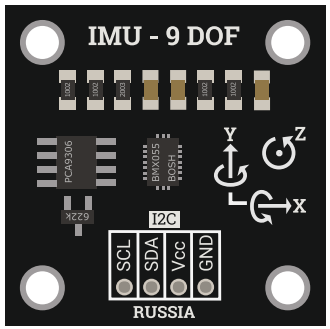
IMU-сенсор на 9 степеней свободы (Trema-модуль V2.0)



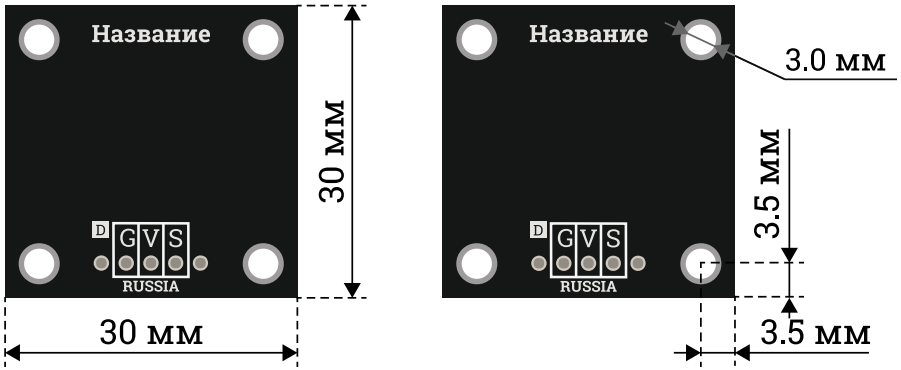
Общие сведения:

Trema-модуль IMU 9 DOF - это модуль, позволяющий определять своё положение в пространстве, например, по углам Эйлера: «крен», «тангаж» и «курс». Помимо углов Эйлера, с модуля можно получать следующие данные: кажущееся угловое ускорение, истинное угловое ускорение, угловую скорость, индукцию магнитного поля, кватернионы и температуру. Все параметры кроме температуры и кватернионов выводятся для трёх осей X, Y и Z. Положение и направление осей указано на плате модуля.

Спецификация:

- Чип BMX055
- Питание модуля: 3,3 В или 5 В (оба напряжения входят в диапазон допустимых значений).
- Диапазоны измерений:
  - акселерометра: ±2g, ±4g, ±8g, ±16g, где g - ускорение свободного падения = 9,81 м/с^2.
  - гироскопа: ±125 °/с, ±250 °/с, ±500 °/с, ±1000 °/с, ±2000 °/с.
  - магнитометра: ±1300 мкТл для осей XY, ±2500 мкТл для оси Z.
- Максимальная чувствительность:
  - акселерометра: 9,5 * 10^{-3} м/с^2.
  - гироскопа: 3,8 * 10^{-3} °/с.
  - магнитометра: 625 мкГс.
- Частота обновления фильтрованных данных (в Гц):
  - акселерометра: 8, 16, 31, 63, 125, 250, 500, 1000.
  - гироскопа: 12, 23, 32, 64, 47, 116, 230.
  - магнитометра: 2, 6, 8, 10, 15, 20, 25, 30.
- Входной уровень «0» на шине I2C: -0,3 ... 0,3 * Vcc В.
- Входной уровень «1» на шине I2C: 0,7 * Vcc ... Vcc + 0,3 В.
- Рабочая температура: -40 ... 85 °С.
- Габариты модуля 30х30 мм.

Все модули линейки "Trema" выполнены в одном формате



Подключение:

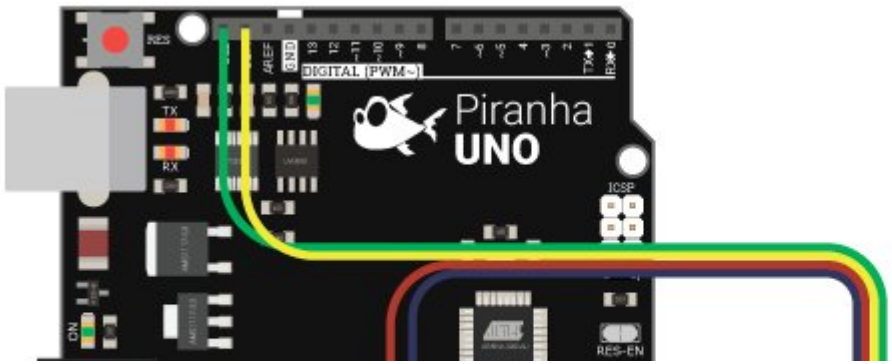
Trema-модуль IMU 9 DOF подключается к аппаратной или программной шине I2C Arduino.

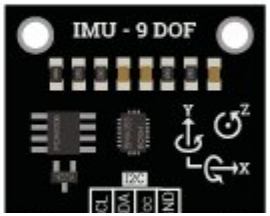
В комплекте имеется кабель для быстрого и удобного подключения модуля к колодке I2C на Trema Shield. Если на шине I2C уже имеется другое устройство, то для подключения модуля, предлагаем воспользоваться I2C Hub.

Модуль удобно подключать 3 способами, в зависимости от ситуации:

Способ - 1 : Используя проводной шлейф и Piranha UNO

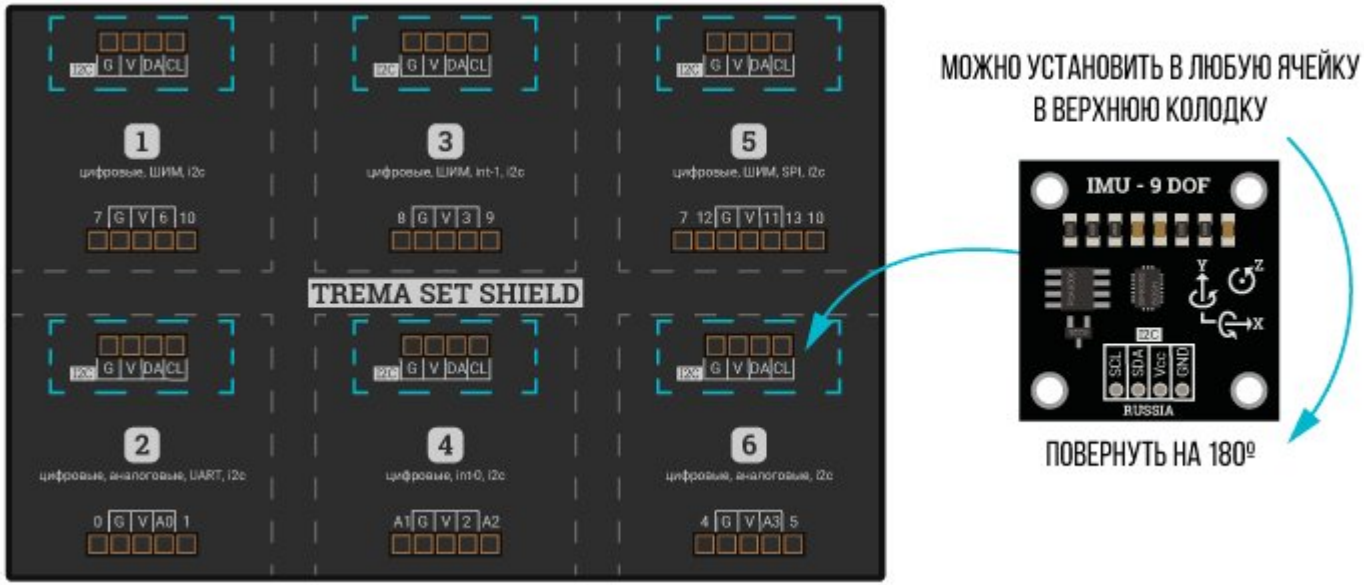
Используя провода «Папа — Мама», подключаем напрямую к контроллеру Piranha UNO.





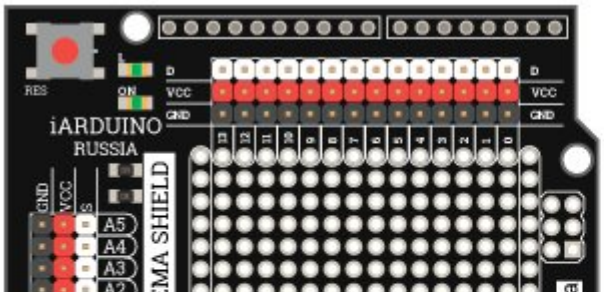
Способ - 2 : Используя Trema Set Shield

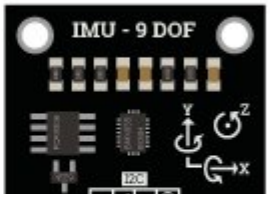
Модуль можно подключить к любому из I2C входов Trema Set Shield.



Способ - 3 : Используя проводной шлейф и Shield

Используя 4-х проводной шлейф, к Trema Shield, Trema-Power Shield, Motor Shield, Trema Shield NANO и тд.





При подключении Trema-модуля IMU 9 DOF к другим платам, например, WEMOS D1 mini или WEMOS D1 mini Pro на базе микроконтроллера ESP8266, и т.д. То перед подключением библиотеки iarduino_Position_BMX055, нужно подключить библиотеку Wire, как это описано в разделе Wiki - расширенные возможности библиотек iarduino для шины I2C.

Питание:

Входное напряжение питания 3,3 В, или 5 В постоянного тока, подаётся на выводы Vcc и GND модуля.

Подробнее о модуле:

Trema-модуль IMU 9 DOF (Inertial Measurement Unit 9 Degrees Of Freedom) - инерционное измерительное устройство на 9 степеней свободы, построен на базе чипа BMX055 (Bosch Module X) - модуль фирмы «Bosch», где X означает что в чип интегрировано несколько датчиков: (accelerometer) акселерометр, (gyroscope) гироскоп, (magnetometer) магнитометр. Все 3 датчика чипа BMX055 выдают показания по 3 осям (X,Y,Z), следовательно, с чипа считываются показания для 9 осей (9 DOF).

Специально для Trema-модуля IMU 9 DOF нами разработана библиотека iarduino_Position_BMX055, которая значительно упрощает процесс получения данных с модуля.

Библиотека способна работать как со всеми датчиками сразу, так и по отдельности. В библиотеке имеются функции аппаратного самотестирования и калибровки датчиков, есть возможность выбора диапазонов измерений, частоты обновлений и единиц измерений выводимых данных. В библиотеку интегрированы фильтры Маджвика (по умолчанию) и Махони (можно выбрать).

Подробнее про установку библиотеки читайте в нашей инструкции..

Дополнительная информация по работе с модулем:

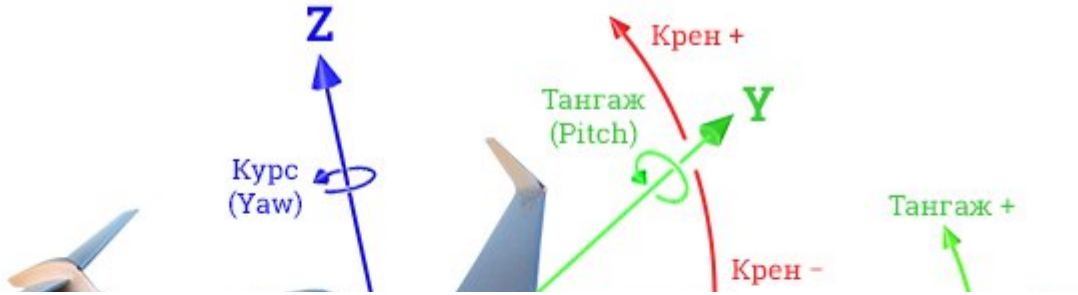
Акселерометр - датчик измеряющий кажущееся угловое ускорение, которое является геометрической разницей между истинным угловым

ускорением и ускорением силы гравитации. Показания датчика можно получать в м/с^2, или в g (количестве ускорений свободного падения). Предположим что датчик неподвижен, или движется равномерно, ось Z направлена вверх (детали на плате модуля смотрят вверх). В таком случае проекция вектора силы гравитации не оказывает влияния на оси XY, их показания равны 0 м/с^2, но оказывает влияние на ось Z и направлена в противоположную сторону (к земле), значит Z = 0 - -g = 9,81 м/с ^{2}, где 0: проекция истинного ускорения модуля на ось Z, g: ускорение свободного падения взятое со знаком минус, так как его вектор противоположен направлению оси Z. Если модуль наклонить, то влияние g на ось Z ослабнет, но увеличится на те оси в направлении которых был наклонён модуль. Таким образом, зная проекцию вектора ускорения свободного падения на оси X, Y, Z, можно вычислить положение датчика относительно поверхности земли (углы: «крен» и «тангаж»), но только если датчик неподвижен, или движется равномерно!

Гироскоп - датчик измеряющий угловую скорость вокруг собственных осей. Показания датчика можно получать в °/с, или рад/с. Данный датчик способен определять воздействие момента внешней силы вокруг своих осей. Используя эти данные можно компенсировать воздействие истинного ускорения на акселерометр, следовательно, используя акселерометр и гироскоп, получать положение этих датчиков относительно поверхности земли (углы: «крен» и «тангаж») во время их неравномерного движения.

Магнитометр - датчик измеряющий индукцию магнитного поля. Показания датчика можно получать в мГс, или в мкТл. Данный датчик способен определять своё положение в пространстве относительно магнитных полюсов земли. Добавление этого датчика к двум предыдущим даёт возможность получить последний угол Эйлера - «курс», а так же повлиять на точность определения предыдущих двух углов «крен» и «тангаж».

Показания всех датчиков можно использовать как входные данные для фильтра Маджвика, Махони, Калмана, или др., для получения кватернионов абсолютной ориентации устройства, из которых рассчитываются углы Эйлера («крен», «курс» и «тангаж»). Некоторые фильтры позволяют получать кватернионы используя данные только первых двух датчиков (без магнитометра), из которых так же можно рассчитать углы Эйлера, но угол «курс» будет не истинным, а рассчитанным, он будет указывать не на север, а на изначальное направление датчика.



Углы Эйлера - позволяют определить положение объекта в трёхмерном (евклидовом) пространстве. Для определения положения используются 3 угла - «крен», «тангаж» и «курс».

- **«Крен»** ***(Roll)*** - определяет наклон тела вокруг продольной оси X. Например крен самолёта показывает насколько градусов в бок наклонились его крылья относительно земной поверхности. Если самолёт находится параллельно земле, то крен = 0°. Если самолёт накренился (наклонился) вправо (левое крыло выше правого), то крен положительный (от 0° до 90°). Если самолёт накренился (наклонился) влево (левое крыло ниже правого), то крен отрицательный (от 0° до -90°). Если самолёт выполняет "бочку" (перевернулся), то крен ±180°.
  В библиотеке iarduino_Position_BMX055 крен привязан к оси Y, а не X, так как под креном легче понимать отклонение оси Y от горизонта земли.
  (на картинке указано стрелками "Крен+", "Крен-" от оси Y).
- **«Тангаж»** ***(Pitch)*** - определяет наклон тела вокруг поперечной оси Y. Например тангаж самолёта показывает насколько градусов поднят (или опущен) его нос относительно земной поверхности. Если самолёт находится параллельно земле, то тангаж = 0°. Если самолёт поднял нос вверх (кабрирует, взлетает), то тангаж положительный (от 0° до 90°). Если самолёт опускает нос (пикирует, приземляется), то тангаж отрицательный (от 0°, до -90°). Если самолёт выполняет "мертвую петлю" то тангаж доходит до ±180° (полёт назад вверх ногами).
  В библиотеке iarduino_Position_BMX055 тангаж привязан к оси X а не Y, так как под углом тангаж легче понимать отклонение оси X от горизонта земли.
  (на картинке указано стрелками "Тангаж+", "Тангаж-" от оси X).
- **«Курс»** ***(Yaw)*** - это самый простой для понимания угол Эйлера (еще его нарывают «Рыскание»), он определяет направление вдоль земной

поверхности. Например, для самолёта курс определяет куда самолёт летит. Если самолёт летит на север, то курс = 0°. Если самолёт отклоняется от севера влево (на запад, или против часовой стрелки если смотреть сверху) то курс положительный (от 0°, через 90° запад, до 180° юг). Если самолёт отклоняется от севера вправо (на восток, или по часовой стрелке если смотреть сверху) то курс отрицательный (от 0°, через -90° восток, до -180° юг).

Примеры:

В примерах используется разработанная нами библиотека iarduino_Position_BMX055.

Самотестирование всех датчиков модуля:

Самотестирование запускается функцией test(), которая тестирует те датчики, для которых был создан объект sensor. Самотестирование выполняется на аппаратном уровне. Функция test() не только запускает самотестирование, но и проверяет наличие ответа от датчиков.

```
/* САМОТЕСТИРОВАНИЕ ДАТЧИКОВ МОДУЛЯ */ //
                                       //
#include <Wire.h>                      //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF.
iarduino_Position_BMX055 sensor(BMX);  // Создаём объект sensor указывая параметр BMX - требуется работать со всеми датчиками
                                       // Если указать параметр BMA - то объект будет работать только с акселерометром.
                                       // Если указать параметр BMG - то объект будет работать только с гироскопом.
                                       // Если указать параметр BMM - то объект будет работать только с магнитометром.
                                       // Если указать параметр BMX - то объект будет работать со всеми датчиками сразу.
void setup(){                          //
     Serial.begin(9600);               // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/се
     while(!Serial){}                  // Ждём готовность Serial к передаче данных в монитор последовательного порта.
     Serial.println("Тест модуля ...");// Выводим надпись в монитор последовательного порта.
     sensor.begin();                   // Инициируем работу с датчиками объекта sensor.
     switch(sensor.test()){            // Получаем результат самотестирования для его сравнения с указанными ниже константами.
         case 0:          Serial.println("Аппаратное самотестирование всех датчиков успешно пройдено!"); break;
         case BMA_ERR_ID: Serial.println("Акселерометр не найден.");                                    break;
         case BMG_ERR_ID: Serial.println("Гироскоп не найден.");                                       break;
         case BMM_ERR_ID: Serial.println("Магнитометр не найден.");                                    break;
```

```
        case BMA_ERR_ST: Serial.println("Акселерометр не прошел самотестирование.");                  break;
        case BMG_ERR_ST: Serial.println("Гироскоп не прошел самотестирование.");                      break;
        case BMM_ERR_ST: Serial.println("Магнитометр не прошел самотестирование.");                   break;
        default:         Serial.println("Модуль не прошел самотестирование по неизвестной причине."); break;
    }                                   //
}                                       //
void loop(){}                           //
```

Во время самотестирования модуль должен находиться в неподвижном состоянии. Если все датчики модуля работают и их показания не выходят за допустимые пределы, то в мониторе последовательного порта появится надпись «Аппаратное самотестирование всех датчиков успешно пройдено!».

Калибровка всех датчиков модуля:

Нормальное положение Trema-модуля IMU 9 DOF в любом устройстве соответствует направлению оси Z вверх от земли (это положение при котором детали на плате модуля смотрят вверх от земли). Не всегда возможно точно расположить модуль, значит акселерометр покажет для осей XY значение отличное от 0 м/с^2, а для оси Z значение отличное от 9,81 м/с^2. При калибровке акселерометра, он скорректирует свои данные так, будто ось Z направлена строго вверх, а оси XY параллельны земле).

Калибровка акселерометра и гироскопа производится в неподвижном состоянии, а при калибровке компаса, модуль нужно вращать в разные стороны. Желательно чтоб во время калибровки компаса каждая из его осей XYZ хотя бы раз была в положении направленной к земле, от земли и параллельно земле во всех направлениях.

Калибровка компаса особенно важна, так как на его показания сильно влияют любые намагниченные предметы (моторы, сервоприводы, металлы, корпуса, провода и т.д.)

```
/* КАЛИБРОВКА ВСЕХ ДАТЧИКОВ МОДУЛЯ */  //
                                        //
#include <Wire.h>                       //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF.
iarduino_Position_BMX055 sensor(BMX);  // Создаём объект sensor указывая что требуется работать со всеми датчиками модуля.
```

```
                                    // Если указать параметр BMA - то объект будет работать только с акселерометром.
                                    // Если указать параметр BMG - то объект будет работать только с гироскопом.
                                    // Если указать параметр BMM - то объект будет работать только с магнитометром.
                                    // Если указать параметр BMX - то объект будет работать со всеми датчиками сразу.
void setup(){                       //
    Serial.begin(9600);             // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/се
    while(!Serial){}                // Ждём готовность Serial к передаче данных в монитор последовательного порта.
    sensor.begin();                 // Инициируем работу с датчиками объекта sensor.
//  Выводим подсказку по калибровке:  // Чем выше диапазон и частота, тем ниже точность.
    Serial.println("КАЛИБРОВКА:");  //
    Serial.println("Установите модуль осью Z вверх.");
    Serial.println("Модуль должен быть неподвижен. Калибровка начнётся через 10 сек.");
    delay(10000);                   //
//  Неподвижная калибровка:         //
    Serial.println("Акселерометр...");//
    sensor.setFastOffset(BMA);      // После калибровки, оси X и Y будут показывать 0.0 м/с², а ось Z 9.81 м/с².
    Serial.println("Гироскоп...");  //
    sensor.setFastOffset(BMG);      // После калибровки, угловая скорость по всем осям будет равна 0.0 °/с.
    Serial.println("Начинаем калибровку магнитометра...");
    Serial.println("Вращайте модуль так чтоб все его оси побывали во всех направлениях");
    Serial.println("Калибровка закончится через 30 сек."); // Время можно изменить.
//  Калибровка во время вращения:   //
    uint32_t i = millis();          // Запоминаем время начала калибровки.
    while( (millis()-i) < 30000 ){  // В течении 30 секунд... Время можно изменить на любое, по вашему желанию.
        sensor.setFastOffset(BMM);  // обращаемся к функции setFastOffset(BMM)
    }                               // пока пользователь вращает модуль.
    Serial.println("Калибровка магнитометра завершена.");
//  Следующие строки не обязательны:  //
    float data[3];                  // Массив для получения калибровочных коэффициентов.
    sensor.getFastOffset(data);     // Получаем калибровочные коэффициенты магнитометра.
}                                   //
void loop(){}                       // Теперь можно читать данные.
```

В этом примере сначала выполняется калибровка акселерометра в неподвижном состоянии, затем выполняется калибровка гироскопа, так же в неподвижном состоянии, после чего выполняется калибровка магнитометра при которой модуль вращается во всех направлениях.

Две последние строки кода setup позволяют сохранить калибровочные коэффициенты магнитометра в массив data, эти калибровочные значения можно использовать вместо калибровки магнитометра, при условии, что он постоянно находится в одном и том же устройстве (на него не влияют новые моторы, сервоприводы, магниты, провода и т.д.).

Предположим что в массив data записались значения: -12.5, 63.7, -10.6, тогда калибровка будет нужна только акселерометру и гироскопу, а для магнитометра мы будем указывать уже известные калибровочные данные. Тогда код будет выглядеть следующим образом:

```
/* КАЛИБРОВКА БЕЗ МАГНИТОМЕТРА */      //
                                       //
#include <Wire.h>                      //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF.
iarduino_Position_BMX055 sensor(BMX);  // Создаём объект sensor указывая что требуется работать со всеми датчиками модуля.
                                       // Если указать параметр BMA - то объект будет работать только с акселерометром.
                                       // Если указать параметр BMG - то объект будет работать только с гироскопом.
                                       // Если указать параметр BMM - то объект будет работать только с магнитометром.
                                       // Если указать параметр BMX - то объект будет работать со всеми датчиками сразу.
void setup(){                          //
     Serial.begin(9600);               // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/се
     while(!Serial){}                  // Ждём готовность Serial к передаче данных в монитор последовательного порта.
     sensor.begin();                   // Инициируем работу с датчиками объекта sensor.
//   Выводим подсказку по калибровке:  // Чем выше диапазон и частота, тем ниже точность.
     Serial.println("КАЛИБРОВКА:");    //
     Serial.println("Установите модуль осью Z вверх.");
     Serial.println("Модуль должен быть неподвижен. Калибровка начнётся через 10 сек.");
     delay(10000);                     //
//   Выполняем калибровку:             //
     Serial.println("Акселерометр...");//
     sensor.setFastOffset(BMA);        // После калибровки, оси X и Y будут показывать 0.0 м/с², а ось Z 9.81 м/с².
```

```
    Serial.println("Гироскоп...");    //
    sensor.setFastOffset(BMG);        // После калибровки, угловая скорость по всем осям будет равна 0.0 °/с.
    Serial.println("Магнитометр..."); //
    float data[3]={-12.5,63.7,-10.6}; // Значения полученные из предыдущего примера.
    sensor.setFastOffset(data);       // Указываем массив с калибровочными коэффициентами вместо выполнения калибровки.
    Serial.println("Готово.");        //
}                                     //
void loop(){}                         // Теперь можно читать данные.
```

В следующих примерах функция выполнения калибровки setFastOffset() закомментирована и не выводятся подсказки о её начале, но в реальных устройствах калибровку выполнять нужно!

Определение положения модуля в пространстве:

```
/* ОПРЕДЕЛЕНИЕ ПОЛОЖЕНИЯ МОДУЛЯ */     //
                                       //
#include <Wire.h>                      //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF.
iarduino_Position_BMX055 sensor(BMX);  // Создаём объект sensor указывая что требуется работать со всеми датчиками модуля.
                                       // Если указать параметр BMA - то объект будет работать только с акселерометром.
                                       // Если указать параметр BMG - то объект будет работать только с гироскопом.
                                       // Если указать параметр BMM - то объект будет работать только с магнитометром.
                                       // Если указать параметр BMX - то объект будет работать со всеми датчиками сразу.
void setup(){                          //
    Serial.begin(9600);                // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/се
    while(!Serial){}                   // Ждём готовность Serial к передаче данных в монитор последовательного порта.
    sensor.begin();                    // Инициируем работу с датчиками объекта sensor.
//  sensor.setScale(BMA_4G);           // Меняем диапазон измерений акселерометра до ±4G.          Возможные значения: BMA_2G
//  sensor.setScale(BMG_500DPS);       // Меняем диапазон измерений гироскопа до ±500°/с.          Возможные значения: BMG_125
//  sensor.setScale(BMM_ENHANCED);     // Меняем количество выборок магнитометра на BMM_ENHANCED.  Возможные значения: BMM_LOW
//  sensor.setBandwidths(BMA_125Hz);   // Меняем частоту обновления данных акселерометра на 125Гц. Возможные значения: BMA_8Hz
//  sensor.setBandwidths(BMG_64Hz);    // Меняем частоту обновления данных гироскопа     на 64Гц.  Возможные значения: BMG_12H
```

```
//   sensor.setBandwidths(BMM_20Hz);   // Меняем частоту обновления данных магнитометра  на 20Гц.  Возможные значения: BMM_2Hz
//   sensor.setFastOffset(BMA);        // Выполняем калибровку акселерометра после установки нового диапазона измерений или ча
//   sensor.setFastOffset(BMG);        // Выполняем калибровку гироскопа после установки нового диапазона измерений или частот
//   sensor.setFastOffset(BMM);        // Выполняем калибровку магнитометра после установки нового диапазона измерений или час
}                                      //
void loop(){                           //
     sensor.read();                    // Функция read() читает данные того датчика, для которого был создан объект sensor.
                                       // Для объекта работающего со всеми датчиками сразу, функция read() может принять
                                       // один из параметров: BMX_DEG, BMX_RAD, BMX_M_S, BMX_G. Если параметра нет, то использ
                                       // sensor.read(BMX_DEG); - читать углы Эйлера в градусах (по умолчанию).
                                       // sensor.read(BMX_RAD); - читать углы Эйлера в радианах.
                                       // sensor.read(BMX_M_S); - читать истинное ускорение в м/с².
                                       // sensor.read(BMX_G  ); - читать истинное ускорение в количестве g.
                                       // Так как объект sensor был создан для работы со всем датчиками,
                                       // то функция read() может принять параметр любого датчика:
                                       // акселерометра (BMA_M_S, BMA_G, BMA_DEG, BMA_RAD), гироскопа (BMG_DEG_S, BMG_RAD_S) и
                                       // Данные прочитанные функцией read() сохраняются в переменных axisX, axisY, axisZ и te
     Serial.print("КУРС=");   Serial.print(sensor.axisZ); Serial.print(", ");
     Serial.print("ТАНГАЖ="); Serial.print(sensor.axisX); Serial.print(", ");
     Serial.print("КРЕН=");   Serial.print(sensor.axisY); Serial.print("\r\n");
}
```

Если в начале данного скетча объявить константу BMX055_DISABLE_BMM, то библиотека не будет работать с магнитометром. В таком случае углы Эйлера будут получены только от акселерометра и гироскопа, при этом угол «курс» будет указывать не на север, а на то направление, куда был ориентирован модуль при старте:

```
/* ОПРЕДЕЛЕНИЕ ПОЛОЖЕНИЯ МОДУЛЯ */     // Положение модуля будет рассчитано без данных от магнитометра.
                                       //
#define BMX055_DISABLE_BMM             // Не использовать магнитометр. Эта строка является единственным отличием кода данного
#include <Wire.h>                      //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF.
```

```
iarduino_Position_BMX055 sensor(BMX);  // Создаём объект sensor указывая что требуется работать со всеми датчиками модуля.
                                       // Если указать параметр BMA - то объект будет работать только с акселерометром.
                                       // Если указать параметр BMG - то объект будет работать только с гироскопом.
                                       // Если указать параметр BMM - то объект будет работать только с магнитометром.
                                       // Если указать параметр BMX - то объект будет работать со всеми датчиками сразу.
void setup(){                          //
     Serial.begin(9600);               // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/се
     while(!Serial){}                  // Ждём готовность Serial к передаче данных в монитор последовательного порта.
     sensor.begin();                   // Инициируем работу с датчиками объекта sensor.
//   sensor.setScale(BMA_4G);          // Меняем диапазон измерений акселерометра до ±4G.          Возможные значения: BMA_2G
//   sensor.setScale(BMG_500DPS);      // Меняем диапазон измерений гироскопа до ±500°/с.          Возможные значения: BMG_125
//   sensor.setScale(BMM_ENHANCED);    // Меняем количество выборок магнитометра на BMM_ENHANCED.  Возможные значения: BMM_LOW
//   sensor.setBandwidths(BMA_125Hz);  // Меняем частоту обновления данных акселерометра на 125Гц. Возможные значения: BMA_8Hz
//   sensor.setBandwidths(BMG_64Hz);   // Меняем частоту обновления данных гироскопа     на 64Гц.  Возможные значения: BMG_12H
//   sensor.setBandwidths(BMM_20Hz);   // Меняем частоту обновления данных магнитометра  на 20Гц.  Возможные значения: BMM_2Hz
//   sensor.setFastOffset(BMA);        // Выполняем калибровку акселерометра после установки нового диапазона измерений или ча
//   sensor.setFastOffset(BMG);        // Выполняем калибровку гироскопа после установки нового диапазона измерений или частот
//   sensor.setFastOffset(BMM);        // Выполняем калибровку магнитометра после установки нового диапазона измерений или час
}                                      //
void loop(){                           //
     sensor.read();                    // Функция read() читает данные того датчика, для которого был создан объект sensor.
                                       // Для объекта работающего со всеми датчиками сразу, функция read() может принять
                                       // один из параметров: BMX_DEG, BMX_RAD, BMX_M_S, BMX_G. Если параметра нет, то исполь
                                       // sensor.read(BMX_DEG); - читать углы Эйлера в градусах (по умолчанию).
                                       // sensor.read(BMX_RAD); - читать углы Эйлера в радианах.
                                       // sensor.read(BMX_M_S); - читать истинное ускорение в м/с².
                                       // sensor.read(BMX_G  ); - читать истинное ускорение в количестве g.
                                       // Так как объект sensor был создан для работы со всем датчиками,
                                       // то функция read() может принять параметр любого датчика:
                                       // акселерометра (BMA_M_S, BMA_G, BMA_DEG, BMA_RAD), гироскопа (BMG_DEG_S, BMG_RAD_S) и
                                       // Данные прочитанные функцией read() сохраняются в переменных axisX, axisY, axisZ и te
     Serial.print("КУРС=");   Serial.print(sensor.axisZ); Serial.print(", ");
     Serial.print("ТАНГАЖ="); Serial.print(sensor.axisX); Serial.print(", ");
```

```
     Serial.print("КРЕН=");   Serial.print(sensor.axisY); Serial.print("\r\n");
}
```

Код данного скетча отличается от предыдущего только наличием строки #define BMX055_DISABLE_BMM, в начале скетча.

Получение данных с каждого датчика по отдельности в одном скетче:

```
/* ОТДЕЛЬНОЕ ЧТЕНИЕ ВСЕХ ДАТЧИКОВ */   //
                                       //
#include <Wire.h>                      //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF.
iarduino_Position_BMX055 sensorA(BMA); // Создаём объект sensorA указывая что ему требуется работать только с акселерометром.
iarduino_Position_BMX055 sensorG(BMG); // Создаём объект sensorG указывая что ему требуется работать только с гироскопом.
iarduino_Position_BMX055 sensorM(BMM); // Создаём объект sensorM указывая что ему требуется работать только с магнитометром.
                                       // Имена создаваемых объектов должны отличаться!
void setup(){                          //
     Serial.begin(9600);               // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/се
     while(!Serial){}                  // Ждём готовность Serial к передаче данных в монитор последовательного порта.
     sensorA.begin();                  // Инициируем работу с датчиком объекта sensorA - это акселерометр.
     sensorG.begin();                  // Инициируем работу с датчиком объекта sensorG - это гироскоп.
     sensorM.begin();                  // Инициируем работу с датчиком объекта sensorM - это магнитометр.
//   sensorA.setScale(BMA_4G);         // Меняем диапазон измерений акселерометра до ±4G.          Возможные значения: BMA_2G
//   sensorG.setScale(BMG_500DPS);     // Меняем диапазон измерений гироскопа до ±500°/с.          Возможные значения: BMG_125
//   sensorM.setScale(BMM_ENHANCED);   // Меняем количество выборок магнитометра на BMM_ENHANCED.  Возможные значения: BMM_LOW
//   sensorA.setBandwidths(BMA_125Hz); // Меняем частоту обновления данных акселерометра на 125Гц. Возможные значения: BMA_8Hz
//   sensorG.setBandwidths(BMG_64Hz);  // Меняем частоту обновления данных гироскопа     на 64Гц.  Возможные значения: BMG_12H
//   sensorM.setBandwidths(BMM_20Hz);  // Меняем частоту обновления данных магнитометра  на 20Гц.  Возможные значения: BMM_2Hz
//   sensorA.setFastOffset();          // Выполняем калибровку акселерометра после установки нового диапазона измерений или ча
//   sensorG.setFastOffset();          // Выполняем калибровку гироскопа после установки нового диапазона измерений или частот
//   sensorM.setFastOffset();          // Выполняем калибровку магнитометра после установки нового диапазона измерений или час
}                                      //
void loop(){                           //
```

```
                                    // Функция read() читает данные того датчика, для которого был создан объект.
                                    // Так как мы создали 3 разных объекта для работы с разными датчиками, то
                                    // данные читаемые функцией read() будут зависеть от объекта для которого она вызвана.
                                    // Функцию read() можно вызвать с параметром, выбрав единицы измерений выводимых данных
                                    // Если функция read вызывается без параметра, то данные выводятся в единицах по умолча
    sensorA.read();                 // sensorA.read(BMA_M_S);   читать кажущееся угловое ускорение в м/с² (по умолчанию).
                                    // sensorA.read(BMA_G);     читать кажущееся угловое ускорение в g.
                                    // sensorA.read(BMA_DEG);   читать углы «крен» и «тангаж» в градусах.
                                    // sensorA.read(BMA_RAD);   читать углы «крен» и «тангаж» в радианах.
    sensorG.read();                 // sensorG.read(BMG_DEG_S); читать угловую скорость в °/с. (по умолчанию).
                                    // sensorG.read(BMG_RAD_S); читать угловую скорость в рад/с.
    sensorM.read();                 // sensorM.read(BMM_MG);    читать индукцию магнитного поля в мГс. (по умолчанию).
                                    // sensorM.read(BMM_MCT);   читать индукцию магнитного поля в мкТл.
                                    // Данные прочитанные функцией read() сохраняются в переменных axisX, axisY, axisZ и te
                                    // Значение этих переменных у каждого объекта своё.
    Serial.print("АКСЕЛЕРОМЕТР(");
    Serial.print("X="); Serial.print(sensorA.axisX); Serial.print(", ");
    Serial.print("Y="); Serial.print(sensorA.axisY); Serial.print(", ");
    Serial.print("Z="); Serial.print(sensorA.axisZ); Serial.print(" м/с²) ");
    Serial.print("ГИРОСКОП(");
    Serial.print("X="); Serial.print(sensorG.axisX); Serial.print(", ");
    Serial.print("Y="); Serial.print(sensorG.axisY); Serial.print(", ");
    Serial.print("Z="); Serial.print(sensorG.axisZ); Serial.print(" °/c) ");
    Serial.print("МАГНИТОМЕТР(");
    Serial.print("X="); Serial.print(sensorM.axisX); Serial.print(", ");
    Serial.print("Y="); Serial.print(sensorM.axisY); Serial.print(", ");
    Serial.print("Z="); Serial.print(sensorM.axisZ); Serial.print(" мГс)\r\n");
}
```

В этом скетче мы создали 3 объекта (sensorA, sensorG и sensorM), указав каждому объекту работать со своим датчиком. Далее для каждого объекта была вызвана функция read(), которая сохранила данные в переменных (axisX, axisY, axisZ и temp) для своего объекта.

Получение данных с акселерометра:

```
/* ЧТЕНИЕ ДАННЫХ АКСЕЛЕРОМЕТРА     */  //
                                       //
#include <Wire.h>                      //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF.
iarduino_Position_BMX055 sensor(BMA);  // Создаём объект sensor указывая что ему требуется работать только с акселерометром.
                                       // Если указать параметр BMA - то объект будет работать только с акселерометром.
                                       // Если указать параметр BMG - то объект будет работать только с гироскопом.
                                       // Если указать параметр BMM - то объект будет работать только с магнитометром.
                                       // Если указать параметр BMX - то объект будет работать со всеми датчиками сразу.
                                       //
void setup(){                          //
     Serial.begin(9600);               // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/се
     while(!Serial){}                  // Ждём готовность Serial к передаче данных в монитор последовательного порта.
     sensor.begin();                   // Инициируем работу с акселерометром, так как именно для работы с ним создан объект se
//   sensor.setScale(BMA_4G);          // Меняем диапазон измерений до ±4G. Возможные значения: BMA_2G (по умолчанию), BMA_4G,
//   sensor.setBandwidths(BMA_125Hz);  // Меняем частоту обновления данных на 125Гц. Возможные значения: BMA_8Hz, BMA_16Hz (по
//   sensor.setFastOffset();           // Выполняем калибровку акселерометра (быструю компенсацию смещения данных) после устан
}                                      //
void loop(){                           //
     sensor.read();                    // Функция read() читает данные того датчика, для которого был создан объект.
                                       // Для объекта работающего с акселерометром, функция read() может принять
                                       // один из четырёх параметров: BMA_M_S, BMA_G, BMA_DEG, или BMA_RAD.
                                       // Если параметра нет, то используется параметр по умолчанию
                                       // sensor.read(BMA_M_S); читать кажущееся угловое ускорение в м/с² (по умолчанию).
                                       // sensor.read(BMA_G);   читать кажущееся угловое ускорение в g.
                                       // sensor.read(BMA_DEG); читать углы «крен» и «тангаж» в градусах.
                                       // sensor.read(BMA_RAD); читать углы «крен» и «тангаж» в радианах.
                                       // Данные прочитанные функцией read() сохраняются в переменных axisX, axisY, axisZ и te
     Serial.print("X="); Serial.print(sensor.axisX); Serial.print(", ");
     Serial.print("Y="); Serial.print(sensor.axisY); Serial.print(", ");
```

```
    Serial.print("Z="); Serial.print(sensor.axisZ); Serial.print(" м/с² \r\n");
}
```

Этот скетч выводит только данные кажущегося углового ускорения по осям XYZ в монитор последовательного порта. Данный скетч идентичен предыдущему, за исключением того, что в нём исключены все строки в которых присутствовали объекты sensorG и sensorM (объявленные для работы с гироскопом и магнитометром). Аналогичным образом можно вывести показания только гироскопа, или только магнитометра.

Получение данных с акселерометра с увеличением его диапазона измерений:

По умолчанию акселерометр измеряет данные кажущегося углового ускорения в диапазоне ±2g, где g - ускорение свободного падения = 9.81 м/с². Если модуль работает с большими истинными ускорениями, то диапазон измерений акселерометра можно расширить вызвав функцию setScale() с указанием требуемого диапазона: BMA_2G = ±2g (по умолчанию), BMA_4G = ±4g, BMA_8G = ±8g, BMA_16G = ±16g.

```
/* ЧТЕНИЕ ДАННЫХ АКСЕЛЕРОМЕТРА     */  // Акселерометр будет работать в диапазоне ±39.24 м/с².
                                        //
#include <iarduino_Position_BMX055.h>   // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF.
iarduino_Position_BMX055 sensor(BMA);   // Создаём объект sensor указывая что ему требуется работать только с акселерометром.
                                        //
void setup(){                           //
    Serial.begin(9600);                 // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/се
    while(!Serial){}                    // Ждём готовность Serial к передаче данных в монитор последовательного порта.
    sensor.begin();                     // Инициируем работу с акселерометром, так как именно для работы с ним создан объект se
    sensor.setScale(BMA_4G);            // Указываем акселерометру производить измерения в новом диапазоне ±4g, где g=9.81 м/с²
//  sensor.setBandwidths(BMA_125Hz);    // Меняем частоту обновления данных на 125Гц. Возможные значения: BMA_8Hz, BMA_16Hz (по
    sensor.setFastOffset();             // Выполняем калибровку акселерометра (быструю компенсацию смещения данных) после устан
}                                       //
void loop(){                            //
    sensor.read();                      // Читаем данные акселерометра в единицах по умолчанию (м/с²).
    Serial.print("X="); Serial.print(sensor.axisX); Serial.print(", ");
    Serial.print("Y="); Serial.print(sensor.axisY); Serial.print(", ");
```

```
    Serial.print("Z="); Serial.print(sensor.axisZ); Serial.print(" м/с²)\r\n");
}
```

Код данного скетча отличается от предыдущего только тем, что в конце кода setup разкомментирована строка вызова функции setScale() и setFastOffset(), устанавливающие новый диапазон измерений и калибровку. Чем шире установленный диапазон измерений, тем меньше точность показаний. Калибровка должна производится при неподвижном датчике. Аналогичным образом можно изменить диапазон измерений гироскопа и магнитометра (см. описании функции setScale).

Получение кватернионов для программ визуализации:

```
/* ЧТЕНИЕ КВАТЕРНИОНОВ */              //
                                       //
#define BMX055_DISABLE_BMM             // Не использовать магнитометр. Курс будет ориентирован на начальное положение модуля
#include <Wire.h>                      //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF.
iarduino_Position_BMX055 sensor(BMX);  // Создаём объект sensor указывая что требуется работать со всеми датчиками модуля.
                                       // Если указать параметр BMA - то объект будет работать только с акселерометром.
                                       // Если указать параметр BMG - то объект будет работать только с гироскопом.
                                       // Если указать параметр BMM - то объект будет работать только с магнитометром.
                                       // Если указать параметр BMX - то объект будет работать со всеми датчиками сразу.
void setup(){                          //
    Serial.begin(9600);                // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/се
    while(!Serial){}                   // Ждём готовность Serial к передаче данных в монитор последовательного порта.
    sensor.begin();                    // Инициируем работу с датчиками объекта sensor.
//  sensor.setScale(BMA_4G);           // Меняем диапазон измерений акселерометра до ±4G.          Возможные значения: BMA_2G
//  sensor.setScale(BMG_500DPS);       // Меняем диапазон измерений гироскопа до ±500°/с.          Возможные значения: BMG_125
//  sensor.setScale(BMM_ENHANCED);     // Меняем количество выборок магнитометра на BMM_ENHANCED.  Возможные значения: BMM_LOW
//  sensor.setBandwidths(BMA_125Hz);   // Меняем частоту обновления данных акселерометра на 125Гц. Возможные значения: BMA_8Hz
//  sensor.setBandwidths(BMG_64Hz);    // Меняем частоту обновления данных гироскопа     на 64Гц.  Возможные значения: BMG_12H
//  sensor.setBandwidths(BMM_20Hz);    // Меняем частоту обновления данных магнитометра  на 20Гц.  Возможные значения: BMM_2Hz
//  sensor.setFastOffset(BMA);         // Выполняем калибровку акселерометра после установки нового диапазона измерений или ча
```

```
//   sensor.setFastOffset(BMG);        // Выполняем калибровку гироскопа после установки нового диапазона измерений или частот
//   sensor.setFastOffset(BMM);        // Выполняем калибровку магнитометра после установки нового диапазона измерений или час
}                                      //
void loop(){                           //
     sensor.read();                    // Функция read() читает данные того датчика, для которого был создан объект.
                                       // В данном скетче параметр функции read() не имеет ни какого значения, так как
                                       // он влияет только на единицы измерений данных в переменных axisX, axisY, axisZ.
    Serial.print(sensor.q1);           // Выводим кватернион q1.
    Serial.print(",");                 // разделяем запятой.
    Serial.print(sensor.q2);           // Выводим кватернион q2.
    Serial.print(",");                 // разделяем запятой.
    Serial.print(sensor.q3);           // Выводим кватернион q3.
    Serial.print(",");                 // разделяем запятой.
    Serial.println(sensor.q4);         // Выводим кватернион q4 и завершаем строку.
}                                      //
```

Данный скетч использовался в видеообзоре.

В библиотеке iarduino_Position_BMX055 имеется больше примеров.

Описание основных функций библиотеки:

Данная библиотека может использовать как аппаратную, так и программную реализацию шины I2C.
О том как выбрать тип шины I2C рассказано в статье Wiki - расширенные возможности библиотек iarduino для шины I2C.

Подключение библиотеки:

```
#include <Wire.h>                      // Подключаем библиотеку Wire
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF
iarduino_Position_BMX055 sensor(BMX);  // Создаём объект sensor указывая датчик, данные которого Вы хотите получать (BMX - пол
                                       // BMA - получать данные только с акселерометра
                                       // BMG - получать данные только с гироскопа
```

```
                                    // BMM - получать данные только с магнитометра
                                    // BMX - получать данные со всех датчиков модуля
```

При создании объекта требуется указать, с каким датчиком (или датчиками) он будет работать. Все функции и методы созданного объекта будут применяться только к тому датчику (датчикам) для которого он был создан.

Допускается создание объекта с одним из 4 параметров:

- `BMA` - объект будет работать только с акселерометром.
- `BMG` - объект будет работать только с гироскопом.
- `BMM` - объект будет работать только с магнитометром.
- `BMX` - объект будет работать со всеми датчиками модуля.

Если требуется работать с разными датчиками по отдельности, допускается создание нескольких объектов.

Освобождение памяти:

При использовании данной библиотеки можно освободить память за счёт отключения кода неиспользуемых датчиков модуля. Это нужно делать только если у Вас не хватает памяти программ. Память освобождается если объявить в начале скетча следующие константы:

- `BMX055_DISABLE_BMA` - не использовать код для работы с акселерометром
- `BMX055_DISABLE_BMG` - не использовать код для работы с гироскопом
- `BMX055_DISABLE_BMM` - не использовать код для работы с магнитометром

```
// Если Вы используете данные только магнитометра (например, создаёте компас), то освободить память можно за счёт неиспользуем
#define BMX055_DISABLE_BMA             // Не использовать код для работы с акселерометром
#define BMX055_DISABLE_BMG             // Не использовать код для работы с гироскопом
#include <Wire.h>                      //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF
iarduino_Position_BMX055 sensor(BMM);  // Создаём объект sensor, указывая что он будет работать с магнитометром (BMM)
```

Смена фильтра при получении углов Эйлера:

Углы Эйлера («крен», «тангаж» и «курс») рассчитываются из кватернионов, а они вычисляются фильтром. При объявлении объекта с атрибутом `BMX` (объект работает со всеми датчиками), подключается фильтр Маджвика (по умолчанию), но если Вы желаете использовать фильтр Махони, то нужно отключить код фильтра Маджвика и подключить код фильтра Махони:

```
// Вместо фильтра Маджвика (по умолчанию) указываем использовать фильтр Махони:
#define BMX055_DISABLE_MADGWICK        // Не использовать код фильтра Маджвика
#define BMX055_ENABLE_MAHONY           // Использовать код фильтра Махони
#include <Wire.h>                      //
#include <iarduino_Position_BMX055.h>  // Подключаем библиотеку iarduino_Position_BMX055 для работы с Trema-модулем IMU 9 DOF
iarduino_Position_BMX055 sensor(BMX);  // Создаём объект sensor, указывая что он будет работать со всеми датчиками (BMX)
```

Функция begin();

- Назначение: Инициализация работы с выбранным датчиком (датчиками) модуля.
- Синтаксис: begin([КАЛИБРОВКА]);
- Параметры:
  - КАЛИБРОВКА - не обязательный параметр (true или false) указывающий на необходимость выполнения калибровки акселерометра и/или гироскопа при инициализации.
- Возвращаемые значения: bool - результат инициализации датчика (true или false).
- Примечание:
  - Функцию необходимо вызывать до обращения к остальным функциям и методам объекта.
  - Функцию достаточно вызвать 1 раз в коде setup.
  - Вызов функции без параметра аналогичен вызову с параметром false.
  - Калибровку можно произвести функцией setFastOffset() в любое время выполнения скетча.
  - При калибровке акселерометра и гироскопа модуль должен быть неподвижен.
  - Подробнее о калибровке рассказано ниже в описании функции setFastOffset().

- Пример:

```
sensor.begin(); // Инициализация датчика с которым работает объект sensor (без выполнения калибровки)
```

Функция test();

- Назначение: Выполнение аппаратного самотестирования датчика (датчиков) модуля.
- Синтаксис: test();
- Параметры: нет.
- Возвращаемые значения: зависят от того, для какого датчика(ов) создан объект.
  - 0 - аппаратное самотестирование датчика (датчиков) успешно пройдено.
  - BMA_ERR_ID - акселерометр не найден.
  - BMG_ERR_ID - гироскоп не найден.
  - BMM_ERR_ID - магнитометр не найден.
  - BMA_ERR_ST - акселерометр не прошел самотестирование.
  - BMG_ERR_ST - гироскоп не прошел самотестирование.
  - BMM_ERR_ST - магнитометр не прошел самотестирование.
- Примечание:
  - Функция инициирует выполнение аппаратного самотестирования датчика (датчиков) модуля для которого был создан объект.
  - В процессе самотестирования датчика (датчиков), модуль должен быть неподвижен.
  - При тестировании проверяется не только работоспособность датчика, но и корректность выдаваемых им показаний.
  - Выполнение функции занимает некоторое время, которое зависит от тестируемого датчика.
  - Если объект работает со всеми датчиками, то и самотестирование будет запущено для всех датчиков.
  - В разделе «Примеры» настоящей статьи есть скетч использования этой функции.
- Пример:

```
if(sensor.test()){Serial.print("Err");} // Выполнение аппаратного самотестирования датчика с которым работает объект sensor
                                        // При выявлении проблем с датчиком, на экране появится надпись "Err".
```

Функция read();

- Назначение: Выполнение чтения показаний датчика (датчиков) модуля.
- Синтаксис: read([ЕДИНИЦЫ_ИЗМЕРЕНИЯ]);
- Параметр: зависит от того, с каким датчиком работает объект.
  - Если объект работает с акселерометром:
    - BMA_M_S - выводить кажущееся угловое ускорение в м/с² (по умолчанию).
    - BMA_G - выводить кажущееся угловое ускорение в количестве ускорений свободного падения (g).
    - BMA_DEG - выводить углы «крен» и «тангаж» в градусах (для неподвижного или равноускоренного датчика).
    - BMA_RAD - выводить углы «крен» и «тангаж» в радианах (для неподвижного или равноускоренного датчика).
  - Если объект работает с гироскопом:
    - BMG_DEG_S - выводить угловую скорость в °/с (по умолчанию).
    - BMG_RAD_S - выводить угловую скорость в рад/с.
  - Если объект работает с магнитометром:
    - BMM_MG - выводить индукцию магнитного поля в мГс (по умолчанию).
    - BMM_MCT выводить индукцию магнитного поля в мкТл.
  - Если объект работает со всеми датчиками:
    - BMX_DEG выводить положение модуля в пространстве в градусах (по умолчанию).
    - BMX_RAD выводить положение модуля в пространстве в радианах.
    - BMX_M_S выводить истинное ускорение модуля в м/с².
    - BMX_G выводить истинное ускорение модуля в количестве ускорений свободного падения (g).
    - Функция может принимать любой параметр доступный для объекта работающего с акселерометром, гироскопом или магнитометром.
- Возвращаемые значения: bool - успех чтения данных (true или false).
- Примечание:
  - Функция возвращает результат успешности чтения, а прочитанные данные сохраняются в переменные: axisX, axisY, axisZ и temp.
  - Параметры функции и читаемые ей данные зависят от того, с каким датчиком работает объект.
  - Если вызвать функцию без параметра, то будет произведено чтение данных указанных для датчика по умолчанию.
  - Если функция вызвана от объекта работающего со всеми датчиками, то независимо от параметра функции, будут рассчитаны

кватернионы, которые сохраняться в переменные: q1, q2, q3, q4.
    - В разделе «Примеры» настоящей статьи есть скетчи использования этой функции.
- Пример:

```
iarduino_Position_BMX055 sensorA(BMA); // Создаём объект sensorA указывая что ему требуется работать только с акселерометром
iarduino_Position_BMX055 sensorG(BMG); // Создаём объект sensorG указывая что ему требуется работать только с гироскопом
iarduino_Position_BMX055 sensorM(BMM); // Создаём объект sensorM указывая что ему требуется работать только с магнитометром
iarduino_Position_BMX055 sensor (BMX); // Создаём объект sensor  указывая что ему требуется работать со всеми датчиками сразу
...                                    //
void loop(){                           //
  sensorA.read(BMA_M_S);               // читаем кажущееся угловое ускорение в м/с² из акселерометра, данные сохранятся в пере
  sensorG.read(BMG_RAD_S);             // читаем угловую скорость в рад/с из гироскопа, данные сохранятся в переменных axisX,
  sensorM.read(BMM_MCT);               // читаем индукцию магнитного поля в мкТл из магнитометра, данные сохранятся в переменн
  sensor.read(BMX_G);                  // читаем истинное ускорение в количестве ускорений свободного падения (g), данные сохр
}                                      //
```

Функция setScale();

- Назначение: Установка диапазона измерений датчика модуля.
- Синтаксис: setScale(ДИАПАЗОН);
- Параметр: зависит от того, с каким датчиком работает объект.
    - Если объект работает с акселерометром:
        - BMA_2G - производить измерения в диапазоне ±2 g (по умолчанию).
        - BMA_4G - производить измерения в диапазоне ±4 g.
        - BMA_8G - производить измерения в диапазоне ±8 g.
        - BMA_16G - производить измерения в диапазоне ±16 g.

          где g - ускорение свободного падения = 9.81 м/с².
    - Если объект работает с гироскопом:
        - BMG_125DPS - производить измерения в диапазоне ±125 °/с (по умолчанию).
        - BMG_250DPS - производить измерения в диапазоне ±250 °/с.

        - BMG_500DPS - производить измерения в диапазоне ±500 °/с.
        - BMG_1000DPS - производить измерения в диапазоне ±1000 °/с.
        - BMG_2000DPS - производить измерения в диапазоне ±2000 °/с.
    - Если объект работает с магнитометром:
        - BMM_LOW_PWR - экономичный режим.
        - BMM_REGULAR - обычный режим (по умолчанию).
        - BMM_ENHANCED - улучшенный режим.
        - BMM_HIGH - режим высокой точности.
    - Если объект работает со всеми датчиками:
        - Функция может принять любой из перечисленных параметров.
        - Для указания параметров нескольким датчикам, функцию вызывают для каждого датчика.
- Возвращаемые значения: нет.
- Примечание:
    - После смены диапазона измерений датчика, нужно выполнить его калибровку вызвав функцию setFastOffset().
    - Чем шире диапазон измерений, тем меньше точность показаний датчиков.
    - У магнитометра фиксированный диапазон измерений для каждой оси, по этому данная функция устанавливает один из рекомендуемых (предустановленных) режимов его работы. Чем точнее режим работы, тем точнее показания магнитометра.
- Пример:

```
                                    // если объект sensor объявлен с параметром BMA - для работы с акселерометром
sensor.setScale(BMA_16G);           // производить дальнейшие измерения акселерометра в диапазоне ±16 g.
sensor.setFastOffset();             // выполнить калибровку акселерометра после смены его диапазона измерений.
                                    // теперь чтение данных акселерометра будет производиться в новом диапазоне.
```

Функция setBandwidths();

- Назначение: Установка полосы пропускания для фильтрованных данных. Полоса пропускания влияет на то с какой скоростью датчик будет сохранять новые показания в свои регистры.
- Синтаксис: setBandwidths(ЧАСТОТА);
- Параметр: зависит от того, с каким датчиком работает объект.

- Если объект работает с акселерометром:
  - BMA_8Hz - данные обновляются с частотой 7.81 Гц.
  - BMA_16Hz - данные обновляются с частотой 15.63 Гц (по умолчанию).
  - BMA_31Hz - данные обновляются с частотой 31.25 Гц.
  - BMA_63Hz - данные обновляются с частотой 62.5 Гц.
  - BMA_125Hz - данные обновляются с частотой 125 Гц.
  - BMA_250Hz - данные обновляются с частотой 250 Гц.
  - BMA_500Hz - данные обновляются с частотой 500 Гц.
  - BMA_1000Hz - данные обновляются с частотой 1000 Гц.
- Если объект работает с гироскопом:

  Тут частота полосы пропускания отличается от частоты обновления данных.
  - BMG_12Hz - данные обновляются с частотой 100 Гц.
  - BMG_23Hz - данные обновляются с частотой 200 Гц (по умолчанию).
  - BMG_32Hz - данные обновляются с частотой 100 Гц.
  - BMG_47Hz - данные обновляются с частотой 400 Гц.
  - BMG_64Hz - данные обновляются с частотой 200 Гц.
  - BMG_116Hz - данные обновляются с частотой 1000 Гц.
  - BMG_230Hz - данные обновляются с частотой 2000 Гц.
  - BMG_523Hz - данные обновляются с частотой 2000 Гц.
- Если объект работает с магнитометром:
  - BMM_2Hz - данные обновляются с частотой 2 Гц.
  - BMM_6Hz - данные обновляются с частотой 6 Гц.
  - BMM_8Hz - данные обновляются с частотой 8 Гц.
  - BMM_10Hz - данные обновляются с частотой 10 Гц (по умолчанию).
  - BMM_15Hz - данные обновляются с частотой 15 Гц.
  - BMM_20Hz - данные обновляются с частотой 20 Гц.
  - BMM_25Hz - данные обновляются с частотой 25 Гц.
  - BMM_30Hz - данные обновляются с частотой 30 Гц.

    - Если объект работает со всеми датчиками:
        - Функция может принять любой из перечисленных параметров.
        - Для указания параметров нескольким датчикам, функцию вызывают для каждого датчика.
- Возвращаемые значения: нет.
- Примечание:
    - После изменения полосы пропускания датчика, рекомендуется выполнить его калибровку вызвав функцию setFastOffset().
    - Полоса пропускания фильтрованных данных определяет с какой скоростью датчик обновляет свои регистры данных новыми показаниями.
    - Чем выше частота обновления данных, тем чаще можно читать новые данные с датчика.
    - Если функцией read() читаются старые данные датчика (которые уже были прочитаны ранее), то функция read() вернёт false и значение переменных axisX, axisY, axisZ, и temp останется неизменным.
- Пример:

```
                                    // если объект sensor объявлен с параметром BMM - для работы с магнитометром
sensor.setBandwidths(BMM_30Hz);     // установить полосу пропускания магнитометра в 30 Гц.
                                    // теперь новые данные магнитометра можно получать до 30 раз в секунду.
```

Функция setFastOffset();

- Назначение: Калибровка - выполнение быстрой компенсации смещения данных датчика модуля.
- Синтаксис: setFastOffset([ЗНАЧЕНИЕ]);
- Параметр: зависит от того, с каким датчиком работает объект.
    - Если объект работает с акселерометром:
        - BMA - выполнить аппаратную калибровку акселерометра.
        - Вызов функции без параметра аналогичен вызову функции с параметром BMA.
    - Если объект работает с гироскопом:
        - BMG - выполнить аппаратную калибровку гироскопа.
        - Вызов функции без параметра аналогичен вызову функции с параметром BMG.
    - Если объект работает с магнитометром:

        - BMM - учесть очередные данные магнитометра для калибровки.
          Калибровка магнитометра (в отличии от акселерометра и гироскопа) выполняется программно, в процессе вращения магнитометра по всем осям, при этом нужно постоянно вызывать функцию setFastOffset(BMM), чем больше вызовов при разных положениях, тем точнее калибровка.
        - Вызов функции без параметра аналогичен вызову функции с параметром BMM.
        - BMM_RESET - сбросить калибровочные коэффициенты магнитометра.
        - Массив из 3 элементов типа float - применить калибровочные коэффициенты.
          О том как получить значения калибровочных коэффициентов после выполнения калибровки рассказано в описании функции getFastOffset().
    - Если объект работает со всеми датчиками:
        - Функция может принять любой из перечисленных параметров.
        - Для калибровки нескольких датчиков, функцию вызывают для каждого датчика.
        - Вызов функции без параметра будет проигнорирован.
- Возвращаемые значения: нет.
- Примечание:
    - Функция выполняется аппаратно для акселерометра и гироскопа, и программно для магнитометра.
    - Калибровку рекомендуется выполнить в начале скетча для тех датчиков, данные которых предполагается использовать в проекте. Так же калибровку рекомендуется выполнить после изменения диапазона измерений датчика или полосы пропускания его данных.
    - В процессе калибровки данные с датчиков не будут соответствовать действительности, а данные акселерометра и гироскопа примут истинные значения спустя 200-300 мс после завершения калибровки.
    - Калибровка акселерометра выполняется за одно обращение к функции, при этом модуль должен находиться в неподвижном состоянии, или двигаться равномерно, ось Z должна быть направлена вверх от земли (это положение при котором детали на плате модуля смотрят вверх). Кажущееся угловое ускорение оси Z будет скорректировано до ускорения свободного падения 9,81 м/с^2, а значение осей XY будет скорректировано до 0,0 м/с^2.
    - Калибровка акселерометра выполняется за одно обращение к функции, при этом модуль должен находиться в неподвижном состоянии. Угловая скорость вокруг всех осей будет скорректирована до 0,0 °/с.
    - Калибровка магнитометра выполняется многократным обращением к функции с параметром BMM, пока модуль вращается по всем осям в разных направлениях. Чем больше вызовов функции при разных положениях модуля, тем точнее будут рассчитаны

калибровочные коэффициенты. После калибровки показания индукции магнитного поля любой оси в противоположных направлениях будут одинаковы по модулю но отличны по знаку.
    - Если магнитометр калибруется повторно (после подачи питания), необходимо сначала сбросить калибровочные коэффициенты вызвав функцию с параметром BMM_RESET.
    - Калибровку магнитометра необходимо производить вдали от магнитов, за исключением тех, которые постоянно присутствуют в устройстве на котором установлен модуль.
    - В разделе «Примеры» настоящей статьи есть скетчи использования этой функции.
- Примеры:

```
                                // если объект sensor объявлен с параметром BMA или BMG
sensor.setFastOffset();         // выполняем калибровку датчика для которого создан объект sensor
```

```
                                // если объект sensor объявлен с параметром BMX - для работы со всеми датчиками.
sensor.setFastOffset(BMA);      // выполняем калибровку акселерометра.
sensor.setFastOffset(BMG);      // выполняем калибровку гироскопа.
uint32_t i = millis();          // запоминаем время начала калибровки магнитометра.
while( (millis()-i) < 30000 ){  // в течении 30 секунд...
    sensor.setFastOffset(BMM);  // выполняем калибровку магнитометра. Его нужно вращать.
}                               // время (30000 мс) можно изменить. Чем больше показаний, тем лучше.
```

```
                                // если объект sensor объявлен с параметром BMM или BMX.
float data[3]={-12.5,63.7,-10.6}; // создаём массив с известными калибровочными коэффициентами магнитометра.
sensor.setFastOffset(data);     // указываем массив вместо выполнения калибровки магнитометра.
```

Функция getFastOffset();

- Назначение: Получение калибровочных коэффициентов магнитометра.
- Синтаксис: getFastOffset(МАССИВ);
- Параметры:
    - МАССИВ - массив типа float состоящий из трёх элементов, для получения калибровочных коэффициентов магнитометра.

- Возвращаемые значения: нет.
- Примечание:
  - Калибровка магнитометра выполняется программно, что позволяет запомнить результат калибровки и использовать его вместо выполнения калибровок после каждого включения.
  - Сначала необходимо выполнить калибровку магнитометра функцией setFastOffset() с параметром BMM (или без параметра), после чего обратиться к функции getFastOffset() указав в качестве параметра массив в который функция сохранит рассчитанные калибровочные коэффициенты.
  - Полученные калибровочные коэффициенты можно сохранить в EEPROM или явно указывать в скетче, вместо калибровки магнитометра.
  - Для указания калибровочных коэффициентов вместо калибровки магнитометра используется функция setFastOffset() которой в качестве параметра указывается массив состоящий из трёх калибровочных коэффициентов типа float.
  - Калибровочные коэффициенты будут отличаться у разных модулей, у модулей установленных на разные устройства, у модулей работающих в разных диапазонах измерений и с разной полосой пропускания, а так же откалиброванных в разных точках земного шара.
- Пример:

```
float data[3];                          // Объявляем массив для получения калибровочных коэффициентов.
...                                     // Выполняем калибровку магнитометра.
sensor.getFastOffset( data );           // Получаем рассчитанные калибровочные коэффициенты.
```

Функция getFilter();

- Назначение: Вывод типа фильтра используемого для расчёта кватернионов.
- Синтаксис: getFilter();
- Параметры: нет.
- Возвращаемые значения: uint8_t - тип используемого фильтра:
  - NO_FILTER - нет подходящего фильтра.
  - FILTER_MADGWICK - используется фильтр Маджвика (по умолчанию).
  - FILTER_MADGWICK_NO_BMM - используется фильтр Маджвика без данных с магнитометра.

    - FILTER_MAHONY - используется фильтр Махони.
- Примечание:
    - Функция поддерживается только объектом, который объявлен для работы со всеми датчиками.
    - Если объявить объект с параметром BMX и вызвать функцию getFilter(), то она вернёт FILTER_MADGWICK.
    - Если в начале скетча отключить магнитометр (#define BMX055_DISABLE_BMM), то вызов функции getFilter(), вернёт FILTER_MADGWICK_NO_BMM.
    - Если в начале скетча отключить фильтр Маджвика (#define BMX055_DISABLE_MADGWICK) и подключить фильтр Махони (#define BMX055_ENABLE_MAHONY), то вызов функции getFilter(), вернёт FILTER_MAHONY.
    - Если совершить некорректные действия, например, отключить фильтр Маджвика и не подключить фильтр Махони, или подключить фильтр Махони и отключить магнитометр и т.д., то вызов функции getFilter(), вернёт NO_FILTER.
- Пример:

```
switch( sensor.getFilter() ){
    case FILTER_MADGWICK:        Serial.println("используется фильтр Маджвика (по умолчанию)." ); break;
    case FILTER_MADGWICK_NO_BMM: Serial.println("используется фильтр Маджвика без данных с магнитометра." ); break;
    case FILTER_MAHONY:          Serial.println("используется фильтр Махони." ); break;
    case NO_FILTER:              Serial.println("нет подходящего фильтра." ); break;
}
```

Переменная axisX:

- Тип: float
- Данные: показание датчика для оси X или угол «тангаж», зависит от параметра функции read().
- Обновление данных: после вызова функции read().
- Примечание: угол «тангаж» при работе только с акселерометром выдаёт значения до ±90°.

Переменная axisY:

- Тип: float
- Данные: показание датчика для оси Y или угол «крен», зависит от параметра функции read().
- Обновление данных: после вызова функции read().

- Примечание: угол «крен» принимает одинаковые значения как при работе с акселерометром, так и при работе со всеми датчиками модуля.

Переменная axisZ:

- Тип: float
- Данные: показание датчика для оси Z или угол «курс», зависит от параметра функции read().
- Обновление данных: после вызова функции read().
- Примечание: не содержит угол «курс» при работе только с акселерометром.

Переменная temp:

- Тип: float
- Данные: содержит температуру в °C и зависит от датчика с которым работает объект.
- Обновление данных: после вызова функции read().
- Примечание: не содержит температуру при работе только с гироскопом.

Дополнительные переменные q1, q2, q3, q4:

- Тип: float
- Данные: кватернионы.
- Обновление данных: после вызова функции read() и не зависят от параметра данной функции.
- Примечание: поддерживаются только объектом, который объявлен для работы со всеми датчиками.

Применение:

- Системы стабилизации, манипуляторы, роботостроение;
- БПЛА, квадрокоптеры, самолёты, подводные и надводные суда;