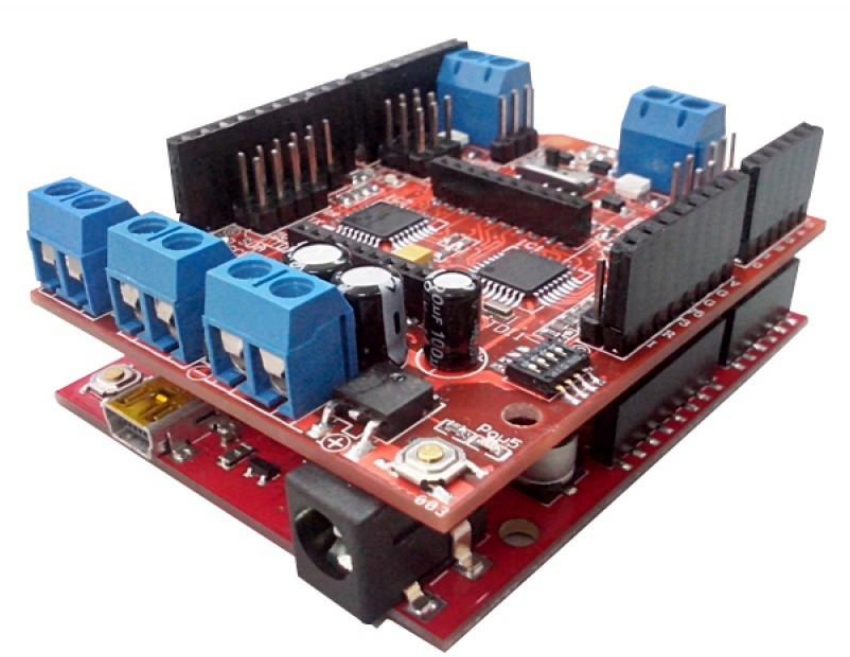
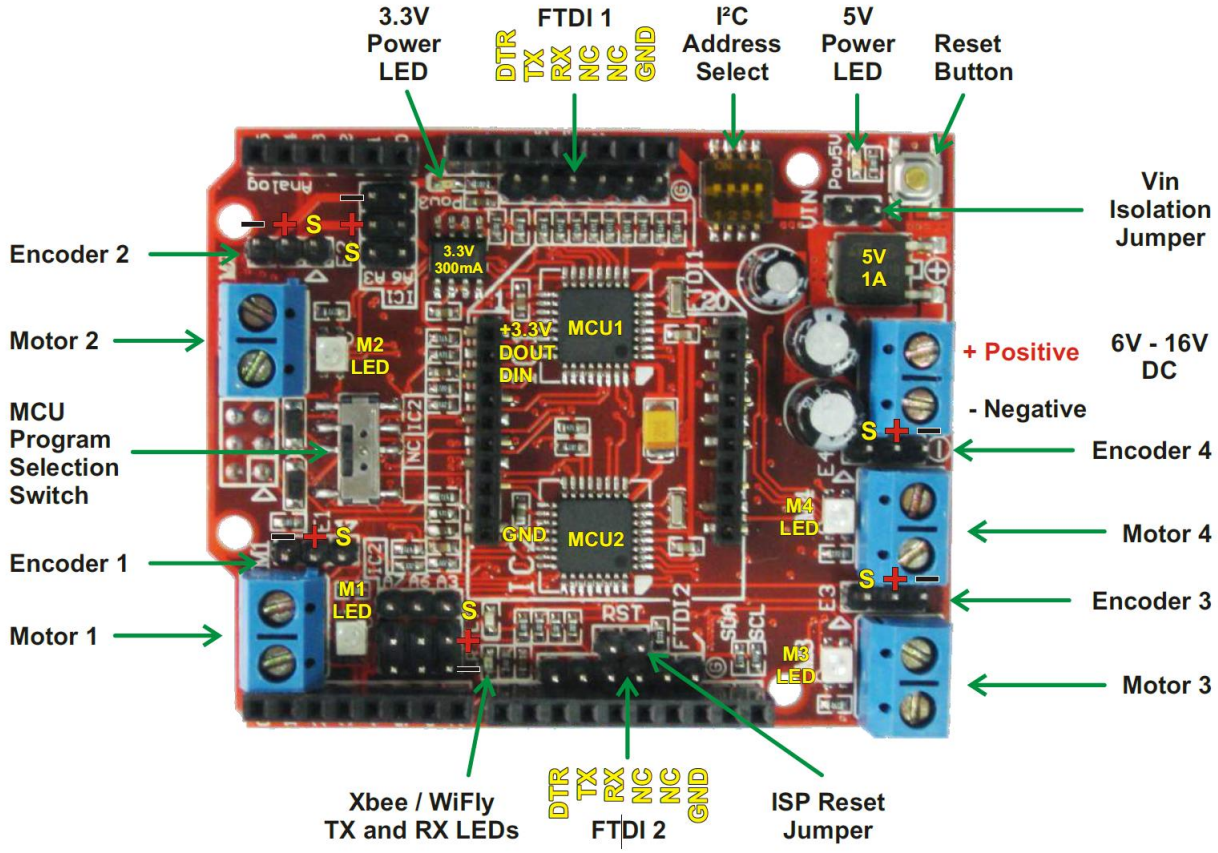
ComMotion Motor Driver Shield for 4 motors
Производитель: DAGU Electronics
(перевод – Владимир_ http://forum.amperka.ru)



Краткое описание

Перед вами 4-канальный контроллер электродвигателей, который позволяет управлять 4 моторами одновременно с функцией считывания показаний энкодеров, возможностью беспроводного управления и взаимодействовать через интерфейс I^2C. Шилд подойдет как для ARDUINO UNO R3, так и других Arduino совместимых контроллеров, например, таких как RedBoard от SparkFan. ComMotion разработан, чтобы управлять 4-DC коллекторными моторами при постоянном 2.5A, с пиковым током до 4A на двигатель. С предустановленных конфигураций вы сможете легко управлять роботами с *Omni* или *Mecanum* колесами, задав скорость, угол и вращение. На плате установлены две микросхемы atmega328p, которые будут делать все математические расчеты для каждого отдельного двигателя.

Каждый процессор Atmega328p имеет свой собственный серийный порт, который может использоваться для подключения GPS, Bluetooth и ЖК-модулей, оставляя для контроллера Arduino последовательный порт для загрузки и отладки кода. Последовательный порт на MCU2 также имеет разъем для подключения модулей Xbee или WiFly, при этом надо просто подключить предварительно настроенные платы Xbee, WiFly или Bluetooth и у вас есть беспроводное управление.



Введение

ComMotion - I²C, это плата, которая управляет 4 контроллерами моторных каналов в формате шилда (ревизия R3).

Преимущество использования I²C в том, что шилд использует только 2 пина. Это оставляет большинство пинов свободным для использования с другими шилдами, датчиками и схемами.

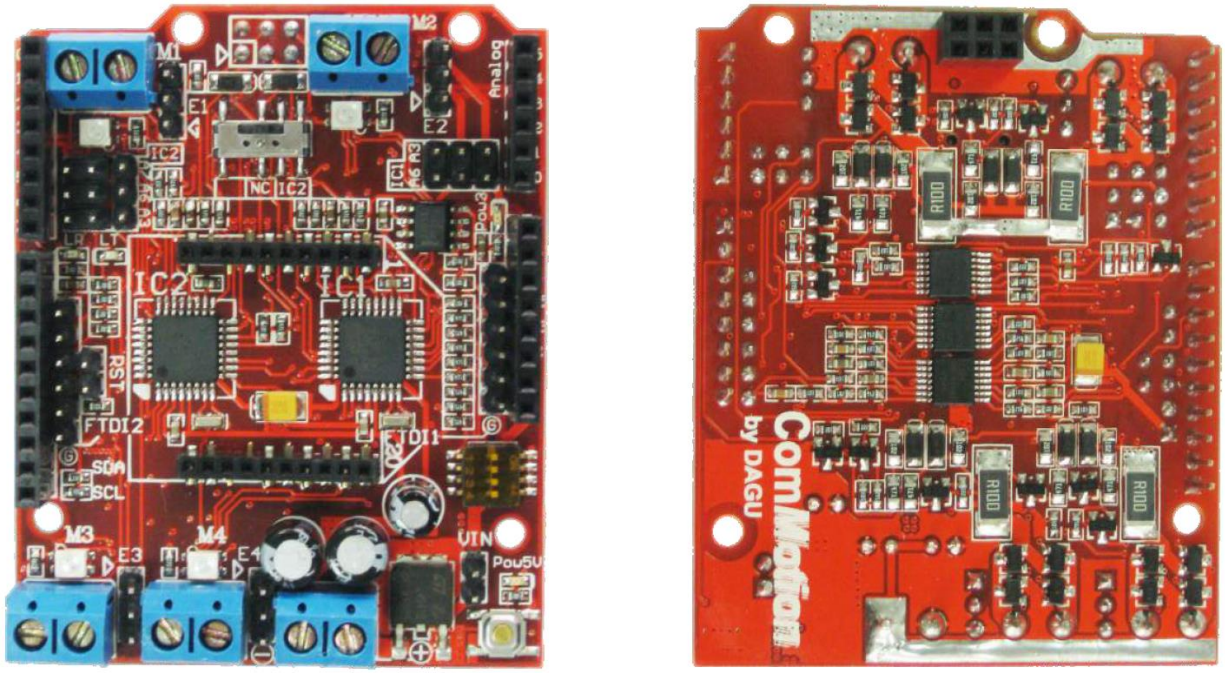
Шилд ComMotion может управлять двигателями (4 DC) с силой тока для каждого двигателя до 2.5 А непрерывно, и с пиковым током до 4 А на каждый двигатель. Текущий контроль используется, чтобы ограничить ток максимума для каждого двигателя. Каждому двигателю соответствует энкодер, что может использоваться для точной регулировки скорости. Обратная связь энкодера также дает двигателю намного больше вращающего момента на низких скоростях.

Заданные конфигурации позволяют Вам управлять роботами или с *Omni* или с *Mecanum* колесами, посылая на шилд ComMotion только 3 простых значения: Скорость, Угол и Вращение. При этом два ATmega328P бортовых процессора выполнят тригонометрические преобразования для вычисления правильной скорости каждого отдельного двигателя.

Если кодирующие устройства не будут использоваться, то шилд будет функционировать как обычно, и контроль частоты вращения двигателя совпадет с фактической частотой моторов. Регулировка скорости будет менее точной, и у двигателей будет намного меньше вращающего момента на низких скоростях.

У каждого процессора есть свой собственный последовательный порт, выведенный через разъем FTDI (Future Technology Devices International —

марка компании, торгующей полупроводниковыми устройствами). Эти последовательные порты могут использоваться для GPS, Bluetooth и жидкокристаллических дисплеев, оставляя последовательный порт Ардуино свободным для загрузки и отладки кода. Сериал порт на MCU2 также вставляется в гнездо для XBEE или приемопередатчика радио WiFly со схемой перевода напряжения в специальное (300 мА, 3.3В).



ComMotion PCB (*printed circuit board — печатная плата*) – 4-слойная, двусторонняя печатная плата, в которой упаковано много особенностей в одном маленьком шилде.

Беспроводной контроль

Конфигурация по умолчанию позволяет шилду ComMotion принимать последовательные команды непосредственно от последовательного порта на MCU2. Просто включите предварительно сконфигурированный Xbee, WiFly или модуль Bluetooth и Вы сразу имеете беспроводной контроль. При этом Ардуино не требуется.

Включение ComMotion

Шилд ComMotion может использовать целых 10А тока с 4 двигателями, достигающими 2.5А каждый, при этом важно использовать хорошие качественные батареи NiMh или литиевые аккумуляторные батареи, внимание: при этом не используйте щелочные батареи.

Т.к. плата Ардуино не разработана для таких токов и высокого напряжения, поэтому Вы должны соединить свою аккумуляторную батарею или электроснабжение прямо к шилду ComMotion.

Если установить (замкнуть) джампер питания (перемычку), то плата Ардуино получит питание от шилда. Поскольку шилд ComMotion может работать с более широким диапазоном напряжения, чем большинство плат Ардуино, может быть необходимо привести его в действие от отдельного питания. Удалите джампер питания, чтобы изолировать Ардуино VIN от электроснабжения ComMotion, тогда Вы можете привести в действие плату Ардуино и шилд ComMotion от отдельных аккумуляторных батарей или электроснабжения.

Из коробки

Шилд ComMotion идет с демонстрационным программным обеспечением, используемым в комплекте с роботом. Когда Вы впервые подключите его, на нем будет воспроизводиться мелодия и несколько раз прозвучит звуковой сигнал от использования всех подключенных двигателей.

Демонстрационная программа может быть включена или выключена, нажав кнопку сброса или при сбросе питания. Даже если Вы не используете робота, это - полезный инструмент, чтобы определить, функционирует ли шилд хорошо. Как только шилд получит данные или команду, демонстрация прекратиться, если Вы опять не перезагрузите конфигурацию в демонстрационный вариант.

Шилд ComMotion предназначен, чтобы работать по I²C, который управляет 4 каналами и умным контроллером моторов. «Умный» термин обозначает, что контроллер будет использовать входы энкодера, чтобы попытаться поддержать устойчивые частоты вращения двигателя независимо от нагрузки на моторы. Это позволяет двигателям работать на низких скоростях и поддерживать полный вращающий момент без остановки.

Если Вы не используете кодирующие устройства, то эта опция может быть отключена в конфигурации. Контроллер будет тогда вести себя, как обычный задатчик моторов.

Использование ComMotion впервые

Первое, что нужно сделать - определить, какой адрес I²C Вы хотите использовать. Даже если Вы планируете использовать последовательный контроль, Вы должны проверить, чтобы удостовериться, что адреса I²C платы не находятся в конфликте ни с какими другими устройствами I²C, которые Вы могли бы использовать дополнительно.

Процессоры общаются друг с другом, используя шину I²C. По умолчанию шилд ComMotion для I²C использует 30 и 31 позицию со всеми dip-переключателями в положении «выкл.» (off). Если имеется противоречие с другим устройством на шине, тогда используйте таблицу ниже, чтобы выбрать новый адрес. Хотя есть 2 процессора на шине I²C, Вы можете общаться только с MCU1. Адрес MCU2, как видно, помогает предотвратить конфликт с другими устройствами.

DECIMAL VALUE	DIP SWITCH SETTING	ADDRESS	
		MCU1	MCU2
0		0	1
1		2	3
2		4	5
3		6	7
4		8	9
5		10	11
6		12	13
7		14	15

DECIMAL VALUE	DIP SWITCH SETTING	ADDRESS	
		MCU1	MCU2
8		16	17
9		18	19
10		20	21
11		22	23
12		24	25
13		26	27
14		28	29
15		30	31

Управление ComMotion

Все команды посылаются в шилд ComMotion, используя или шину I²C или последовательный порт. Команды идут пакетами данных. Это фактически проще, чем это звучит и как только пакеты конфигурации приходят, Вы будете главным образом использовать разъем 3, чтобы управлять частотами вращения двигателя.

Первый байт каждого пакета данных говорит плате, какую информацию посылают. Остальная часть пакета должна закончить эту команду.

У байтов может только быть значение от 0 до 255, но иногда значения будут больше (это числа под названием integer - целые числа). Целые числа 16-битные и нужно постараться, прежде чем мы сможем послать их. Самый легкий путь при помощи IDE Ардуино и highByte() и lowByte() команд. Как только целые числа будут разделены на два байта, они могут быть переданы по шине I²C или последовательному потоку данных.

Например, чтобы управлять скоростью каждого двигателя индивидуально Вы сначала посылаете номер 3, который указывает, что это пакет, а потом команду, чтобы управлять двигателями. После чего посылаете желаемую скорость каждого двигателя от -255 до +255. Даже если Вы не используете 4 двигателя, Вы все еще должны послать 4 значения скорости. Просто используйте 0 для любых неиспользованных двигателей.

Существует 6 команд, необходимых для шилда ComMotion.

1. Конфигурация Контроллера - установка таких значений, как напряжение низкого уровня заряда, максимум проезжает ток и т.д.
2. Конфигурация Энкодера – установка частоты вращения мотора RPM (об/мин), разрешения энкодера, резервирование питания и установка времени.
3. Управление моторами - управление скоростью двигателей.

4. Конфигурация Последовательного порта – установка скорости передачи в бодах для каждого порта и избранный порт для команд последовательного контроля.
5. Пересылка последовательных данных - посылает данные в последовательный порт.
6. Статус шилда - посылает информацию (напряжение батареи, текущий мотор и т.д.)

Базовая конфигурация

Вы должны будете сконфигурировать шилд ComMotion, по крайней мере однажды, чтобы выполнить Ваш проект. Это можно сделать, послав основной пакет данных конфигурации, который состоит из 9 байтов, как указано ниже:

Байт	**Функция и краткое описание**
1	1=basic config. Определяет, что это пакет данных базовой конфигурации.
2	mode 0=I²C Serial mode 1=demo mode
3	configuration 0=3x Omni 1=4x Omni 2=Mecanum 3=individual +16=encoders off
4	low battery voltage 60 = 6.0В используйте 60 для 2S LiPo, используйте 90 для 3S LiPo
5	max. current M1 255 = 2.55 A
6	max. current M2 255 = 2.55 A
7	max. current M3 255 = 2.55 A
8	max. current M4 255 = 2.55 A
9	I²C offset 0=no offset I²C ранее расширил диапазон доступных адресов I²C
10	I²C master address 1-по умолчанию. Это адрес внешнего I²C контроллера

Байт 1: Должен всегда = 1. Это говорит шилду, что пакет данных - данные о базовой конфигурации.
Байт 2: Демонстрационный мод = 1 – по умолчанию. Установите 0, если Вы хотите управлять шилдом I²C или последовательными командами.
Байт 3: Выберите свою конфигурацию шасси или выберите отдельное устройство управления двигателем. Добавьте 16, если Вы не используете кодирующие устройства.
Байт 4: Значение по умолчанию равняется 60 (6 В). Это значение - минимальное напряжение батареи x10. Ниже этого напряжения происходит отключение.
Байты 5-8: Вы можете установить максимальный ток для каждого двигателя. По умолчанию значение = 255, которое равно 2.55 А.
Байт 9: Если необходимо, Вы можете отключить адрес I²C, чтобы избежать конфликтов. Значение по умолчанию 0.
Байт 10: если шилд ComMotion подключен совместно с другим устройством I²C, то это - адрес внешнего контроллера I²C. Если Вы используете внешнее устройство I²C, тогда это значение должно быть равно его адресу, чтобы получать информацию и предупреждения. Значение по умолчанию = 1.

Конфигурация энкодера (декодирующего устройства)

Если Вы не используете энкодеры, тогда Вы можете проигнорировать эту команду.

Поскольку ComMotion использует время между изменениями положения энкодера чтобы определить фактическую скорость двигателя, при этом энкодер определяет действительную скорость. Когда Вы задаете контроллеру желаемую скорость от -255 до +255, контроллер конвертирует это число в процентах от максимальной скорости.

Краткое описание функции байта

1	2=encoder config. Определяет, что пакет данных - данные конфигурации энкодера
2	макс.об/мин RPM мотора в номинальном напряжении (верхний байт)
3	макс.об/мин RPM мотора в номинальном напряжении (нижний байт)
4	энкодера разр. x100 800=8 измерений при вращения мотора на 1 оборот (верхний байт)
5	энкодера разр. x100 800=8 измерений при вращения мотора на 1 оборот (нижний байт)
6	резервное питание % 50=50%
7	макс. измеренное время в мс. (1-255). Значение 0 неправильно и будет автоматически изменено на 1.

Байт 1: Должен всегда быть 2. Это говорит шилду, что пакет данных - данные конфигурации энкодера.

Байт 2 и 3: Это - максимальная скорость двигателей под нагрузкой при номинальном напряжении.

Байт 4 и 5: Это - разрешение x100 энкодера. Например, 225=2¼ состояние изменяет направление вращения мотора.

Не путайте разрешение энкодера с вращением колеса, которое зависит от передаточного отношения редуктора.

Байт 6: Зарезервированное питание, как процент от 0% - 50%. По умолчанию составляет 10%.

Байт 7: По умолчанию составляет 10 мс, но возможно некоторое экспериментирование в зависимости от применения.

Конфигурация энкодера по умолчанию установлена для двигателя с максимальной скоростью 13.500 об/мин и кодирующим устройством с разрешением 8-ми положений на 1 вращение мотора. Это годится для комплекта робота, однако, для шасси Ровера 5 скорость должна быть установлена в 8.500 об/мин и набор разрешений энкодера - 2 положения за оборот. Неправильная настройка контроллера будет препятствовать тому, чтобы двигатели достигли своей максимальной скорости на 100%, или наоборот, достигли своей максимальной скорости ранее, чем на 100%.

Гарантировать достижение желаемой скорости вращения двигателей при изменении нагрузки можно, только обеспечивая некоторый запас мощности. Это гарантирует, что даже на максимальной скорости робот все еще может поехать по прямой линии или по желаемой траектории.

Если запас слишком мал, тогда моторы не обеспечат точность на высоких скоростях под нагрузкой и робот не переместится максимально быстро. Как правило, 10-15% запаса (мощности) должно быть достаточно, но

это будет зависеть от веса робота, трения в колесах и на ландшафте. Тут может потребоваться некоторое экспериментирование.

Максимальное время останова используется контроллером, чтобы определить, работает ли двигатель просто очень медленно, или фактически стоит. Большие значения могут сделать возможными более медленные скорости, но также замедлят время отклика контроллеров. С низким разрешением энкодеру будут требоваться более длительные времена останова. Тут также будет требоваться некоторое экспериментирование.

Для комплекта робота, у которого есть двигатели на 13,500 об/мин и разрешения энкодера - 8 положений на 1 оборот, максимальное время останова 10-15 мс работает хорошо.

Для Ровера 5 шасси, у которого есть двигатели на 8,500 об/мин и разрешение энкодера 2 положения за оборот (использование только 1 из 2 энкодеров) лучше принять максимальное время останова 25-50 мс.

Комплекты DAGU используют 8 магнитных полюсов и датчики эффекта Холла, однако, шилд ComMotion будет работать с любым кодирующим устройством - это обеспечивает цифровой выход. Высокие разрешения требуются для хорошего диапазона регулировки скорости.

Шасси Ровер 5

Комплект робота

Управление двигателями

Как только шилд сконфигурирован, мы можем управлять двигателями с пакетом данных управления двигателем. Этот пакет данных Вы будете использовать чаще всего. Если Вы устанавливаете конфигурацию шасси для 3х-колесного Omni, 4х-колесного Omni или колеса Mecanum, тогда контроллер автоматически вычислит тригонометрию, требуемую для определения отдельных частот вращения двигателя. Это позволит Вам управлять шасси робота всего 3 целыми числами.

Скорость: желаемая скорость от -255 до +255. Положительные значения - вперед, отрицательные величины - назад.

Угол: 0° - вперед, 90º - вправо, 180° - влево, 270° - оставлен. Отрицательные величины полностью изменяют угол.

Вращение: значение от -255 до +255. Положительные значения - вращение по часовой стрелке.

Если Вы устанавливаете конфигурацию для отдельного устройства управления двигателем тогда, Вы должны послать только 4 числа: скорость

каждого двигателя от -255 до +255. Положительные значения управляют двигателем «вперед», отрицательные величины управляют двигателем «назад».

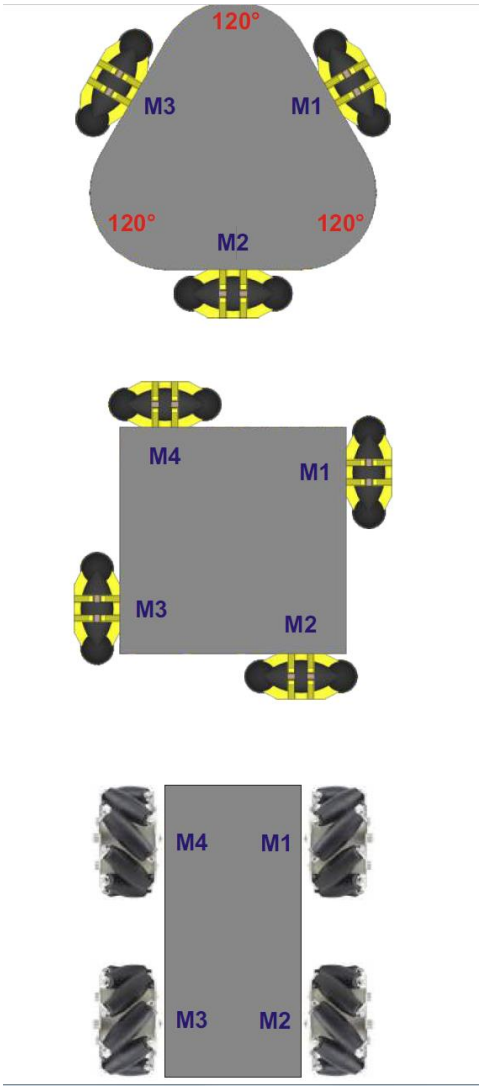
Описание функции байта

1 3=motor control Определяет, что пакет данных - данные об устройстве управления двигателем

	Omni и mecanum колеса	**Индивидуальное управления двигателем**
2	скорости высокий байт	скорость M1 высокий байт
3	скорости низкий байт	скорость M1 низкий байт
4	угла высокий байт	скорость M2 высокий байт
5	углов низкий байт	скорость M2 низкий байт
6	вращений высокий байт	скорость M3 высокий байт
7	вращений низкий байт	скорость M3 низкий байт
8	не требуемый	скорость M4 высокий байт
9	не требуемый	скорость M4 низкий байт

Байт 1: должен всегда быть 3. Это говорит шилду, что пакет данных - данные об устройстве управления двигателем.

Как 16-битные числа, они должны быть разделены в байты целых чисел командами highByte(), lowByte() при использовании IDE Ардуино. Тогда они смогут быть переданы по шине I²C или последовательному потоку данных.



Использование конфигурации шасси

У шилда ComMotion есть 3 конфигурации, запрограммированные в него. Когда одна из этих конфигураций выбрана, тогда шилд вычислит скорость и направление каждого двигателя, на основании трех целых чисел - скорости, угла и вращения.

Используя любую из этих 3 предопределенных конфигураций, двигатели всегда нумеруются в направлении по часовой стрелке, как показано на рисунках ниже. Согласуйте моторные числа с выходами моторов на шилде.

Независимо от того, которую конфигурацию Вы выбираете, тест, которым определяется правильность подключения - попытка шасси вращаться. Если двигатель вращается в неправильном направлении, тогда поменяйте провода в винтовых зажимах.

Mecanum колеса

Если Вы используете mecanum колеса, тогда удостоверьтесь, что они ориентируются так же, как показано на рисунке.

Omni колеса

Вы заметили, что omni колеса расположены на треугольном шасси вокруг центра, но на квадратном шасси колеса стоят обычно.
Это вызвано тем, что с omni колесами не имеет значения, как они расположены.

Только угловой вариант.

Иногда размещение Ваших колес позволяет Вам делать шасси меньшего размера, потому что это позволяет более эффективно управлять двигателями.

Другие конфигурации

Если Ваше шасси не соответствуют одной из этих конфигураций, тогда отдельно сконфигурируйте шилд для управления двигателями. Также Ваше программное обеспечение может управлять двигателями, которые удовлетворяют Вашему шасси.

Последовательная (Serial) конфигурация

Этот пакет данных позволяет Вам устанавливать скорость передачи в бодах для последовательных портов на MCU1 и MCU2 и определять, может ли ComMotion шилд принять команды непосредственно от последовательного порта или просто передать данные владельцу I²C для обработки. Скорость передачи в бодах по умолчанию для каждого последовательного порта установлена 9600.

Для остальной части этого руководства порт 1 относится к последовательному порту на MCU1, а порт 2 относится к последовательному порту на MCU2.

Описание функции байта

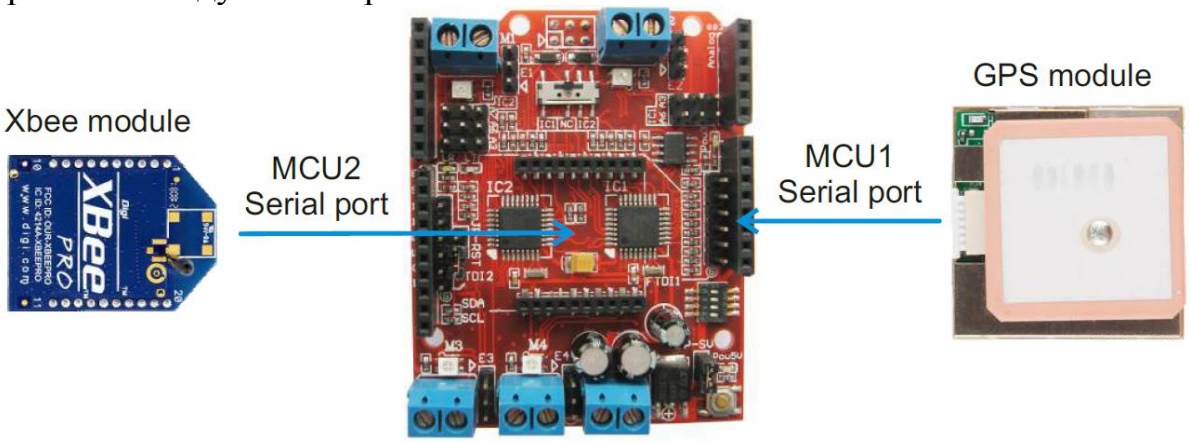
1	4=serial configuration определяет, что пакет данных - последовательные данные конфигурации	
2	buad rate high byte	установка скорости передачи в бодах для порта 1.
3	buad rate low byte	установка скорости передачи в бодах для порта 1.
4	buad rate high byte	установка скорости передачи в бодах для порта 2.
5	buad rate low byte	установка скорости передачи в бодах для порта 2.
6	serial mode (по умолчанию 4):	
	0=передают все последовательные данные владельцу I²C для обработки.	
	1=прием пакетов данных о контроле от порта 1 - передают все данные владельцу I²C для обработки.	
	2=прием пакетов данных о контроле от порта 2 - передают все данные владельцу I²C для обработки.	
	3=прием пакетов данных о контроле от порта 1 - передают назад все данные в порт 1.	
	4=прием пакетов данных о контроле от порта 2 - передают назад все данные в порт 2.	

В способах 0,1 и 2 данные от последовательных портов 1 и 2 идут в шину I²C. Указание, куда прибыли данные от каждого последовательного пакета начнется с идентификационного заголовка #SP1 или #SP2.

Шилд ComMotion примет команды от последовательного порта на MCU2 и передаст все данные обратно в последовательный порт на MCU2 по умолчанию (метод 4). Это означает это, если Вы устанавливаете Bluetooth, Xbee или модуль WiFly тогда шилд ComMotion может управляться с

помощью беспроводных технологий без Ардуино или любого другого внешнего процессора, и все последовательные данные, полученные из последовательного порта на MCU1 можно передать обратно через последовательный порт на MCU2.

В примере ниже, предварительно сконфигурированный модуль Xbee включен в шилд ComMotion, и модуль GPS, связанный с портом 1. Все данные, полученные от модуля GPS, будут автоматически переданы назад через Xbee модуль на порт 2.



Посылка последовательных (serial) данных

Этот пакет данных используется, чтобы послать данные в последовательный порт. Максимальный предел данных за пакет составляет 30 байтов из-за буфера I²C. Если Вы должны послать больше чем 30 байтов последовательных данных, Вы должны послать многократные последовательные пакеты данных. Предотвратить переполнение буфера при посылке многократных пакетов можно, ограничивая размер пакета данных и посылая большее количество меньших пакетов. Это замедлит скорость передачи, но позволит буферу очищаться между пакетами.

Описание функции байта

0	5=serial data	определяют, что пакет данных - последовательные данные
1	select serial port	0=previous 1=MCU1 порт 2=MCU2 порт

Последовательные данные 2-31 любой длины до 30 байтов

Байт 0: должен всегда быть 5, чтобы сказать шилду ComMotion, что Вы хотите послать последовательные данные.
Байт 1: выбирает последовательный порт и посылает в него данные. Остальная часть пакета данных является посылаемыми данными.

Получение последовательных данных

Как упомянуто в Последовательной Конфигурации, все последовательные данные, полученные от любого последовательного порта, будут направлены последовательным способом. Если, например, у шилда ComMotion есть последовательный ЖК-монитор на последовательном порту

1 и модуль GPS на сериальном порту 2, тогда установите mode в 0. Ардуино тогда получит все данные от модуля GPS до шины I²C, и может показать данные, посланные в порт 1.

Требования состояния шилда

Шилд ComMotion может дать Вам информацию, которую Вы можете счесть полезной в Ваших программах. Запрашивая данные статуса от шилда, Вы можете прочитать напряжение батареи, а также, сколько использует тока каждый двигатель. Вы можете прочитать аналоговые датчики, связанные с запасными аналоговыми входами. Если Вы используете энкодеры, Вы можете прочитать значение каждого, чтобы получить величину пройденного расстояния. Вы можете даже запросить от шилда информацию об ошибке, например, от чего остановился двигатель.

Описание функции байта

1	6=запрос статуса	Определяет, что пакет данных - запрос о состоянии
2	запрос данных	Каждый бит – специальный запрос, позволяющий получить любую информацию о положении.

Байт 1: должен всегда быть 6. Это говорит шилду, что пакет данных - запрос о состоянии

Байт 2: каждый бит - различный запрос, как упомянуто ниже. Вы можете просить любую комбинацию статуса.

Статус отображается от бита 0 к биту 7. Например, если биты 3 и 4 будут высоки (8+16), то 6 байтов будут возвращены для аналоговых входов на MCU1, и будут являться аналоговыми входами от MCU2.

Бит 0: Возвращает 8 байтов, количество отсчетов энкодера от каждого моторного высокого байта с начала работы.

Бит 1: Сброс всех значений энкодера. Если бит 0 будет высок, то значения будут прочитаны прежде, чем будут перезагружены.

Бит 2: Возвращает 8 байтов, текущее равенство каждого моторного высокого байта с начала.

† Бит 3: Возвращает 6 байтов, аналоговые входы A3, A6 и A7 от MCU1, высокий байт с начала. A7 - напряжение батареи.

Бит 4: Возвращает 6 байтов, аналоговые входы A3, A6 и A7 от MCU2, высокий байт с начала.

Бит 5: Возвращает 1 байт, журнал ошибок для двигателей.

Бит 6: Возвращает 1 байт, журнал ошибок для коммуникаций.

Бит 7: Очищает журналы ошибок. Если бит 5 или 6 будет высок, то журналы ошибок будут прочитаны сначала.

† Напряжение батареи = MCU1 аналоговый вход A7 * 30 / 185. Результат 84 = 8.4 В.

Журнал ошибок моторов

Когда Вы запросите моторный журнал ошибок, Вы получите единственный байт. Значение 0 обозначает отсутствие ошибок.

Каждый бит указывает на различную ошибку:

Бит 0: высокое значение указывает, что мотор 1 использует ток максимума, как установлено в данных конфигурации шилда.

Бит 1: высокое значение указывает, что мотор 2 использует ток максимума, как установлено в данных конфигурации шилда.

Бит 2: высокое значение указывает, что мотор 3 использует ток максимума, как установлено в данных конфигурации шилда.

Бит 3: высокое значение указывает, что мотор 4 использует ток максимума, как установлено в данных конфигурации шилда.

Бит 4: высокое значение указывает, что мотор 1 использует чрезмерный ток и произошел останов.

Бит 5: высокое значение указывает, что мотор 2 использует чрезмерный ток и произошел останов.

Бит 6: высокое значение указывает, что мотор 3 использует чрезмерный ток и произошел останов.

Бит 7: высокое значение указывает, что мотор 4 использует чрезмерный ток и произошел останов.

Нормальное текущее состояние двигателя достигает предельных значений в данных конфигурациях тогда, когда мощность к двигателю будет уменьшена, чтобы ограничить ток максимальным значением. Если моторный ток превышает безопасный уровень для шилда, тогда шилд производит останов мотора в попытке предотвратить повреждение.

В случае короткого замыкания некоторые батареи могут подавать очень высокие токи, и шилд может не быть в состоянии произвести останов мотора достаточно быстро. Если Вы используете литиевые батареи 2.5 А (быстрая зарядка), плавкий предохранитель должен быть соединен последовательно с каждым двигателем как добавочный уровень защиты против коротких замыканий.

Коммуникационный журнал ошибок

Когда Вы будете запрашивать коммуникационный журнал ошибок, Вы получите единственный байт. Значение 0 обозначает отсутствие ошибок.

Каждый бит указывает на различную ошибку:

Бит 0: высокая значение указывает на переполнение буфера I²C.

Бит 1: высокое значение указывает на последовательный порт 1 (переполнение буфера).

Бит 2: высокое значение указывает на последовательный порт 2 (переполнение буфера).

Бит 3: зарезервирован для будущих обновлений программного обеспечения.

Бит 4: зарезервирован для будущих обновлений программного обеспечения.

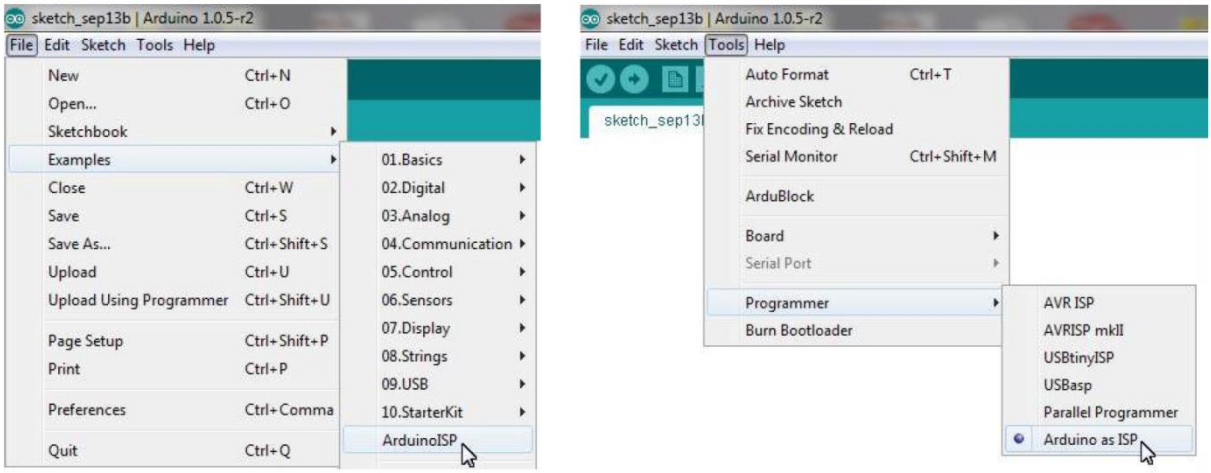
Бит 5: зарезервирован для будущих обновлений программного обеспечения.

Бит 6: зарезервирован для будущих обновлений программного обеспечения.
Бит 7: зарезервирован для будущих обновлений программного обеспечения.

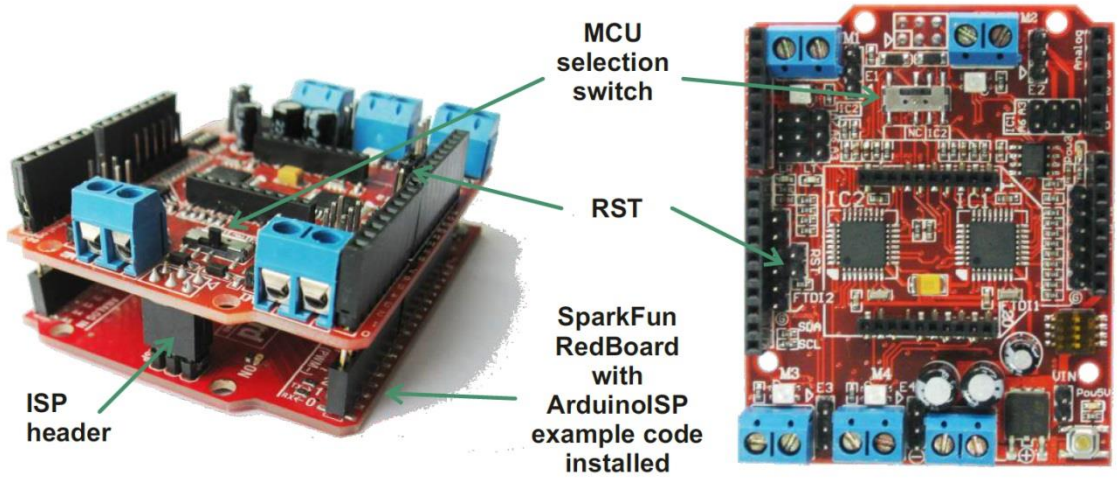
Перепрограммирование ComMotion

В редких случаях флэш-память может быть испорчена во время замыкания. Если это действительно произошло, тогда ComMotion код может быть легко восстановлен, используя пример кода ArduinoISP, который включен в IDE Ардуино.

Загрузите пример кода ArduinoISP в плату Ардуино и затем включите шилд ComMotion. В меню Инструменты (Tools), установите тип платы как UNO и выберите "Arduino is ISP" в качестве программатора.



Закоротите джампер (перемычку) RST на шилде ComMotion. Это соединит D10 платы Ардуино и Reset шилда ComMotion. Теперь используйте выключатель выбора MCU на шилде, чтобы выбрать, какой процессор надо перепрограммировать. Загрузите код ComMotion и используйте "Загрузку используя программатор" для загрузки кода.



Тот же самый код должен быть загружен на оба процессора, так что загрузите сначала MCU1, а потом MCU2. Когда код будет загружен на оба процессора, тогда поместите выключатель шилда в центральное положение, чтобы изолировать заголовок ISP и открыть джампер RST.

В крайних случаях плавкие предохранители, возможно, также должны быть перезагружены. Это надо сделать перед прошивкой загрузчика в каждый процессор. Загрузчик будет переписан, когда мы загрузим код, и это - самый правильный способ гарантировать параметры настройки. Как только загрузчик будет загружен в оба процессора, загрузите код снова.

Изменение кода

Только опытные программисты могут пытаться изменить код. Если Вы намереваетесь изменить код, тогда Вы должны понять, что код – чувствителен ко времени. Вы не можете использовать функции, такие как delay() или delayMicroseconds(), поскольку это будет мешать тому, чтобы код управления частотой вращения двигателями работал правильно. Вы должны удостовериться, что в работе с обоими процессорами используется переменный "байт mcu"; и определить, на каком mcu это работает. Используйте заголовочную таблицу [IOpins.h] как Вашу монтажную схему.

Технические характеристики ComMotion

- Процессоры: 2x ATmega328P (16 МГц)
- Напряжение поставки: 6 В - 16 В
- Логическое напряжение: 5 В 1000 мА*
- Беспроводная поддержка: Xbee / гнездо WiFly с переводом напряжения Xbee / питание WiFly: 3.3 В, 300 мА
- Значение напряжения батареи: 0 В – 17 В
- Разрешение монитора батареи: ≈ 0.02 В
- Аналоговые входы: 5x 10 битов (A3, A6 MCU1 – A3, A6, A7 MCU2)
- Моторные драйверы: 4x FET-“H” мост
- Допустимый непрерывный ток: 2.5 А (каждый двигатель)
- Предельный ток останова: 4 А (каждый двигатель)
- Значение тока: 0А – 5 А (каждый двигатель)
- Текущее измеряемое разрешение: ≈5 мА (каждый двигатель)
- Напряжение на шине I²C: 5 В или 3.3 В (на пине IO_REF)
- Частота шины I²C: 100 кбит/с
- Адреса I²C: 16 выбираемых пар (конфигурируемое программное обеспечение)
- Последовательные порты: 2x 5-вольтовая логика TTL (заголовки FTDI)

*Текущий предел для 5-вольтового регулятора - предохранитель по напряжению и температуре окружающей среды.

Номинальные токи (не выше) при температуре окружающей среды 25°C.
1A - 6В 580 мА - 8.4В 285 мА - 12В 180 мА – 16В