# РАЗРАБОТКА БИБЛИОТЕКИ ДЛЯ МИКРОЭЛЕКТРОМЕХАНИЧЕСКОЙ ИНЕРЦИАЛЬНОЙ ИЗМЕРИТЕЛЬНОЙ СИСТЕМЫ НА БАЗЕ ДАТЧИКА IMU-20948

# Часть 1. Введение в используемые библиотеки

# Pimoroni (IMU-20948)

# Библиотека Horizon для Raspberry Pi будет основана на библиотеке [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222), написанной на python и реализуещей следующие функции:

# [Инициализация датчика;](#init)

# [Настройка частоты сэмплирования акселерометра;](#set_accelerometer_sample_rate)

# [Настройка частоты сэмплирования гироскопа;](#set_gyro_sample_rate)

# [Настройка фильтра низких частот для акселерометра;](#set_gyro_low_pass)

# [Настройка фильтра низких частот для гироскопа;](#set_gyro_low_pass)

# [Настройка диапазона измерения акселерометра;](#setAccelRange)

# [Настройка диапазона измерения гироскопа;](#setGyroRange)

# [Групповое чтение гироскопа и акселлерометра;](#read_accelerometer_gyro_data)

# [Чтение данных магнитометра;](#read_magnetometer_data)

# Некоторые из перечисленных функций, такие как инициализация и чтение данных с магнитометра, определенно требуют внесения правок, что описано в [разделе 2.1](#init_explanation) и [приложении 2](#read_magnetometer_data). Остальные же планируется использовать без изменений, однако они требуют проверки на корректность работы.

# Дополнительный функционал будет позаимствован у других библиотек и имеет подводные камни, на которые следует обратить внимание при разработке библиотеки Horizon.

1. **Adafruit (IMU-20x)**

Библиотека [Adafruit](https://github.com/adafruit/Adafruit_CircuitPython_ICM20X/blob/main/adafruit_icm20x.py#L758) предоставляет следующие функции:

* Настройка частоты сэмплирования магнитометра.

Подробнее о переносе данного функционала [в разделе 2.7](#mag_data_rate).

1. **SparkFun (IMU-20948)**

Библиотека [SparkFun](https://github.com/sparkfun/Qwiic_9DoF_IMU_ICM20948_Py/blob/main/qwiic_icm20948.py#L370) предоставляет следующие функции:

* Перезагрузка датчика;
* Переход в режим сна.

Данный функционал можно перенести в разрабатываемую библиотеку без существенных изменений.

# CompFilter (MPU9250)

Библиотека [MPU-6050-9250-CompFilter](https://github.com/MarkSherstan/MPU-6050-9250-I2C-CompFilter/blob/master/RPi/main.py) реализует следующие функции:

* Вычисление углов Эйлера;
* Слияние через комплементарный фильтр.

Вопрос переноса и адаптации данного функционала в библиотеку Horizon изложен [в разделе 2.5](#эйлер_фильтры).

1. **MakerPortal (MPU9250)**

Библиотека [makerportal](https://github.com/makerportal/mpu92-calibration/blob/main/complete/mpu9250_full_calibration.py) реализует отдельную утилиту, которая обеспечивает калибровку IMU-датчика без жесткой зависимости от используемого модуля. Тем не менее, имеющийся код требует ряда правок, о чем сказано в соответствующем разделе [ниже](#калибровка).

Библиотека также предоставляет функционал записи калибровочных значений в .CSV файл для дальнейшего использования основной измерительной программой.

Подробнее о самих принципах калибровки можно прочитать на стр. 6 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg).

1. **WickedLukas (ICM-20948)**

Библиотека от [wickedlukas](https://github.com/WickedLukas/ICM20948/blob/master/ICM20948.cpp) предоставляет следующий функционал:

* Отдельное включение датчиков (акселерометра, гироскопа, термо, магнитометра);
* Отдельное чтение показаний акселерометра;
* Отдельное чтение показаний гироскопа.

О реализации раздельного считывания данных а также включения/выключения датчиков см. разделы [2.3](#раздельное_считывание) и [2.7](#вкл_выкл) соответственно.

# Итоговая сноска методов

|  |  |  |
| --- | --- | --- |
| **Имя метода** | **Функционал** | **Библиотека** |
| init | Инициализация класса, проверка датчика, настройка компонентов по умолчанию (таблицы 2-4 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg)) | [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222) |
| acc\_toggle | Включение/выключение акселерометра | [SparkFun](https://github.com/sparkfun/Qwiic_9DoF_IMU_ICM20948_Py/blob/main/qwiic_icm20948.py#L370) |
| gyro\_toggle | Включение/выключение гироскопа | [SparkFun](https://github.com/sparkfun/Qwiic_9DoF_IMU_ICM20948_Py/blob/main/qwiic_icm20948.py#L370) |
| mag\_toggle | Включение/выключение магнитометра | [SparkFun](https://github.com/sparkfun/Qwiic_9DoF_IMU_ICM20948_Py/blob/main/qwiic_icm20948.py#L370) |
| set\_accelerometer\_sample\_rate | Настройка частоты сэмплирования акселерометра (таблица 3 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg)) | [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222) |
| set\_gyro\_sample\_rate | Настройка частоты сэмплирования гироскопа  (таблица 2 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg)) | [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222) |
| magnetometer\_data\_rate | Настройка частоты сэмплирования магнитометра (таблица 4 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg)) | [Adafruit](https://github.com/adafruit/Adafruit_CircuitPython_ICM20X/blob/main/adafruit_icm20x.py#L758) |
| setAccelRange | Настройка диапазона измерений акселерометра (таблица 3 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg)) | [IMUSensor](https://github.com/niru-5/imusensor/blob/master/imusensor/MPU9250/MPU9250.py) |
| setGyroRange | Настройка диапазона измерений гироскопа  (таблица 2 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg)) | [IMUSensor](https://github.com/niru-5/imusensor/blob/master/imusensor/MPU9250/MPU9250.py) |
| set\_accelerometer\_low\_pass | Настройка фильтра низких частот акселерометра (таблица 6 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg)) | [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222) |
| set\_gyro\_low\_pass | Настройка фильтра низких частот гироскопа (таблица 5 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg)) | [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222) |
| read\_accelerometer\_gyro\_data | Чтение данных с акселерометра и гироскопа | [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222) |
| read\_magnetometer\_data | Чтение данных с магнитометра | [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222) |
| bank | Переключение банка | [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222) |
| read\_temperature | Считать данные со встроенного термометра | [pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py#L222) |
| compFilter | Вычисление углов Эйлера | [CompFilter](https://github.com/MarkSherstan/MPU-6050-9250-I2C-CompFilter/blob/master/RPi/main.py) |
| compFilter | Слияние через комплиментарный фильтр | [CompFilter](https://github.com/MarkSherstan/MPU-6050-9250-I2C-CompFilter/blob/master/RPi/main.py) |
| mpu\_plot\_test | Калибровка датчиков | [makerportal](https://github.com/makerportal/mpu92-calibration/blob/main/complete/mpu9250_full_calibration.py) |
| swReset | Перезагрузка | [SparkFun](https://github.com/sparkfun/Qwiic_9DoF_IMU_ICM20948_Py/blob/main/qwiic_icm20948.py#L370) |
| sleep | Режим сна | [SparkFun](https://github.com/sparkfun/Qwiic_9DoF_IMU_ICM20948_Py/blob/main/qwiic_icm20948.py#L370) |

# Часть 2. Алгоритм реализации библиотеки Horizon

1. **Инициализация**Шаги, описанные далее, приводят датчик в рабочее состояние: включают его, устанавливают частоты сэмплирования, ФНЧ, диапазоны работы, включают сглаживание времени опроса, настраивают магнитометр и Master-I2C шину.
2. **Включение датчика**  
   self.bank(0)  
   self.write(ICM20948\_PWR\_MGMT\_1, 0b11000001) – ресет датчика

time.sleep(0.1)

self.write(ICM20948\_PWR\_MGMT\_1, 0b00000001) – автоматический выбор тактовой частоты

self.write(ICM20948\_PWR\_MGMT\_2, 0b00000000) – вкл гироскопа и акселерометра

1. **Сглаживание времени опроса**  
   self.bank(2)  
   self.write(ODR\_ALIGN\_EN, 0x01) – разрешает выравнивание времени запуска ODR при записи в любой из следующих регистров: GYRO\_SMPLRT\_DIV, ACCEL\_SMPLRT\_DIV\_1, ACCEL\_SMPLRT\_DIV\_2, I2C\_MST\_ODR\_CONFIG (сноска из документации).

**Настройка гироскопа: установка диапазона работы в 500dps, включение ФНЧ на 50Гц**self.set\_gyro\_full\_scale(500)

self.set\_gyro\_sample\_rate(50)

self.set\_gyro\_low\_pass(True, 3)  
**Настройка акселерометра: установка диапазона работы на 2g, включение ФНЧ на 50Гц**self.set\_accelerometer\_full\_scale(2)

self.set\_accelerometer\_sample\_rate(50)

self.set\_accelerometer\_low\_pass(True, 3)

1. **Настройка пина прерывания**  
   self.bank(0)  
   self.write(ICM20948\_INT\_PIN\_CFG, 0b00110000)
2. **Настройка I2C Master модуля для работы с магнитометром**Этот шаг на данный момент реализуется методом i2c\_master\_enable(), который описан в [приложении 1](#enable_master_i2c).
3. **Ресет магнитометра**self.bank(3)

self.mag\_write(AK09916\_CNTL3, 0b01)

1. **Установка режима магнитометра**  
   self.mag\_write(AK09916\_CNTL2, 0x06) – установка на продолжительный опрос, частота 50Гц  
   **Примечание**: о настройке частоты сэмплирования магнитометра см. [раздел 2.7](#set_mag_sample_rate)
2. **Интерфейс доступа к считанным данным**

В базовой библиотеке метод для получения значений с датчика, инициирует их считывание и возвращает результат.

Вместо этого полученные данные необходимо сохранять в свойствах класса, что позволит

* отделить корректировку данных от работы с регистрами;
* обрабатывать один и тот же набор данных разными фильтрами или прочими сервисными методами, не вызывая очередное их измерение;

Далее необходимо реализовать возможность установки оффсетов, которые автоматически корректируют значения, записываемые в данные свойства.

1. **Разовое/раздельное считывание показаний акселерометра и гироскопа**

Базовая библиотека реализует групповое считывание данных акселерометра и гироскопа методом read\_accelerometer\_gyro\_data. В образовательных целях следует добавить возможность раздельного считывания данных.

Библиотека от wickiedLuckas позволяет сделать вывод, что методы раздельного считывания реализуются считыванием по 6 байт с двух различных регистров (см. таблицу ниже).

|  |  |
| --- | --- |
|  | **Команда** |
| **Акселерометр + Гироскоп**  **(групповое считывание)** | read\_bytes(ICM20948\_ACCEL\_XOUT\_H, 12) |
| **Акселерометр** | read\_bytes(ICM20948\_ACCEL\_XOUT\_H, 6) |
| **Гироскоп** | read\_bytes(ICM20948\_GYRO\_XOUT\_H, 6) |

Внимание: в коде [библиотеки pimoroni](https://github.com/pimoroni/icm20948-python/blob/master/library/icm20948/__init__.py) переменная ICM20948\_GYRO\_XOUT\_H инициализирована с опечаткой (см. строка 37).

1. **Калибровка**

Функционал калибровки обеспечивается кодовой базой [данной библиотеки](https://github.com/makerportal/mpu92-calibration/blob/main/complete/mpu9250_full_calibration.py) от Makerportal, функции которой многократно считывают данные с датчика, дают подсказки пользователю, вычисляют оффсеты и записывают полученные калибровочные значения в файл.

1. Из библиотеки требуется убрать код, связанный с визуализацией данных;

2. Настроить считывание данных в соответствии с интерфейсом модуля ICM-20948 Horizon;

3. Переписать функцию записи калибровочных значений в файл таким образом, чтобы работать с JSON а не CSV;

4. Аналогично доработать функцию считывания калибровочных данных из файла и внедрить её в инициализацию объекта ICM.

1. **Углы Эйлера, фильтрация**

Функционал вычисления углов Эйлера базируется на методе compFilter() [библиотеки](https://github.com/MarkSherstan/MPU-6050-9250-I2C-CompFilter/blob/master/RPi/main.py#L111) от MarkSherstan, который требует ряда изменений при внедрении:

* 1. Получение аргументов, сохранение/возвращение результатов необходимо адаптировать под интерфейс базовой библиотеки.
  2. Требуется разбить функционал compFilter() на два метода: **первый** вычисляет и возвращает/сохраняет значения углов на основе последних показаний акселерометра, а **второй** – собственно фильтрует значения

После проработки описанных выше шагов можно добавить поддержку фильтра Калмана

1. **Включение/выключение датчиков**

Частично реализуется переносом и адаптацией метода [enable\_sensor(accel, gyro, temp)](https://github.com/WickedLukas/ICM20948/blob/master/ICM20948.cpp#L1644), который предназначен для включения/выключения акселерометра, гироскопа и термометра.

Настройка и запуск магнитометра требует многих действий при инициализации, обратить которые которые может быть достаточно накладно с точки зрения эффекта на работу всей платформы. Вследствие чего, вопрос о включении магнитометра стоит и оставить при инициаолизации всего IMU-датчика.

1. **Настройка частоты сэмплирования магнитометра**

Реализуется переносом метода [magnetometer\_data\_rate(mag\_rate)](https://github.com/adafruit/Adafruit_CircuitPython_ICM20X/blob/main/adafruit_icm20x.py#L758)

Примечание: доступные варианты частоты магнитометра можно посмотреть в таблице 4 [документа](https://disk.yandex.ru/i/FreiswEkEjUQWg).

1. **Перезагрузка датчика**

Перенос метода [swReset()](https://github.com/sparkfun/Qwiic_9DoF_IMU_ICM20948_Py/blob/main/qwiic_icm20948.py#L370)

1. **Переход в режим сна и обратно**

Перенос метода [sleep(on)](https://github.com/sparkfun/Qwiic_9DoF_IMU_ICM20948_Py/blob/main/qwiic_icm20948.py#L393)

**ПРИМЕЧАНИЕ**

Изложенный алгоритм допускает изменение порядка реализации некоторых пунктов, которые напрямую не зависят от других и являются периферийными.

Также внедрение в библиотеку перечисленных выше методов требует учитывать при переносе кода следующие нюансы:

* Необходима сверка регистров (обычно разнятся только префиксом);
* Изменение метода записи/чтения с регистра на аналогичный, реализованный в базовой библиотеке;
* Изменение метода для смены банка на аналогичный, реализованный в базовой библиотеке.

Примеры отличий:

|  |  |  |
| --- | --- | --- |
|  | Изначальный метод | Базовая библиотека |
| Регистры | \_AK09916\_CNTL2 | AK09916\_CNTL2 |
| MagDataRate.SHUTDOWN | AK09916\_CNTL2\_MODE\_OFF |
| self.AGB0\_REG\_PWR\_MGMT\_1 | ICM20948\_PWR\_MGMT\_1 |
| Методы | writeByte(addr, reg, val) | write(reg, val) |
| readByte | read\_bytes(reg, length=1) |
| write\_mag\_register | mag\_write |

# Приложение 1. Настройка I2C Master модуля для работы с магнитометром

1. **Ресет I2C Master модуля**  
   self.bank(0)

temp = self.read(ICM20948\_USER\_CTRL)

self.write(ICM20948\_USER\_CTRL, temp | 0b10)

1. **Включение модуля I2C Master (вкл передачи данных по вспомогательной шине)**  
   self.bank(0)  
   temp = self.read(ICM20948\_USER\_CTRL)

self.write(ICM20948\_USER\_CTRL, temp | 0b00100000) – вызов ресета

time.sleep(0.1)   
**Примечание**: пауза важна, так как нельзя допустить выполнения ресета во время передачи данных по шине

1. **Установка тактовой частоты шины**  
   self.bank(3)

self.write(ICM20948\_I2C\_MST\_CTRL, 0x07) – установка 400 кГц тактовой частоты (см. ст. 81 документации)

time.sleep(0.01)

1. **Настройка частоты сэмплирования магнитометра I2C Master модулем**  
   self.bank(0)

self.write(ICM20948\_LP\_CONFIG, 0b00010000) – частота определяется частотой сэмплирования гироскопа

self.sleep(0.01)

# Приложение 2. Алгоритм чтения с регистров магнитометра

Ниже представлено описание метода mag\_read\_bytes(self, reg, length=1)

1. self.bank(3)
2. self.write(ICM20948\_I2C\_SLV0\_ADDR, AK09916\_I2C\_ADDR | 0x80) – указание адреса магнитометра
3. self.write(ICM20948\_I2C\_SLV0\_REG, reg) – произвести чтение c регистра reg  
   time.sleep(0.005)
4. self.write(ICM20948\_I2C\_SLV0\_CTRL, 0b10000000 | (0b1111 & length)) – разрешение чтения данных со SLAVE-устройства с заданной частотой дискретизации и сохранение данных в регистре EXT\_SENS\_DATA\_00; length – кол-во байт
5. return self.read\_bytes(ICM20948\_EXT\_SLV\_SENS\_DATA\_00, length) – считывание данных с ICM и их возвращение

# Приложение 3. Алгоритм записи в регистры магнитометра

Ниже представлено описание метода mag\_write(self, reg, value)

1. self.bank(3)
2. self.write(ICM20948\_I2C\_SLV0\_ADDR, AK09916\_I2C\_ADDR | 0x80) – указание адреса магнитометра
3. self.write(ICM20948\_I2C\_SLV0\_REG, reg) – указание регистра, в который требуется произвести запись
4. self.write(ICM20948\_I2C\_SLV0\_DO, value) – произвести запись value

# Приложение 4. Алгоритм чтения показаний магнитометра

Ниже представлено описание метода read\_mag\_data(self)

Примечание: предполагатся, что перед попыткой чтения показаний с магнитометра уже были была выполнена [настройка Master-I2C шины](#enable_master_i2c), включение магнитометра и установка одного из CONTINIOUS режимов работы.

1. data = self.mag\_read\_bytes(AK09916\_HXL, 8) – считывание данных и обновление Status 2 (см. ст. 23 документации на AK09916)
2. x, y, z, \_ = struct.unpack("<hhhh", bytearray(data)) – сохранение показаний магнитометра в x,y,z
3. скалирование  
   x \*= 0.15

y \*= 0.15

z \*= 0.15