. Вы могли заметить, что похожий **'SoftwareSerial'** петля-обратно **не удастся** (как это было бы в реальной жизни - программное обеспечение не может делать две вещи одновременно).

Вы также можете попробовать **SPI** заикливание с помощью **ПРЫГАТЬ** подключить пин 11 (MOSI) обратно к пин 12 (MISO).





3) Вы можете **подсчитать количество и / или измерить расстояние между переходами определенного уровня на любом 'Uno' или 'Mega' выход пин X** которые происходят в результате сложного Инструкция Arduino или библиотека функциональный модуль (как примеры: `'analogWrite()'`, или `'OneWire::reset()'`, или `'Servo::write()'`), следующее:

**ПРЫГАТЬ** пин **Икс** прервать пин **2** и внутри вашего `'myIntr()'` использовать `'digitalRead()'` и `'micros()'` вызов, и сравнить с сохраненными уровнями и временем (от предыдущих прерываний). При необходимости вы можете изменить чувствительность к краям для следующего прерывания, с помощью `'detachInterrupt()'` а также `'attachInterrupt()'` от **внутри** ваш `'myIntr()'`, Обратите внимание, что вы не сможете отслеживать пин переходы, которые происходят слишком близко друг к другу (ближе, чем общее время выполнение ваш `'myIntr()'` функциональный модуль), например, те, которые происходят с переносами I2C или SPI, или с высоким скорость передачи **'Serial'** переводы (даже если ваше прерывание функциональный модуль не будет мешать переходу синхронизация этих аппаратных передач). Также обратите внимание, что программно-опосредованные переводы (например, `'OneWire::write()'` а также `'SoftwareSerial::write()'`) находятся Преднамеренно защищен от прерываний (их библиотечный код временно отключает все прерывания, чтобы предотвратить сбои синхронизация), поэтому вы не можете проводить измерения внутри тех, кто использует этот метод.

Хотя вы можете вместо этого сделать те же измерения расстояния между краями **визуально** в **Цифровой формы волны** окно, если вас интересует минимальное или максимальное расстояние между большим количеством переходов или считая переходы, делая это с помощью этого `'myIntr()'` -plus- **ПРЫГАТЬ** Техника удобнее. И вы можете измерьте, например, вариации расстояния между основными переходами программа (из-за того, что ваше программное обеспечение использует разные пути выполнение в разное время выполнение), сделать вид программа "Профилирование".

## меню

### Файл:





<u><b>Загрузить INO или PDE Prog (Ctrl-L)</b></u> 	Позволяет пользователю выбрать программа файл с выбранным расширением. программа немедленно получает Анализировать
<u><b>Изменить/Просмотреть (Ctrl-E)</b></u>	Открывает загруженный программа для просмотра / редактирования.
<u><b>Сохранить</b></u> 	Сохранить отредактированное содержание программа возвращается к оригинальному программа файл.
<u><b>Сохранить как</b></u>	Сохранить отредактированное содержимое программа под другим именем файл.
<u><b>следующий ('#include')</b></u> 	Продвигает <b>Панель с кодом</b> отображать следующий '#include' файл
<u><b>предыдущий</b></u> 	Возвращает <b>Панель с кодом</b> дисплей к предыдущему файл
<u><b>Выход</b></u>	Выход из UnoArduSim после напоминания пользователю сохранить любые измененные файл.

### Найти:

<u><b>Ascend Call Stack</b></u> 	Перейти к предыдущему абоненту функциональный модуль в стек вызовов - <b>Панель с переменными</b> отрегулируйте, чтобы показать местный переменные для этого функциональный модуль
<u><b>Стек вызовов Descend</b></u> 	Перейти к следующему названию функциональный модуль в стек вызовов - <b>Панель с переменными</b> отрегулировать, чтобы показать локальный переменные для этого функциональный модуль
<u><b>Установить текст Искать (Ctrl-F)</b></u> 	Активировать <b>Инструмент-Bar</b> Найти поле ввода для определения текста, который нужно искать следующим (и добавляет первое слово из выделенной в данный момент строки в <b>Панель с кодом</b> или <b>Панель с переменными</b> если один из них имеет фокус).
<u><b>Найти Следующий текст</b></u> 	Перейти к следующему вхождению текста в <b>Панель с кодом</b> (если он имеет активный фокус), или к следующему вхождению текста в <b>Панель с переменными</b> (если вместо этого он имеет активный фокус).
<u><b>Найти Предыдущий текст</b></u> 	Перейти к предыдущему вхождению текста в <b>Панель с кодом</b> (если он имеет активный фокус), или к предыдущему вхождению текста в <b>Панель с переменными</b> (если вместо этого он имеет активный фокус).



## Выполнить:

<b><u>Шаг с заходом (F4)</u></b>	Шаги выполнения вперед на одну инструкцию, или <i>в так называемый функциональный модуль</i> ,
<b><u>Шаг с обходом (F5)</u></b>	Шаги выполнения вперед на одну инструкцию, или <i>одним полным звонком функциональный модуль</i> ,
<b><u>Шаг с выходом (F6)</u></b>	Авансы выполнения по <i>Достаточно, чтобы оставить текущий функциональный модуль</i> ,
<b><u>Выполнить До (F7)</u></b> 	Работает программа, <i>остановка на нужной линии программа</i> - прежде чем использовать Выполнить До, вы должны сначала щелкнуть основной момент на нужную строку программа.
<b><u>Выполнить Пока (F8)</u></b> 	Запускает программа до тех пор, пока не произойдет запись в переменная с текущим основным момент в <b>Панель с переменными</b> (нажмите на один, чтобы установить начальный основной момент).
<b><u>Выполнить (F9)</u></b>	Работает программа.
<b><u>Приостановить (F10)</u></b> 	Остановки программа выполнения ( <i>и замораживает время</i> ).
<b><u>Сброс</u></b> 	Сбрасывает программа (все значения переменные сбрасываются в значение 0, а все указатели переменные сбрасываются в 0x0000).
<b><u>Анимация</u></b>	Автоматически пошаговые последовательные строки программа с <i>добавленной искусственной задержкой</i> и выделение текущей строки кода. Работа в режиме реального времени и звуки теряются.
<b><u>Замедленное движение</u></b>	Замедляет время в 10 раз.

## Опции:

<b><u>Шаг с обходом Structors/ Операторы</u></b>	Пролетите сквозь конструкторы, деструкторы и перегрузку оператора функциональные модули во время любого шага (т.е. он не остановится внутри этих функциональные модули).
<b><u>Регистр-Allocation</u></b>	Назначьте локальные функциональный модуль для регистров АТмега свободно вместо стека (генерирует несколько уменьшенное использование RAM).
<b><u>Ошибка при неинициализированном</u></b>	Пометить как ошибку Анализировать везде, где ваш программа пытается использовать переменная без предварительной инициализации его значения (или хотя бы одного значения внутри массив).
<b><u>добавленной 'loop()' задержка</u></b>	Добавляет 1000 микросекунд задержки каждый раз 'loop()' вызывается (если нет других вызовов программа на 'delay()' в любом месте) - полезно избегать слишком большого отставания от реального времени.
<b><u>Разрешить вложенные прерывания</u></b>	Разрешить повторное включение с 'interrupts()' изнутри подпрограммы обслуживания прерываний пользователя.



## Конфигурировать:

<b><u>'I/O' Устройства</u></b>	Открывает диалоговое окно, позволяющее пользователю выбрать тип (ы) и номера требуемого 'I/O' Устройства. Из этого диалогового окна вы также можете Сохранить 'I/O' Устройства в текст файл и / или Загрузить 'I/O' Устройства из ранее сохраненного (или отредактированного) текста файл (включая все соединения пин, интерактивные настройки и введенные значения).
<b><u>Настройки</u></b>	Открывает диалоговое окно, позволяющее пользователю устанавливать предпочтения, в том числе автоматический отступ исходного программа-строк, разрешать синтаксис Expert, выбирать шрифт гарнитура, выбирать больший размер шрифта, устанавливать границы массив, разрешать ключевые слова логического оператора, показывая программа загрузка , выбор версии плата и длина буфера TWI (для I2C Устройства).

## ПеремОбновить:

<b><u>Разрешить авто (-) Сокращаться</u></b>	Разрешить UnoArduSim для сокращаться отображается расширенный массивы / объектов при отставании в режиме реального времени.
<b><u>минимальная</u></b>	Только освежить <b>Панель с переменными</b> отображать 4 раза в секунду.
<b><u>Основной момент изменения</u></b>	Основной момент изменил значения переменная при работе (может вызвать замедление).

## Окна:

<b><u>Serial Монитор</u></b>	Подключите последовательный ввод / вывод устройство к пинах 0 и 1 (если его нет) и потяните больший 'Serial' Монитор TX / RX с текстом окно.
<b><u>Восстановить все</u></b>	Восстановите все свернутые дочерние элементы окна.
<b><u>Pin Цифровые Осциллограммы</u></b>	Восстановить свернутый Pin Цифровые Осциллограммы окно.
<b><u>Pin Аналоговый Осциллограммо</u></b>	Восстановить свернутый Pin Аналоговый Осциллограммо окно.

## Помощь:

<b><u>Быстрый Помощь Файл</u></b>	Открывает UnoArduSim_QuickHelp PDF файл.
<b><u>Полный Помощь Файл</u></b>	Открывает UnoArduSim_FullHelp PDF файл.
<b><u>Исправления Ошибка</u></b>	Просмотреть важные исправления ошибка с момента предыдущего выпуска.
<b><u>Изменение / Улучшение</u></b>	Просмотр значительных изменений и улучшений по сравнению с предыдущим выпуском.
<b><u>Около</u></b>	Отображает версию, авторское право.

## 'Uno' или 'Mega' Плата и 'I/O' Устройства

'Uno' или 'Mega' и подключенный 'I/O' Устройства все точно смоделированы электрически, и вы сможете дома получить хорошее представление о том, как ваш программы будет вести себя с фактическим оборудованием, и все электрические пин конфликты будут помечены.

## Синхронизация

UnoArduSim выполняет достаточно быстро на ПК или планшете, что может ( *в большинстве случаев* ) действия модели программа в режиме реального времени, **но только если ваш программа включает в себя** хоть какой то маленький 'delay()' звонки или другие звонки (такие как 'print()' или 'SPI.transfer()' и т. д.) естественно синхронизируйте его с реальным временем (см. ниже).

Для этого UnoArduSim использует таймер обратного вызова Окна функциональный модуль, который позволяет ему точно отслеживать реальное время. выполнение из ряда инструкций программа моделируется в течение одного таймера, а инструкции, для которых требуется больше выполнение (например, вызовы 'delay()' ) может понадобиться использовать несколько таймеров. Каждая итерация таймера обратного вызова функциональный модуль корректирует системное время с использованием системных аппаратных часов, так что программа выполнение постоянно регулируется, чтобы не отставать от режима реального времени.

*Единственный раз выполнение ставка должен отставать от реального времени* когда пользователь создал плотные петли **без дополнительной задержки** или 'I/O' Устройства сконфигурированы для работы с очень высокими частотами 'I/O' устройство (и / или скорость передачи), которые будут генерировать чрезмерное количество событий изменения уровня пин и связанных с ними перегрузок обработки. UnoArduSim справляется с этой перегрузкой, пропуская некоторые интервалы таймера для компенсации, а затем замедляет прогрессирование программа до **ниже реального времени** ,

Кроме того, программы с большим массивы отображается или снова с плотными петлями **без дополнительной задержки** может вызвать высокую частоту вызовов функциональный модуль и генерировать высокую **Панель с переменными** загрузка обновлений дисплея приводит к тому, что она отстает от реального времени - UnoArduSim автоматически снижает частоту обновления переменная, чтобы не отставать, но когда требуется еще большее снижение, выберите **Минимальный**, от **ПеремОбновить** меню указать только четыре обновления в секунду.

Точное моделирование времени выполнение с точностью до миллисекунды для каждой инструкции или операции программа **не сделано** - только очень приблизительные оценки для большинства были приняты для целей моделирования. Тем не менее, синхронизация из 'delay()' , а также 'delayMicroseconds()' функциональные модули и функциональные модули 'millis()' а также 'micros()' все совершенно точно, **и до тех пор, пока вы используете хотя бы одну задержку функциональные модули** в петле где-то в вашей программа, **или** вы используете функциональный модуль, который естественно связывает себя с операцией в реальном времени (например, 'print()' который привязан к выбранному скорости передачи), то смоделированная производительность вашей программа будет очень близка к реальному времени (опять же, за исключением явного чрезмерного высокочастотного события изменения уровня пин или чрезмерных разрешенных пользователем обновлений Переменные, которые могут замедлить его).

Чтобы увидеть эффект отдельных инструкций программа в *курица бежит* может быть желательно иметь возможность замедлить ход событий. Коэффициент замедления времени 10 может быть установлен пользователем в меню **Выполнить** ,

## 'I/O' Устройство Синхронизация

Эти виртуальные Устройства получают в реальном времени сигнализацию об изменениях, которые происходят на их входе пинах, и вырабатывают соответствующие выходы на своих выходах пинах, которые затем могут распознаваться 'Uno' или 'Mega' - поэтому они по своей природе синхронизируются с программа выполнение. Внутренний 'I/O' устройство синхронизация устанавливается пользователем (например, с помощью выбора скорости передачи или тактовой частоты), а события симулятора устанавливаются для отслеживания внутренней работы в реальном времени.

## Звуки

Каждый 'PIEZO' устройство издает звук, соответствующий изменениям электрического уровня, происходящим на подключенном пин, независимо от источника таких изменений. Чтобы синхронизировать звуки с программа выполнения, UnoArduSim запускает и останавливает воспроизведение соответствующего звукового буфера, когда выполнение запускается / останавливается.

Начиная с версии 2.0 звук теперь был изменен для использования аудио API Qt - к сожалению, его QAudioOutput 'class' не поддерживает зацикливание звукового буфера, чтобы избежать исчерпания сэмплов (как это может случиться во время более длительных операционных задержек). Поэтому, чтобы избежать подавляющего большинства раздражающих щелчков звука и прерывания звука во время задержек ОС, звук теперь отключается в соответствии со следующим правилом:

Звук отключается до тех пор, пока UnoArduSim не является "активным" окно (кроме случаев, когда новый дочерний объект окно только что был создан и активирован), **и даже** когда UnoArduSim является "активным" основным окно, но указатель мыши **снаружи** из его основная клиентская зона окно.

Обратите внимание, что это означает, что звук будет временно приглушен как король при наведении курсора мыши **над ребенком окно**, и будет отключен **если этот ребенок окно нажал, чтобы активировать его** (пока снова не щелкнет основной UnoArduSim окно, чтобы снова активировать его) ,

**Звук всегда можно включить, щелкнув в любом месте внутри клиентской области главного KN203 UnoArduSim.**

Из-за буферизации, с В реальном времени задержка составляет до 250 миллисекунд с соответствующего времени события на пин подключенного 'PIEZO'.

## Ограничения и неподдерживаемые элементы

### Включено Файлы

A '<>' - в скобках '#include' из '<Servo.h>', '<Wire.h>', '<OneWire.h>', '<SoftwareSerial.h>', '<SPI.h>', '<EEPROM.h>' а также '<SD.h>' **является** поддерживается, но они только эмулируются - сам файлы не ищется; вместо этого их функциональность напрямую "встроена" в UnoArduSim и действительна для фиксированной поддерживаемой версии Arduino.

Любой цитируемый '#include' (например, "supp.ino", "Myutil.cpp", или "Mylib.h") поддерживается, но все такие файлы должны **проживать в тот же каталог в качестве родителя программа файл** который содержит их '#include' (нет поиска в других каталогах). '#include' функция может быть полезна для минимизации количества кода программа, показанного в **Панель с кодом** в любое время. Жатка файлы с '#include' (т.е. те, которые имеют ".h" расширение) дополнительно заставит симулятор попытаться включить одноименный файл, имеющий ".cpp" расширение (если оно также существует в каталоге родительского программа).

### Динамическое распределение памяти и оперативная память

операторы 'new' а также 'delete' поддерживаются, как и родные Arduino 'String' объектов, **но не прямые звонки** 'malloc()', 'realloc()' а также 'free()' что они полагаются

Чрезмерное использование ОЗУ для объявлений переменная помечается во время Анализировать, а переполнение памяти ОЗУ отмечается во время программа выполнение. пункт в меню **Опции** позволяет вам эмулировать обычное распределение регистров ATmega, как это было бы сделано AVR компилятор, или моделировать альтернативную схему компиляции, которая использует только стек (в качестве опции безопасности в случае, если ошибка появляется в моем моделировании распределения регистров). Если бы вы использовали указатель для просмотра содержимого стека, он должен точно отражать то, что появилось бы в реальной аппаратной реализации.

## 'Flash' Распределение памяти

Память 'Flash' 'byte', 'int' а также 'float' переменные / массивы и соответствующий им доступ для чтения функциональные модули поддерживаются. Любые 'F()' Звонок функциональный модуль ('Flash' Макро) из любая буквенная строка **является** поддерживается, но единственной поддерживаемой строкой памяти 'Flash' прямого доступа функциональные модули являются: 'strcpy\_P()' а также 'memcpy\_P()', поэтому для использования другого функциональные модули вам нужно сначала скопировать строку 'Flash' в обычную оперативную память 'String' переменная, а затем работать с этой оперативной памятью 'String', Когда вы используете 'PROGMEM' Ключевое слово-модификатор переменная, оно должно появиться *перед* имя переменная, и это переменная **также должен быть объявлен** в виде 'const',

## 'String' Переменные

Родной 'String' библиотека почти полностью поддерживается с несколькими очень (и незначительными) исключениями.

'String' операторы поддерживаются +, + =, <, <=, >, > =, ==, **знак равно**, а также [], Обратите внимание, что: 'concat()' занимает **Один** аргумент, который является 'String', или 'char', или 'int' быть добавленным к оригиналу 'String' объект, **не** два аргумента, как ошибочно указано на веб-страницах Arduino Reference).

## Библиотеки Arduino

Только 'SoftwareSerial.h', 'SPI.h', 'Wire.h', 'OneWire.h', 'Servo.h', 'Stepper.h', 'SD.h', 'TFT.h' а также 'EEPROM.h' для Arduino V1.8.8 релиз в настоящее время поддерживается в UnoArduSimV2.6. Пытаясь '#include' ".cpp" а также ".час" файлы из других пока не поддерживаемых библиотеки будут не работа поскольку они будут содержать инструкции по сборке низкого уровня и неподдерживаемые директивы, а также нераспознанные файлы!

## указатели

Поддерживаются указатели на простые типы, массивы или объектов. Указатель может быть приравнен к массив того же типа (например, 'iptr = intarray' ), но тогда бы *нет последующей проверки границ массива* на выражение как 'iptr[index] ',

Функциональные модули может возвращать указатели, или 'const' указатели, но любой последующий уровень 'const' возвращаемый указатель игнорируется.

Там есть *без поддержки* для функциональный модуль звонки через **объявленные пользователем функциональный модуль-указатели** ,

## 'class' а также 'struct' Объектов

Хотя поддерживается полиморфизм и наследование (на любую глубину), 'class' или 'struct' можно определить только на максимум **один** база 'class' (т.е. **Многолучевая** наследование не поддерживается). Поддерживаются вызовы инициализации конструктора Base-'class' (через двоеточие) в строках объявления конструктора, но **не** инициализация членов с использованием той же записи двоеточия. Это означает, что объектов, которые содержат 'const' не-'static' переменные, или ссылочный тип переменные, не поддерживаются (это возможно только с указанными инициализациями элементов во время строительства)

Перегрузки оператора назначения копирования поддерживаются вместе с конструкторами перемещения и назначениями перемещения, но пользовательское преобразование объект ("переменная") функциональные модули не поддерживается.

## Сфера

Там нет поддержки для 'using' ключевое слово, или для 'namespace' или для 'file' сфера. Все нелокальные объявления по реализации предполагаются глобальными.

Любые 'typedef', 'struct', **или** 'class' определение (то есть, которые могут быть использованы для будущих объявлений), должны быть сделаны **Глобальный** сфера ( **местный** определения таких предметов внутри функциональный модуль не поддерживаются).

## Отборочные 'unsigned', 'const', 'volatile', 'static'

'unsigned' Префикс работает во всех нормальных юридических контекстах. 'const' ключевое слово, при использовании, должно **предшествуют** имя переменная или имя функциональный модуль или 'typedef' имя, которое объявляется - размещение его после имени приведет к ошибке Анализировать. За Объявления функциональный модуль, только функциональные модули с возвратом указателя может иметь 'const' появляются в их декларации.

Все UnoArduSim переменные являются 'volatile' реализацией, поэтому 'volatile' Ключевое слово просто игнорируется во всех объявлениях переменная. Функциональные модули не разрешается объявлять 'volatile' также не являются аргументами вызова функциональный модуль.

'static' ключевое слово разрешено для обычного переменные, а также для членов объект и member-функциональные модули, но явно запрещено для самих экземпляров объект ( 'class' / 'struct' ), для не члена функциональные модули и для всех аргументов функциональный модуль.

## Директивы Компилятор

'`#include`' и регулярно '`#define`' оба поддерживаются, но **не макрос** '`#define`', '`#pragma`' директивы и директивы условного включения ( '`#ifdef`', '`#ifndef`', '`#if`', '`#endif`', '`#else`' а также '`#elif`' ) являются также **не поддерживается**, '`#line`', '`#error`' и предопределенные макросы (например, '`_LINE_`', '`_FILE_`', '`_DATE_`', а также '`_TIME_`' ) являются также **не поддерживается**.

## Ардуино-языковые элементы

Все родные элементы языка Arduino поддерживаются, за исключением сомнительных '`goto`' инструкция (единственное разумное использование, которое я могу себе представить, это переход (к бесконечному циклу аварийного и безопасного отключения) в случае возникновения ошибки, с которой ваш программа не может иначе справиться)

## С / С ++ - языковые элементы

Сохраняющие бит "квалификаторы битовых полей" для членов в определениях структуры **не поддерживается**,

'`union`' является **не поддерживается**.

Странный "оператор запятой" **не поддерживается** (потому вы не можете выполнить несколько выражений, разделенных запятыми, когда обычно ожидается только одно выражение, например, в '`while()`' а также '`for( ; ; )`' конструкции).

## Функциональный модуль Шаблоны

Определяемый пользователем функциональные модули, который использует ключевое слово "`template`", чтобы позволить ему принимать аргументы типа "`generic`", **не поддерживается**,

## Эмуляция в реальном времени

Как отмечено выше, выполнение раз из множества различных отдельных возможных инструкций Arduino программа **не** смоделированы точно, так что для того, чтобы работать в режиме реального времени, ваш программа будет нуждаться в некотором доминировании '`delay()`' инструкция (по крайней мере, один раз в '`loop()`' ) или инструкция, которая естественным образом синхронизируется с изменениями уровня пин в реальном времени (например, '`pulseIn()`', '`shiftIn()`', '`Serial.read()`', '`Serial.print()`', '`Serial.flush()`' так далее.).

Видеть **Синхронизация** а также **Звуки** выше для более подробной информации об ограничениях.

## Примечания к выпуску - начиная с версии V2.5

### Исправления Ошибка

#### V2.8.2– сентябрь 2020

- 1) '`analogRead()`' не принимал 'A5' как действительный номер аналоговый пин.
- 2) Начиная с V2.6, переключатели 'PULSER' не всегда отображали выбранное состояние (но в остальном работали правильно).
- 3) Начиная с V2.4, при переключении с устойчивого 'HIGH' пин уровень до '`analogWrite(pin, 0)`' привело к тому, что изменение уровня 0 было пропущено.
- 4) Since V2.8, выполнение (без ошибок) всплывающие окна с ошибкой не справлялись с кодом проблемы основной момент.
- 5) Прерывание продолжающегося периодического переключения вывода пин с помощью '`digitalWrite()`' или '`digitalRead()`' вызвала потерю предыдущих переключений PWM на этом пин в окне Waveforms.
- 6) Исправлены проблемы с подключением двух '`INPUT`' пинов с '`JUMP`' устройство: 'I/O' устройство Изменения пин больше не просто появляются вместо переключки на пин (теперь они появляются на обоих пинов) и прикрепляются '`PULSER`' или '`FUNCGEN`' периодические сигналы теперь распространяются также на переключку pin.
- 7) Мгновенно закрыть '`PUSH`' Устройства не возвращались в открытое состояние, когда указатель мыши покидал устройство.

#### V2.8.1– июнь 2020

- 8) Дополнительные пустые строки, удаленные с помощью предпочтения автоматического форматирования, привели к тому, что первоначально выделенная строка в Изменить/Просмотреть была смещена вниз на количество пустых строк, которые были удалены над ней.
- 9) '`analogRead()`' принимал только полный номер цифровой пин.
- 10) '`class`', который содержал перегрузки функциональный модуль, отличающиеся только атрибутом '`unsigned`' аргумента функциональный модуль, может вызывать неправильную перегрузку функциональный модуль. Это затронуло LCD '`print(char/int/long)`', где вместо этого вызывался бы перегрузочный отпечаток '`unsigned`' base-10 функциональный модуль, и это заставило LCD '`printFloat()`' печатать '46' вместо десятичной точки '.'.
- 11) Автоматическое завершение вставки (после 'Enter') уже соответствующего встроенный не сработало после возврата назад, чтобы исправить ошибку при вводе в строку.
- 12) Устройство 'TFT' с пустым 'RS'; может вызвать сбой при выполнении.

#### V2.8.0– июнь 2020

- 1) Когда произошло предупреждение Анализировать (но не ошибка Анализировать), V2.7 может попытаться основной момент проблемную строку в неправильном буфере (в самом последнем '`#include`' вместо буфера), и это может вызвать молчаливый сбой (даже до появления всплывающего окна с предупреждением), если номер строки был за пределами этого '`#include`' буфера.
- 2) Полученные байты в более крупном мониторе окна 'I2CSLV', 'SPISLV', 'TFT', 'LCDI2C' и 'LCSPI' Устройства еще не все были зарегистрированы правильно (это не влияло на их функционирование).
- 3) Переменные типа '`unsigned char`' были повышены до типа '`int`' в арифметических выражениях, но неправильно поддерживал их '`unsigned`' статус, ведущий к ошибке '`unsigned`' Полученное выражение.



- 4) Изменение (или отключение) пин 'LED4' или '7SEG' устройство от первоначального действительного параметра пин не смогло отсоединить его верхние 3 пинах и могло привести к необъяснимым сбоям - кроме того, изменения, внесенные в V2.7 для согласования обработки всех 'LED' типы полностью сломали 'LED4' устройство.
- 5) Ошибка в изменениях, внесенных в Версию 2.6 для поддержки прерываний 'LOW', привела к тому, что прерывания 'FALLING', подключенные к пин 3, повредили определение изменения уровня прерывания на пин 2.
- 6) 'SoftwareSerial' неправильно отключал пользовательские прерывания во время каждого получаемого им символа.
- 7) Когда был загружен 'PROGIO' программа, содержащий ошибку Анализировать, ошибка была помечена, но программа уже был заменен на 'PROGIO' программа по умолчанию внутри монитора 'PROGIO' окно.
- 8) Начиная с версии V2.4, изменения пин на Пин 1 и Пин 0 не были замечены во время загрузки.
- 9) Заикливание операций чтения или записи на открытом SD файл привело к сбою последовательности выполнение, что привело к возможной внутренней ошибке UnoArduSim из-за глубины уровня сфера.
- 10) Попытка изменить 'MISO' пин на 'SD\_DRV' устройство повредила MOSI пин.
- 11) Все предыдущие версии не смогли предупредить, что с помощью ' `analogWrite()` ' на пинах 9 или 10 повредят 'Servo' синхронизация для всех активных 'Servo' Устройства.
- 12) Ввод другого числа поверх уже действующего 'EN' пин на 'LCD\_D4' устройство может вызвать сбой (частично набранное число вернет -1 для значения пин).

## **V2.7 – март 2020**

- 1) Когда была принята тема ОС ОС Окна (по умолчанию), **Панель с кодом** не показывал цветовую подсветку, представленную в V2.6 (вместо этого только серый основной момент, полученный в результате переопределения системы).
- 2) Версия 2.6 непреднамеренно прервала форматирование автоматической вкладки ' `switch()` ' построить.
- 3) Новая функция навигации стек вызовов, представленная в версии 2.6, показала **неверные значения** для местных переменные, когда он не находится внутри выполняемого в данный момент функциональный модуль, и произошел сбой с вложенными вызовами членов функциональный модуль.
- 4) ' `TFT::text()` ' работал, но ' `TFT::print()` ' функциональные модули не было (их просто заблокировали навсегда). Кроме того, ' `TFT::loadImage()` ' не удалось, если ' `Serial.begin()` ' было сделано раньше (что является нормальным случаем, и теперь требуется).
- 5) Версия 2.6 представила ошибка, который отображал неправильное значение для текущего и прошедшего байтов 'RX' для 'I2CSLV', 'SPISLV', 'TFT', 'LCDI2C' and 'LCDSPI' Устройства (и их монитора окна).
- 6) Изменения, сделанные в V2.4, вызвали **Выполнить | Анимация** выделение, чтобы пропустить много строк кода выполнил.
- 7) Начиная с версии 2.4, отмена 'SS\*' или 'CS\*' на 'I/O' устройство в инструкции, следующей сразу за ' `SPI.transfer()` ' приведет к тому, что устройство не сможет получить переданный байт данных. Кроме того, байт прием в ' `SPI_MODE1` ' а также ' `SPI_MODE3` ' не был помечен до начала следующего байта, отправленного мастером (и байт был полностью потерян, если устройство 'CS\*' был отменен ранее).
- 8) В новом ' `SPI_SLV` ' режим разрешен начиная с V2.4, ' `bval = SPI.transfer()` ' вернул только правильное значение для ' `bval` ' если передача байта уже завершена и ждет, когда ' `transfer()` ' назывался.
- 9) Поле редактирования 'DATA' на 'SPISLV' Устройства теперь получает значение по умолчанию 0xFF, когда больше нет байтов для ответа.
- 10) Состояние синхронизации LED было неправильным для 'PSTEPR' Устройства, имеющим более 1 микрошаг на полный шаг.
- 11) Электрический конфликты, вызванный реакцией 'I/O' Устройства на переходы на тактовом сигнале 'SPI', сигналом 'PWM' или ' `tone` ' сигнал, не сообщалось, и может привести к необъяснимым (поврежденным)

прием данных.

12) Когда интервал между прерываниями был слишком мал (менее 250 микросекунд), (ошибочное) изменение в V2.4 изменило синхронизация из встроенный функциональные модули, которые используют либо системные таймеры, либо циклы команд для генерации задержек (примеры каждого из них: `'delay()'` а также `'delayMicroseconds()'`). Последующее изменение в V2.5 вызвало неправильное выравнивание `'shiftOut()'` данные и тактовые сигналы, когда прерывание произошло между битами.

13) Принятие встроенный текста автозавершения функциональный модуль с помощью клавиши Enter не позволило удалить типы параметров из текста вставленного вызова функциональный модуль.

14) Загрузка нового (user-interrupt-толкнул) программа, когда ранее запущенный программа все еще имел ожидание прерывания, может вызвать сбой во время загрузки (из-за ошибочной попытки выполнение новой процедуры прерывания).

15) Автозаполнение члена Объект (доступно через 'ALT'-стрелка вправо) для объектов внутри `'#include'` файлы теперь доступны, как только их `'#include'` файл успешно разобранный.

16) Объявление функциональный модуль с непреднамеренным пробелом в имени параметра функциональный модуль вызвало нечеткое сообщение об ошибке Анализировать.

17) Когда выполнение остановлен в модуле, отличном от основного программа, Файл | Предыдущее действие не удалось включить.

18) Одиночные кавычки изогнутые скобки (`'{'` а также `'}'`) все еще считались (неправильно) как скобки сфера в Анализировать, а также запутанное автоматическое форматирование отступа табуляции.

19) `'OneWire::readBytes(byte* buf, int count)'` не удалось немедленно обновить отображаемое `'buf'` содержание в Панель с переменными.

20) Octal-latch 'OWISLV' Устройства показал выходные уровни пин, которые отставали от одной записи в регистр-защелку.

## **V2.6.0- | январь 2020**

1) ошибка, введенный в V2.3, привел к падению, когда добавленная кнопка Close использовалась в **Найти / Заменить** диалог (а не его **Выход** кнопка строки заголовка).

2) Если пользователь программа сделал `'#include'` другого пользователя файлы, **Сохранить** кнопка внутри **Изменить/Просмотреть** не удалось бы фактически сохранить измененный файл, если бы существовала ошибка Анализировать или Выполнение, помеченная внутри другого файл.

3) **Отмена** после **Сохранить** также может привести к путанице - по этим причинам **Сохранить** а также **Отмена** функциональность кнопок была изменена (см. **Изменения и улучшения**,

4) Несбалансированные скобки внутри `'class'` определение может вызвать зависание начиная с V2.5.

5) Прямое логическое тестирование на `'long'` возвращенные значения `'false'` если ни один из 16 младших битов не был установлен.

6) UnoArduSim отмечал ошибку, когда указатель переменная был объявлен как цикл переменная внутри скобок `'for()'` заявление.

7) UnoArduSim запрещал логические тесты, сравнивающие указатели с `'NULL'` или `'0'`,

8) UnoArduSim запрещал арифметику указателей, включающую целое число переменная (были разрешены только целочисленные константы).

9) Прерывания, установленные с помощью `'attachInterrupt(pin, name_func, LOW)'` был воспринят только на **переход** в `'LOW'`,

10) При использовании более одного 'I2CSLV' устройство неадресный ведомый может интерпретировать более поздние данные шины как соответствующие его адресу шины (или глобальному вызову 0x00), и, таким образом, ложно сигнализировать ACK, повреждая уровень ACK шины и вешая `'requestFrom()'`.

11) Передача числового значения `'0'` (или `'NULL'`) в качестве аргумента функциональный модуль для указателя в вызове функциональный модуль теперь разрешено.

12) Уровень отступа табуляции после вложения `'switch()'` конструкции было слишком мелким, когда выбор 'auto-indent formatting' **Конфигурировать | Настройки** использовался.

- 13) Вычитание двух совместимых указателей теперь приводит к типу `'int'`,
- 14) UnoArduSim ожидал пользовательский конструктор по умолчанию для члена объекта, даже если он не был объявлен как `'const'`,
- 15) На перерыве выполнение, нарисованная позиция 'STEPR', 'SERVO' или 'MOTOR' Двигатель может отставать на 30 миллисекунд от своего фактического последнего вычисленного положения.
- 16) `'Stepper::setSpeed(0)'` вызывал сбой из-за деления на ноль.
- 17) Одна линия `'if()'`, `'for()'`, а также `'else'` конструкции больше не вызывают слишком много вкладок авто-отступа.

## **V2.5.0– октябрь 2019**

- 1) А ошибка введен в V2.4 сломал инициализацию карты 'SD' (вызвал сбой).
- 2) Использование подсистемы 'SPI' в новом `'SPI_SLAVE'` режим работал неправильно в `'SPI_MODE1'` а также `'SPI_MODE3'`,
- 3) Всплывающие окна автозаполнения (согласно запросу 'ALT-right=arrow') были исправлены для функциональные модули, имеющего параметры объект; список всплывающих окон теперь также включает унаследованные (базовые) члены класса, а автозаполнения теперь также отображаются для `'Serial'`,
- 4) Начиная с версии 2.4, возвращаемое значение для `'SPI.transfer()'` а также `'SPI.transfer16()'` было неверно, если во время этой передачи была запущена подпрограмма прерывания пользователя.
- 5) В версии 2.4 быстрые периодические сигналы были показаны как имеющие большую длительность, чем их фактическая длительность, очевидная при просмотре с большим увеличением.
- 6) Конструктор `'File::File(SdFile &sdf, char *fname)'` не работал, так `'File::openNextFile()'` (который опирается на этот конструктор) также не работает.
- 7) UnoArduSim неправильно объявлял ошибку Анализировать на объект-переменные, а объект возвращал функциональные модули, объявленный как `'static'`.
- 8) Заявления о назначении с `'Servo'`, `'Stepper'`, или `'OneWire'` объект переменная на LHS, и объект - возвращение функциональный модуль или конструктора на RHS, вызвало внутреннюю ошибку числа связанных объектов, что приводит к возможному врезаться.
- 9) Булевы тесты на объектов типа `'File'` всегда возвращались `'true'` даже если файл не был открыт.

## Чанг ES / Улучшения

### V2.8.2- сентябрь 2020

- 1) Чтобы избежать путаницы, UnoArduSim теперь позволяет '`< >`' использовать треугольные скобки и двойные кавычки взаимозаменяемо вокруг названия файл в '`#include`' операторы как для локального пользователя, так и для '3rdParty' файлы.
- 2) В '**MOTOR**' устройство теперь может принимать три скрытых значения от '**IODevs.txt**' файл: 'F' или 'B' (Freewheel- выбегом или режим торможения), затем постоянный момент нагрузки в процентах от момента остановки, затем момент нагрузки инерция как кратное '**MOTOR**' инерция ротора.
- 3) UnoArduSim теперь принимает '`long int`' как синоним '`long`'.
- 4) UnoArduSim теперь выдает одноразовое предупреждение для использования '`volatile`' когда глобальная переменная модифицируется программой прерывания пользователя.

### V2.8.0- июнь 2020

- 5) Добавлен новый 'Mega2560' плата **Настройки** опция (Плата номер == 10). Это особенности еще много 'I/O' пинах, еще 4 внешних прерывания (пинах 18, 19, 20, 21), еще три '**HardwareSerial**' порты (на пинах 22-27), а объем оперативной памяти увеличен с 2 КБ до 8 КБ.
- 6) 'SFTSER' устройство был переименован в 'ALTSER' (поскольку теперь он также может использоваться с 'Serial1', 'Serial2' и 'Serial3').
- 7) Щелчок внутри поля редактирования устройство пин теперь выбирает целое число, чтобы упростить ввод, а нажатие внутри поля редактирования номера Устройства в Конфигурировать | 'I/O' Устройства делает то же самое.
- 8) Добавлена оранжевая подсветка соответствующей строки источника для всплывающих окон с предупреждениями Анализировать и Выполнение.
- 9) Автоформатирование теперь удаляет пустые строки, которые появляются после строк ключевых слов или после предшествующих пустых строк.
- 10) Добавлена поддержка '`digitalPinToInterrupt()`' звонки.
- 11) Классы теперь всегда получают конструктор по умолчанию, если у них нет указанного пользователем конструктора (даже если у них нет базового класса или членов объект).

### V2.7 марта 2020 г.

- 1) В дополнение к текущей строке кода (зеленый, если готов к запуску, красный, если ошибка), UnoArduSim теперь поддерживает для каждого модуля последнюю закодированную строку, нажатую пользователем или стековую навигацию (выделена темным оливковым фоном), делая проще установить и найти временные линии точка останова (теперь разрешена одна для каждого модуля, но на 'Run-To' действует только одна из отображаемых в данный момент модулей).
- 2) Добавлен новый 'I/O' Устройства (и поддерживающий сторонний код библиотеки), включая 'SPI' и 'I2C' **Порты расширения** 'SPI' и 'I2C' **Мультиплексор LED** Контроллеры и дисплеи (LED, массивы, 4-буквенно-цифровые и 4-цифра или 8-цифра 7-сегментные дисплеи).
- 3) '**wire**' операции больше не запрещены изнутри подпрограмм прерывания пользователя (это поддерживает внешние прерывания от 'I2C' Порт Расширения).
- 4) Сигналы Цифровой теперь показывают промежуточный уровень (между '**HIGH**' а также '**LOW**') когда пин не является толкнул.
- 5) Чтобы избежать путаницы при переходе через один '`SPI.transfer()`' В соответствии с инструкциями, UnoArduSim теперь гарантирует, что подключенный 'I/O' Устройства теперь получит свой (с логической задержкой) окончательный фронт тактового сигнала 'SCK' до возврата функциональный модуль.
- 6) При автоматическом форматировании вкладок **предпочтение** включен, набрав закрывающий изогнутая

скобка '}' в **Изменить/Просмотреть** теперь вызывает переход к позиции отступа вкладки соответствующего открытия-изогнутая скобка '{' партнер.

7) **Переформатировать** кнопка была добавлена к **Изменить/Просмотреть** (вызвать немедленное автоматическое переформатирование отступа табуляции) - эта кнопка активна только в том случае, если включена опция автоматического отступа.

8) Более четкое сообщение об ошибке теперь появляется, когда ключевое слово префикса (например, 'const', 'unsigned', или 'PROGMEM') следует за идентификатором в объявлении (он должен предшествовать идентификатору).

9) Инициализированному глобальному переменной, даже если он никогда не используется позже, теперь всегда назначается адрес памяти, и поэтому он будет виден.

## **V2.6.0 январь 2019**

1) добавленной **Character-LCD** дисплей Устройства, имеющий 'SPI', 'I2C' и 4-бипараллельный интерфейс. Поддерживающий исходный код библиотеки был добавлен в новую папку установки 'include\_3rdParty' (и может быть доступен с помощью обычного '#include' директива) - пользователи могут вместо этого выбрать вместо записи свои собственные функциональные модули к толкнул ЖК устройство.

2) **Панель с кодом** улучшена подсветка с отдельными цветами основной момент для готовой строки кода, для строки кода ошибки и для любой другой строки кода.

3) **Найти** меню и панель инструментов Действия 'func' (предыдущий вверх и следующий вниз) больше не переходят к предыдущей / следующей стартовой линии функциональный модуль, а вместо этого теперь поднимаются (или опускается) стек вызовов, выделяя соответствующую кодовую строку в вызывающем (или вызываемом) функциональный модуль, соответственно, где **Панель с переменными** содержимое откорректировано, чтобы показать переменные для функциональный модуль, содержащего текущую выделенную строку кода.

4) Чтобы избежать путаницы, **Сохранить** сделано внутри **Изменить/Просмотреть** вызывает немедленное **Компилировать** если Сохранить был успешным, используя последующий Отмена или Выход теперь будет только возвращать текст на этот последний сохраненный текст.

5) Добавлен импульсный вход Шаговый двигатель ('PSTEPR') с входами 'STEP' (импульсный), 'EN\*' (активировать) и 'DIR' (направление), а также настройка микро-шагов на шаг (1,2,4,8 или 16),

6) И 'STEPR', и 'PSTEPR' Устройства теперь имеют 'sync' LED (ЗЕЛЕНЫЙ для синхронизации или КРАСНЫЙ при отключении на один или несколько шагов).

7) У 'PULSER' Устройства теперь есть выбор между микросекундами и миллисекундами для 'Period' и 'Pulse'.

8) Автозаполнения Встроенный-функциональный модуль больше не сохраняют тип параметра перед именем параметра.

9) При переключении обратно на предыдущий **Панель с кодом**, его ранее выделенная строка теперь снова выделена.

10) В качестве помощи для установки временной точки останова, используя Отмена или Выход из **Изменить/Просмотреть** оставляет основной момент в **Панель с кодом** на строке, последней посещенной курсором в **Изменить/Просмотреть**,

11) Пользовательский (или сторонний) 'class' теперь разрешено использовать 'Print' или 'Stream' как его базовый класс. В поддержку этого была добавлена новая папка 'include\_Sys' (в папке установки UnoArduSim), которая предоставляет исходный код для каждой базы 'class', В этом случае звонки на такие базы-'class' функциональные модули будет обрабатываться идентично коду пользователя (в который можно войти), а не как встроенный функциональный модуль, в который нельзя войти (например, 'Serial.print()').

12) Автозаполнения Member-функциональный модуль теперь включают имя параметра **вместо** его тип.

13) UnoArduSim Анализировать теперь позволяет имени объект в объявлении переменная предваряться его необязательным (и соответствующим) **'struct'** или ключевое слово **'class'**, за которым следует **'struct'** или **'class'** имя.

## **V2.5.0 Oct 2019**

- 1) Добавлена поддержка **'TFT.h'** библиотека (за исключением **'drawBitmap()'** ) и добавил связанный **'TFT' 'I/O'** Устройство (128 на 160 пикселей). Обратите внимание, что во избежание чрезмерных задержек в реальном времени во время больших **'fillXXX()'**  перевод, часть передач **'SPI' в середине заполнения** будет отсутствовать в автобусе **'SPI'**.
- 2) Во время больших переводов файл через **'SD'**, часть передач **'SPI'** в середине последовательности байтов также будет отсутствовать в шине **'SPI'**,
- 3) Снижение **'Stream'** - использование служебных байтов, чтобы **'RAM free'** значение более точно соответствует компиляции Arduino.
- 4) UnoArduSim теперь предупреждает пользователя, когда **'class'** имеет несколько членов, объявленных в одной строке объявления.
- 5) Использование **'File | Save As'** теперь устанавливает текущий каталог, который сохранен в каталог.
- 6) Два пропавших без вести **'remove()'**  член функциональные модули был добавлен в **'String'** учебный класс.
- 7) UnoArduSim теперь запрещает базовые вызовы конструктора в конструкторе функциональный модуль прототип, если непосредственно не следует полное определение тела функциональный модуль (чтобы согласиться с Arduino компилятор).
- 8) издание время перехода Форма волны цифровой была уменьшена для поддержки визуализации быстрых сигналов **'SPI'** при максимальном увеличении.
- 9) UnoArduSim теперь позволяет объявлять некоторые конструкторы **'private'** или **'protected'** (для внутреннего использования класса).