Руководство по эксплуатации Tuner Studio

Версия прошивки 0.3

18 августа 2023 года

Генри Райт

Содержание

[1 Сводка 2](#_Toc143372379)

[2 Быстрый старт 2](#_Toc143372380)

[2.1 Подключение 2](#_Toc143372381)

[2.2 Считывание данных порта и управляющих выходов 3](#_Toc143372382)

[2.3 Вставка вашего собственного кода 5](#_Toc143372383)

[2.4 Добавление переменной измерения (отправляется в Tuner Studio) 5](#_Toc143372384)

[2.5 Добавление калибровочной переменной (сохраненной в EEPROM) 8](#_Toc143372385)

[3 Документация по функциям 9](#_Toc143372386)

[3.1 Биты выполнения задачи и предупреждения 9](#_Toc143372387)

[3.2 Фильтр нижних частот 9](#_Toc143372388)

[3.3 CAN 10](#_Toc143372389)

[3.3.1 Обзор CAN 10](#_Toc143372390)

[3.3.2 Реализация 10](#_Toc143372391)

[3.3.3 МОЖЕТ Транслировать 11](#_Toc143372392)

[3.3.4 МОЖЕТ получать сообщение 11](#_Toc143372393)

# Краткие сведения

Эта программа предназначена для работы в системах на базе Arduino, запрограммированных через Arduino IDE, таких как Arduino Mega2560. Он обеспечивает связь с Tuner Studio (TS), инструментом калибровки и сбора данных, доступным по адресу [https://www.tunerstudio.com /](https://www.tunerstudio.com/)

Основная функциональность заключается в том, чтобы:

1. Определите переменные, которые сохраняются во внутренней памяти (EEPROM) Arduino и вызываются из нее.
2. Отправляйте переменные из памяти в TS и получайте новые данные при изменении переменных в TS.
3. Запишите (запишите) эти переменные в EEPROM.
4. Отправляйте переменные измерения с определенной скоростью в TS для отображения и протоколирования.
5. Получать команды для выполнения определенных функций на Arduino. т.е. включать или выключать выход.
6. Обеспечьте систему синхронизации на основе задач для выполнения задач с обратной связью, если эти задачи переполнены.
7. Реализуйте простую реализацию широковещательной передачи по CAN\_BUSи приема объектов сообщений с использованием приемопередатчика MCP2515 через spi.

Этот код обеспечивает основные принципы коммуникации и формирует основу для функциональности, созданной пользователем, которая будет добавлена в этот код.

The Последнюю версию можно найти в подкаталоге ->release.

Документацию можно найти во вложенном каталоге ->docs.

Последнюю версию этой программы можно найти на github: <https://github.com/HWright9/TunerStudioOS>.

# Быстрый тарт

## Подключение

Краткий обзор, позволяющий приступить к компиляции и запуску программыram. Подробности приведены в разделах ниже.

1. Вам нужен Arduino Mega2650 Uno или Nano. Просто на данный момент запустите его без каких-либо экранов или подключений, отличных от USB к ПК.
2. Скачать Tuner Studio можно с [https://www.tunerstudio.com /](https://www.tunerstudio.com/). Для разработчиков наиболее полезной будет платная версия, однако программа будет работать и с бесплатной версией.
3. Загрузите и установите последнюю версию Arduino IDE [https://www.arduino.cc /](https://www.arduino.cc/).
4. Загрузите всю папку с программой с github <https://github.com/HWright9/TunerStudioOS>. Я рекомендую разместить его в Documents -> Arduino directory.
5. Либо скомпилируйте и загрузите программу в Arduino с помощью IDE, либо используйте файл release.hex в меню -> release для прошивки контроллера.
   1. Обратите внимание, что при повторном использовании старого Arduino лучше всего запустить эскиз EEPROM\_clear из примеров, чтобы убедиться, что EEPROM не содержит каких-либо оставшихся данных.
6. Откройте tuner studio и запустите новый проект. При появлении запроса найдите файл release.ini в -> папка release.
   1. Убедитесь, что вы выбрали размер EEPROM в конфигурации проекта. 1 КБ для UNO или Nano и 8 КБ для Mega.
7. Откройте базовуюстроку.msq, чтобы загрузить начальные переменные в Tuner Studio.
8. Подключитесь к контроллеру через последовательный порт. Битрейт равен 115200. Com-порт будет таким же, как используется с Arduino IDE. (Подсказка: вы должны быть отключены от Tuner Studio, чтобы прошить новый код на Arduino).
9. При появлении запроса запишите все переменные в контроллер.
10. Убедитесь, что прожиг завершен.
11. Сбросьте настройки контроллера.
12. Теперь вы должны быть подключены.

Ваш экран должен выглядеть так, как показано на рисунке ниже:

A picture containing device, measuring instrument, gauge, clock

Description automatically generated

Рисунок Основной Кластер датчиков по умолчанию

## Считывание данных порта и управляющих выходов

Кнопка тестирование оборудования имеет несколько опций. Откройте их все, и вы увидите

1. Состояние всех цифровых портов, являются ли они входными или выходными. Горит = зеленый = Логический максимум (5 В).
   1. Примечание1. У Mega всего 54 порта, хотя показано 6 из4.
   2. Примечание 2. Нумерация портов совпадает с нумерацией, указанной на корпусе Arduino.
2. Вводы Analog, как показано на графике в реальном времени. Они также доступны в виде датчиков на главном экране.
3. Раздел "Тестовые порты", в котором вы можете переопределить статус цифрового порта.

A close-up of a computer screen

Description automatically generated with low confidence

Рисунок Пункты меню при тестировании оборудования

Если вы перейдете в раздел Тестовые порты. Вы должны быть в состоянии “включить тестовый режим”. Если он выделен серым цветом, сначала вам нужно включитьфайл “Разрешить команды тестирования оборудования” в системных настройках.

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок Системные настройки

Как только это заработает, вы сможете вручную переопределить каждый порт. Обратите внимание на следующее:

1. Порты, определенные в коде как входные, изменят значение, видимое программой, но не изменят физический порт.
2. Порты, определенные как ВЫХОДНЫЕ, изменят физическое состояние порта, а также то, что будет видно программе.
3. Помните, что некоторые порты используются для других функций, таких как последовательная связь, и вы, возможно, не сможете их изменить или их изменение приведет к ошибкам в других функциях.
4. Ваш контроллер может фактически не поддерживать все показанные порты и аналоговые каналы.

A screenshot of a computer

Description automatically generated

Рисунок Вид тестирования оборудования и переопределения

## Вставка вашего собственного кода

Код наследует все обычные библиотеки Arduino. В начале работы лучше всего открыть весь проект в удобном редакторе C, таком как notepad++. Откройте всю папку целиком как рабочую область, чтобы вы могли быстро перемещаться между файлами.

Открыть пользовательские функции.ino

Напишите простую функцию, такую как

A screenshot of a computer program

Description automatically generated with low confidence

Убедитесь, что вы добавили прототип функции в файл userfunctions.h. Затем вы переходите к файлу TunerStudio\_OS\_Dev.ino и находите правильную скорость выполнения задачи для вызова функции.,,

Примечание-ы:

1. Избегайте написания большого количества инструкций code и if в планировщиках задач в TunerStudio\_OS\_Dev.ino, это быстро приводит к беспорядку. Гораздо лучше создать функцию, а затем просто вызвать ее из appopriate FUNC\_XXXmsTask.
2. Обратите внимание, как мы используем setDigitalPort и readDigitalPort вместо digitalRead и digitalWrite. Эта оболочка - это то, что позволяет командам тестового порта работать.

A picture containing text, line, font, screenshot

Description automatically generated

Когда вы закончите. Сохраните свою работу и скомпилируйте код в Arduino. Затем загрузите и запустите новую программу.

## Добавление переменной измерения (отправляется в Tuner Studio)

Чтобы добавить новую переменную измерения, вам нужно добавить сигнал в структуру Out\_TS. Вы делаете это, редактируя Out\_TS\_t typedef в globals.h

Допустимыми типами переменных измерения являются:

Битовое поле (минимум 8 байт), U08, S08, U16, S16, U16, S32 или F32.

Как только вы добавите его сюда, высможете ссылаться на него в коде, используя префикс Out\_TS.Vars.new\_variable\_name

Важно убедиться, что порядок и типы устройств, указанные здесь, точно соответствуют тому, что указано в файле TunerStudioOS.ini, иначе у вас возникнут проблемы со связью.

A close-up of a text

Description automatically generated

Рисунок Out\_TS\_t Пример Typedef

В файле TunerStudioOS.ini найдите раздел [Выходные каналы].

Убедитесь, что вы вставляете свою переменную точно в том же порядке, что и в структуре Out\_TS\_t. Используйте переменную nextOffset для автоматической установки следующего соответствующего байтового адреса. При определении битового поля вы должны использовать последнеесмещение, чтобы предотвратить увеличение адреса для каждого бита внутри байтового адреса.

Также возможно точно указать адрес каждого байта, однако перенумеровывать их все становится утомительным, если во время разработки вы хотите вставить новую переменную между существующими переменными.

Следуйте примерам в исходном коде.

Tuner Studio также необходимо знать, сколько байтов следует ожидать с помощью переменной ochBlockSize. Размер - это размер структуры Out\_TS\_t, если вы знаете, сколько байтов вы добавили, вы можете увеличить это число на это количество байтов. Если вы не уверены, то этот номер передается в Tuner studio с параметром ochBlockSizeSent, однако его необходимо обновить вручную в разделе OutputChannels.

Если у вас возникли проблемы со связью после обновления выходных каналов, причиной обычно является неправильный размер блока.

A screenshot of a computer

Description automatically generated

Рисунок . Пример выходных каналов

Окно системных настроек содержит правильный размер переменной ochBlockSize, отправленной с Arduino.

A screenshot of a computer

Description automatically generated

Рисунок . Системные настройки, показывающие размер блоков и страницы. Приведенные здесь значения рассчитаны и доступны при подключении к Arduino

Как только файл .ini и код будут обновлены, загрузите код в Arduino, а затем перезагрузите файл TunerStudio.ini. TS должен автоматически подключиться.

Если после обновления возникают проблемы с coms, вы можете использовать журнал отладки coms в TS для получения дополнительной информации. В большинстве случаев это проблема с неправильным значением ochBlockSize not being correct или несоответствующим порядком переменных.

## Добавление калибровочной переменной (сохраненной в EEPROM)

Переменная калибровки существует в оперативной памяти, а также синхронизируется с EEPROM и Tuner Studio. Процесс заключается в следующем:

Когда Arduino включается, EEPROM копируется в оперативную память.

Когда Tuner Studio включится, он запросит отправку копии RAM по последовательной ссылке. Затем Tuner studio сравнивает эти данные со своей собственной копией параметров калибровки.

Если переменную необходимо обновить, Tuner Studio отправляет переменную в RAM Arduino, но данные фактически не сохраняются в EEPROM до тех пор, пока не будет выдана команда записи. Затем Arduino записывает любые изменения в переменных оперативной памяти в EEPROM.

Arduino

EEPROM

ОЗУ

Студия тюнера

(.msq)

При инициализации контроллера

С помощью команды Записать

TS Запрашивает данные by pвозраст при подключении

TS Отправляет данные по мере изменения переменных

Последовательное соединение

ПК

Важно отметить, что запись в EEPROM может занимать 3,6 мс на байт, поэтому 8 КБ может занимать до 30 секунд. Если связь с TS прервана, TS сбросит Arduino при попытке повторного подключения. Таким образом, обновления EEPROM должны выполняться асинхронно. Кроме того, Tuner Studio разбивает EEPROM на страницы, чтобы сделать ее более управляемой. Страницы могут быть любого размера, но для синхронизации и отправки данных страниц большего размера требуется больше времени.

Если вам нужно изменить размер страницы, начните с файла storage.h. Помните, что фактический размер страницы никогда не должен превышать определенный размер страницы, иначе переменные будут перезаписывать друг друга, и TS будет выдавать ошибки.

Переменные в EEPROM могут быть одиночными битами, скалярами или многомерными массивами.

Чтобы определить новую калибровочную переменную , выполните поиск правильного определения страницы в globals.h.

Добавьте свою переменную с правильным типом данных. Обратите внимание, что при использовании переменной, следующей за некоторыми битами, биты всегда являются частью целого байта. Следующей скалярной или массивной переменной будет nextOffset.

После определения вы можете ссылаться на свою новую переменную в коде с помощью configPageX. имя\_новой\_ переменной.

A close-up of a text

Description automatically generated

Рисунок Пример определения страницы 1.

В файле TunerStudioOS.ini найдите раздел страница = X. С X, соответствующим странице, на которую вы добавили свою переменную. Аналогично разделу [OutputChannels] вы должны добавить свою переменную в точно таком же порядке и с точно такими же единицами измерения.

nextOffset и lastOffset также работают здесь, как и в OutputChannels.

# Документация по функциям

Код предоставляет ряд функций, помогающих в выполнении общих задач управления и измерений. При необходимости их можно удалить.

## Биты выполнения задачи и предупреждения

Код реализует простой планировщик задач.

Убедитесь, что выполнение вашего кода не занимает слишком много времени. Любой полный цикл не может быть длиннее 5 мс, иначе задача на 5 мс пропустит шаг выполнения.

## Фильтр нижних Частот

uint16\_t lowPassFilter\_u16(uint32\_t input, uint8\_t alpha, uint32\_t prior);

Функция реализует фильтр нижних частот или фильтр задержки, используя новый ввод, предыдущее входное значение и “альфа-значение” от 1 до 255, которое устанавливает фильтрацию на выходе. Высокие значения альфа-сигнала увеличивают фильтрацию, при этом 128 составляют 50% фильтрации. Код гарантирует, что выходные данные будут сходиться к входным даже при очень высокой фильтрации, при самом тяжелом фильтре выходные данные будут приближаться к входным со скоростью одного бита за цикл.

Уравнение имеет вид:

Выход = Предыдущий \* (1-альфа) + вход \* альфа

Пример фильтрации показан ниже.

A screen shot of a computer

Description automatically generated with low confidence

Рисунок Пример фильтра нижних частот. Фильтруется красный сигнал, белый - входной. Альфа приблизительно 200

## МОЖЕТ

### МОЖЕТ Обзор

В операционной системе предусмотрена простая реализация широковещательной сети Aera Network (CAN) контроллера Aera. В этой сети каждый узел передает сообщения на шину с определенной скоростью. Отправитель не знает, какие модули получают сообщения.

Каждый модуль получает все сообщения и сравнивает их с несколькими заранее определенными парсерами получения сообщений. Анализатор декодирует сообщение и загружает данные из сообщения в переменные для внутреннего использования.

Этот вид обмена сообщениями обычно используется в автомобилях для передачи данных на дисплеи приборной панели или в контроллер передачи.

В этой системе важно определить, больше не принимается ли ожидаемое сообщение n, и, по умолчанию, выполнить “безопасное” состояние. В противном случае последнее полученное значение будет сохранено навсегда.

### Реализация

Код ожидает контроллер MCP2515, подключенный через интерфейс SPI. Распиновка отличается в зависимости от платы. Большинство контактов и инициализации настраиваются с помощью настроек в Tuner Studio. Фактические сообщения CAN жестко запрограммированы в canbus.ino

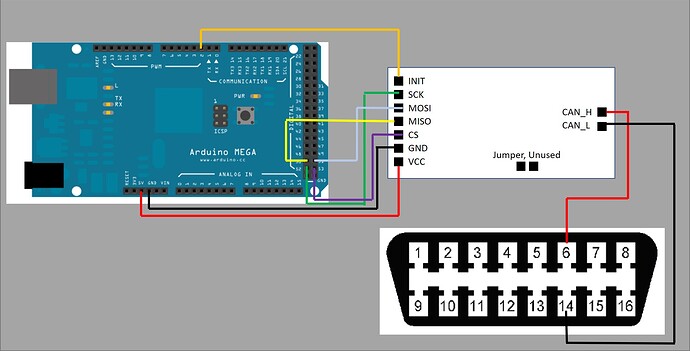


Рисунок Пример подключения MCP2515 на Arduino Mega

Если включен режим CAN. Arduino пытается инициализировать модуль и в случае успеха начинает трансляцию. В случае неудачи, код отправит биты состояния в Tuner Studio, которые можно отобразить в нижней части экрана.



Если у вас возникли ошибки шины CAN. Проверьте следующее:

1. Все ли узлы в сети настроены на одну и ту же частоту?
2. Есть ли у вас подходящие оконечные резисторы? На каждом конце шины должно быть по 120 Ом. На большинстве плат MCP2515 предусмотрено включение оконечного резистора.

### МОЖЕТ Транслировать

CAN broadcast использует простые функции, вызываемые из основного планировщика задач.

Пример: аннулирование canBroadcast\_5ms(void)

The message Переменные сообщения затем могут быть проанализированы в объект canmsg[] и отправлены с соответствующей скоростью.

### МОЖЕТ получать сообщение

CAN receive использует вывод прерывания от MCP2515, чтобы сообщить Arduino, что данные CAN ожидают. Начиная с реализации 0.3, это фактически не прерывает код для получения сообщения, этот вывод опрашиваетсяed из основного цикла.

Если пользователь реализует прерывание an, не забывайте, что переменные должны быть объявлены изменчивыми при использовании в ISR, а также что интенсивный трафик шины CAN может привести к перегрузке контроллера.

Как только сообщение получено, идентификатор сообщения проверяется по каждому определенному объекту message. Затем сравнивается длина в объекте, чтобы убедиться, что было получено полное сообщение. Если обе эти проверки пройдены, сообщение анализируется и счетчик таймаута сбрасывается.