

Свежие комментарии

- Семен к записи [AVR Урок 15. Внутренняя энергонезависимая память EEPROM. Часть 4](#)
- Стас к записи [STM Урок 64. HAL. LTDC. Часть 2](#)
- Семен к записи [AVR Урок 15. Внутренняя энергонезависимая память EEPROM. Часть 4](#)
- Семен к записи [AVR Урок 15. Внутренняя энергонезависимая память EEPROM. Часть 4](#)
- Вова к записи [AVR Урок 16. Интерфейс TWI \(I2C\). Часть 2](#)

Форум. Последние ответы

- alexander в [Программирование МК STM32](#)
5 дн., 19 час. назад
- Narod Stream в [Программирование МК STM32](#)
4 нед., 1 день назад
- Zandy в [Программирование МК STM32](#)
4 нед., 1 день назад
- Narod Stream в [Программирование МК STM32](#)
1 месяц, 1 неделя назад
- Narod Stream в [Программирование МК STM32](#)
1 месяц, 1 неделя назад

Февраль 2018

Пн	Вт	Ср	Чт	Пт	Сб	Вс
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				
« Янв						

Архивы


- Февраль 2018
- Январь 2018
- Декабрь 2017
- Ноябрь 2017
- Октябрь 2017
- Сентябрь 2017
- Август 2017
- Июль 2017

Главная > Программирование AVR > AVR УРОК 39. Акселерометр LSM6DS3. Часть 1

AVR УРОК 39. Акселерометр LSM6DS3. Часть 1

Posted on Январь 16, 2017 by Narod Stream
Опубликовано в [Программирование AVR](#)
— Нет комментариев ↓


Яндекс.Директ



Катушки Тесла оптом от производителя

Стань эксклюзивным представителем в своем городе компании Tesla Technologies.
tesla-technologies.ru Адрес и телефон

Яндекс.Директ



Частотные регулируемые приводы тут!

Только оригинал: Eaton, Vacon, ABB. Цены ниже китайских аналогов! Всегда в наличии!

Урок 39 Часть 1

Акселерометр LSM6DS3

Сегодня мы продолжим знакомство, которое мы начали в [уроке 38](#), с интересной отладочной платой, выполненной на базе микроконтроллера **Atmega 328p**.

На прошлом занятии мы смогли подключить данную плату к программатору и использовать её как понтонную отладочную плату для программирования в среде Atmel Studio.


Сегодня мы усложним свой проект и попробуем подключить к данной плате хороший акселерометр **LSM6DS3**, который также ещё является и гироскопом. Но мы пока будем работать с данным датчиком только как с акселерометром, а как с гироскопом мы попробуем поработать с ним возможно в последующих занятиях, либо вы сможете изучить этот вопрос самостоятельно, так как все тонкости подключения и общения с датчиком мы в данном занятии обязательно рассмотрим. С данным датчиком мы уже работали,

искать здесь ...

Фильтровать

Катушки Тесла

Надежное оборудование для вашего Шоу



каталог товаров >

Заходите на канал Narod Stream

- [Июнь 2017](#)
- [Май 2017](#)
- [Март 2017](#)
- [Февраль 2017](#)
- [Январь 2017](#)
- [Декабрь 2016](#)
- [Ноябрь 2016](#)

подключая его к микроконтроллеру STM32F401 и у нас это успешно получилось. Поэтому обязательно данные уроки советую посмотреть для того, чтобы вам стала немного понятнее организации структуры данного датчика.

Тем не менее в этом уроке я характеристики данного датчика также приведу:

Диапазон показаний $\pm 2g/\pm 4g/\pm 8g/\pm 16g$;

Чувствительность 0.061 – 0.49 mg/digit;

Отклонение от нуля ± 40 mg.

Скорость считывания данных 12,5 Гц – 6,66 кГц.

С некоторыми остальными показателями, регистрами, значениями и другими тонкостями акселерометра мы познакомимся в ходе его программирования.

Проект мы также создадим для контроллера Atmega 328p и назовём его **Accel_LSM6DS3**.

Исходный код в main.c удалим полностью и скопируем готовый из проекта прошлого занятия LIGHTS.

Полностью уберём код из бесконечного цикла

```
while (1)
{
}
```

Создадим заголовочный файл **main.h** обычным образом и весь код с подключением всех заголовочных файлов с добавлением ещё файлов, которые нам в последствии пригодятся, а также с объявлением частоты тактирования, из файла main.c перенесём в данный файл

```
#ifndef MAIN_H_
#define MAIN_H_
//
#define F_CPU 16000000L
//
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdlib.h>
//
#endif /* MAIN_H_ */
```

Ну и, соответственно, подключим данный файл в файле **main.c**

```
#include "main.h"
//
int main(void)
```

Для того, чтобы пользоваться данным датчиком, нам недостаточно его настроить и считывать показания. Данные показания ещё нужно как-то отследить и увидеть. На помощь нам придёт интерфейс **USART**. Данный интерфейс в контроллере **Atmega328** организован приблизительно таким же образом, как и в **Atmega8**, с некоторыми незначительными отличиями. Отличия касаются в наименованиях регистров в связи с тем, что в роли

Ср-ва измерений/ контроля



Продажа и поверка.
Измерительные приборы,
геодезия, КИПиА. Низкие цены!



Рубрики

- [1-WIRE](#) (3)
- [ADC](#) (6)
- [DAC](#) (4)
- [GPIO](#) (26)
- [I2C](#) (19)
- [SPI](#) (13)
- [USART](#) (8)
- [Программирование AVR](#) (131)
- [Программирование PIC](#) (8)
- [Программирование STM32](#) (217)
- [Тесты устройств и аксессуаров](#) (1)

		141 653
		14 731
33 ДЕНЬ		33 640
		4 551
07 ДНЕЙ		5 501
		1 110
24 ЧАСА		2 220
		628
СЕГОДНЯ		82
		40
		НА ПИШУ

второго USART в Atmega328 может выступать интерфейс SPI. Тем не менее для начала мы скопируем файлы **usart.h** и **usart.c** из проекта **урока 14** под названием **Test12**. Также не стоит забывать подключить файл **usart.h** в файле **main.h**

```
#include <stdlib.h>
//
#include "usart.h"
//
```

Теперь зайдём в файл **usart.c** и внесём там ряд изменений в свете требований МК Atmega328, иначе у нас при сборке возникнет масса ошибок. Все имена данных регистров имеются в технической документации

```
#include "usart.h"
//
void USART_Init( unsigned int ubrr )//Инициализация модуля USART
{
    //Зададим скорость работы USART
    UBRR0H = (unsigned char)(ubrr>>8);
    UBRR0L = (unsigned char)ubrr;

    UCSR0B=(1<<RXEN0)|( 1<<TXEN0); // Включаем прием и передачу по USART
    UCSR0B |= (1<<RXCIE0); //Разрешаем прерывание при передаче
    UCSR0A |= (1<<U2X0); // Удвоение частоты
    UCSR0C = (1<<UCSZ01|
(1<<UCSZ00);// асинхронный режим (UMSEL=0), без контроля четности (UPM1=0 и UPM0=0),
    //1 стоп-бит (USBS=0), 8-бит посылка (UCSZ01=1 и UCSZ00=1)
}
//
void USART_Transmit( unsigned char data ) //Функция отправки данных
{
    while ( !(UCSR0A & (1<<UDRE0)) );
    //Ожидание опустошения буфера приема
    UDR0 = data; //Начало передачи данных
}
```

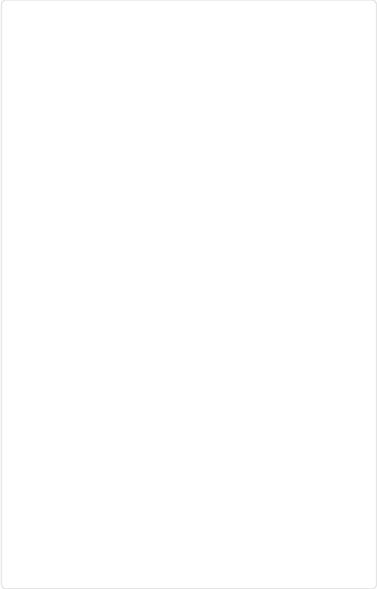
Для частоты тактирования 16 МГц значение UBRR0 будет именно 16 при условии, что удвоение у нас также включено

Table 17-12. Examples of UBRRn Settings for fosc = 16.0000 MHz

Baud Rate (bps)	fosc = 16.0000 MHz			
	U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error
115.2k	8	-3.5%	16	2.1%
4800	207	0.2%	416	-0.1%

Также уберем инициализацию портов из функции **main()** и вызовем там функцию инициализации **USART**

```
int main(void)
{
    USART_Init (16); //115200
    while (1)
```





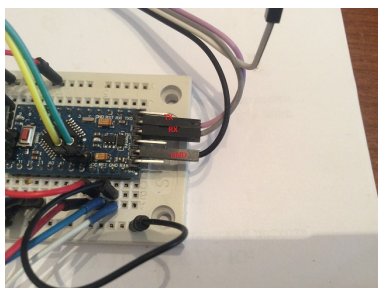
Arduino P

Cayenne

Теперь давайте запустим и настроим терминальную программу и проверим работоспособность нашего USART. Для этого мы также скопируем код из Test12.c, касающийся передачи данных и вставим в main()

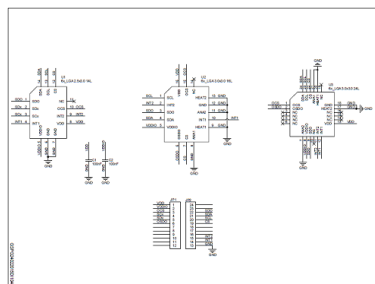
```
USART_Init (16); //115200
USART_Transmit('O');//Передаем при
включении
USART_Transmit('k');//сообщение
"Ok!", что свидетельствует
USART_Transmit('!');//о правильно
работе программы
USART_Transmit(0x0d);//переход в
начало строки
USART_Transmit(0x0a);//переход на
новую строку
while (1)
```

Переходник USART подключим к следующим ножкам платы

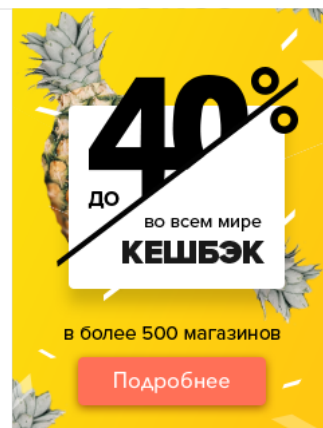


Данный датчик у меня находится на плате от STM **STEVAL-MKI160V1**. Данная плата предназначена в свою очередь для соединения и работы с оценочной платой **X-NUCLEO-IKS01A1**, которая нам будет уже не нужна. Также данный датчик можно найти и без платы, и будет это гораздо даже дешевле.

Так как у меня именно такая плата, то приведу её схему (нажмите на картинку для увеличения изображения)



В данной схеме уже распаяны соответствующие конденсаторы, необходимые для фильтрации различных помех. Также данная схема представлена без датчиков, с одними только разъёмами. У меня на такой плате установлен только один датчик, именно тот, который мы рассматриваем сегодня. Он на схеме

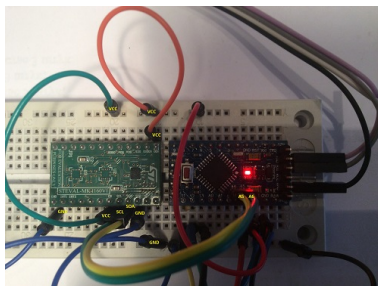


ol sensors,
tors. Drag &
r. Free dow

можно сказать представлен в виде контактной площадки, находящейся на рисунке слева. На данной плате снизу распаяны две контактные площадки в виде штырьков с интервалом 2,54 мм, что позволяет воткнуть её в безопасную макетную плату, что мы и сделаем. И посредством гибких проводов соединим с микроконтроллером.

Самое главное! **Внимание!** Данный датчик, как видно из его технической документации питается напряжением от **1,71** вольт до **3,6** вольт, но не как не от 5 вольт. Поэтому питать схему нужно напряжением не более **3,6** вольт. У меня в программаторе есть переключатель, который я перевёл в положение **3,3** вольт.

Подключается данный датчик двумя способами — посредством шины I2C и шины SPI. Мы будем пользоваться I2C, так как урок для STM был именно с применением данной шины, а следовательно, мне было легче писать сценарий для урока по AVR. Также в технической документации мы можем прочитать, какие ещё контакты кроме двух контактов шины I2C нужно ещё подсоединить. Получится примерно вот так:



Сборку проекта, тестирование и дальнейшие действия продолжим в [следующей части](#).

*Предыдущий
урок*

*Программирование
МК AVR*

*Следующая
часть*

Техническая документация:

[Документация на датчик](#)
[Документация на оценочную плату](#)

Приобрести плату Atmega 328p Pro Mini можно [здесь](#).

Программатор (продавец надёжный)
[USBASP USBISP 2.0](#)

Приобрести платы с датчиком LSM6DS3 можно у следующих продавцов:

Надёжный продавец [LSM6DS33 STEVAL-MK1160V1](#)

Здесь дешевле [LSM6DS33 STEVAL-MK1160V1](#)

Здесь другая плата, намного дешевле,
но от другого разработчика [LSM6DS3](#)

Смотреть
ВИДЕОУРОК (нажмите на
картинку)



Post Views: 463

AVR УРОК 38.
Atmega 328p Pro

Mini

AVR УРОК 39.
Акселерометр
LSM6DS3. Часть

Добавить
комментарий

Ваш e-mail не будет опубликован.
Обязательные поля помечены *

Комментарий

Имя *

E-mail *

Сайт

+

=

Отправить комментарий

