Обработка датчиков 2

Павел Пронин
C++ разработчик

Проверка связи

Если у вас нет звука:

- убедитесь, что на вашем устройстве и на колонках включён звук
- обновите страницу вебинара (или закройте страницу и заново присоединитесь к вебинару)
- откройте вебинар в другом браузере
- перезагрузите компьютер (ноутбук) и заново попытайтесь зайти

Поставьте в чат:

\+ если меня видно и слышно

— если нет

Павел Пронин

О спикере:

- Разработчик на C++ более 8-ми лет
- Опыт в разработке беспилотных автомобилей
- С 2022 года разработчик в компании разработки мобильных игр Playrix (компания разрабатывает такие игры как homescapes и gardegscapes)

Вспоминаем прошлое занятие

Вопрос: Как изменяется сопротивление NTC термистора при увеличении температуры?

Вспоминаем прошлое занятие

Вопрос: Как изменяется сопротивление NTC термистора при увеличении температуры?

Ответ: Сопротивление уменьшается

Вспоминаем прошлое занятие

Вопрос: Как измеряется расстояние до объекта в ультразвуковом дальномере?

Вспоминаем прошлое занятие

Вопрос: Как измеряется расстояние до объекта в ультразвуковом дальномере?

Ответ: На основании времени распространения звука до объекта и обратно

Вспоминаем прошлое занятие

Вопрос: На что реагирует пироприемник?

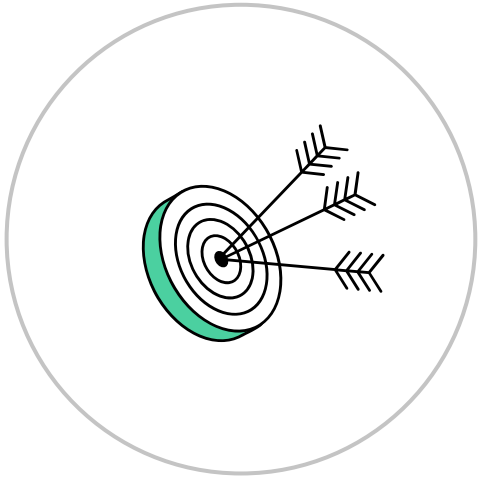
Вспоминаем прошлое занятие

Вопрос: На что реагирует пироприемник?

Ответ: На перемещение объекта, имеющего отличную температуру по сравнению с фоном

Цели занятия

- Узнаем, что такое технология МЭМС
- Познакомимся с МЭМС датчиками движения и научимся их подключать
- Познакомимся с интерфейсом I2C и научимся использовать его для работы с датчиками
- Научимся использовать специальную библиотеку для считывания данных с модуля MPU6050



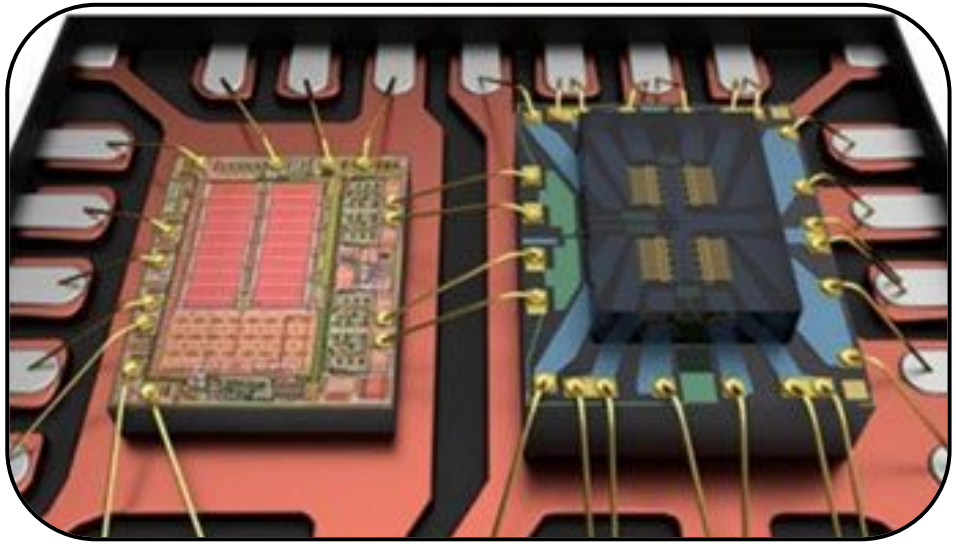
План занятия

1. Что такое МЭМС
2. Как подключить МЭМС датчики движения
3. Как использовать интерфейс I2C
4. Как использовать специальную библиотеку для MPU6050
5. Итоги

Что такое МЭМС

Внутренняя структура МЭМС

Микроэлектромеханическая система, или МЭМС, представляет собой миниатюрное устройство, которая изготовлена как из механических, так и из электрических компонентов, используя методы микрообработки.



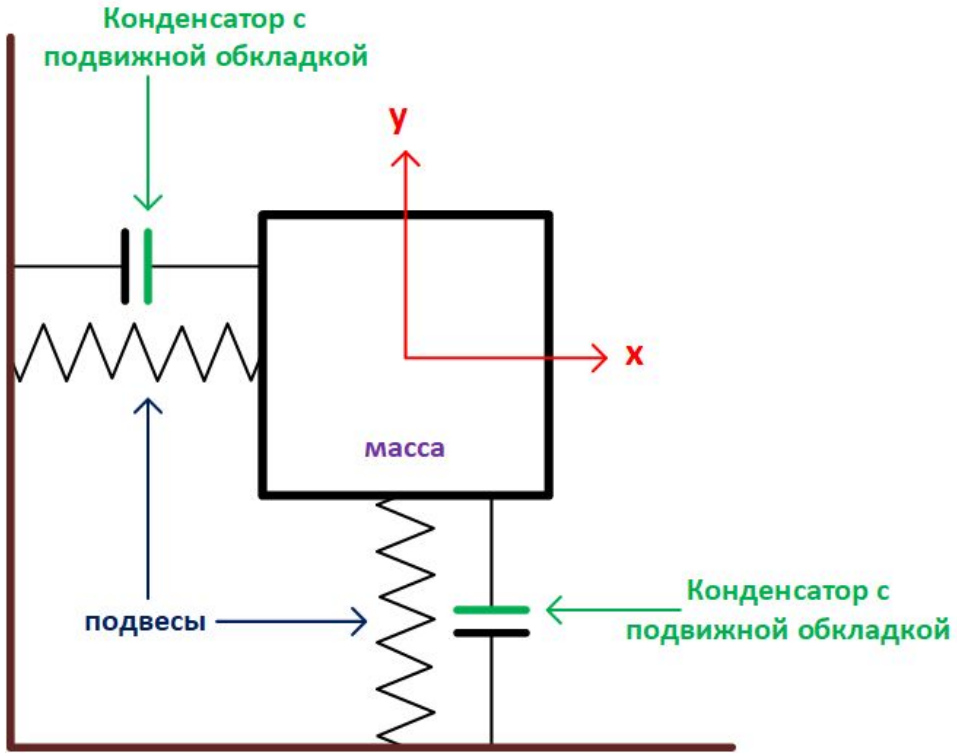
Источник

Где используется технология МЭМС

- аудиокомпоненты (микрофоны)
- оптические переключатели в лазерных системах
- магнетометры
- акселерометры
- гироскопы (датчики угловой скорости) и т.д.

МЭМС акселерометр

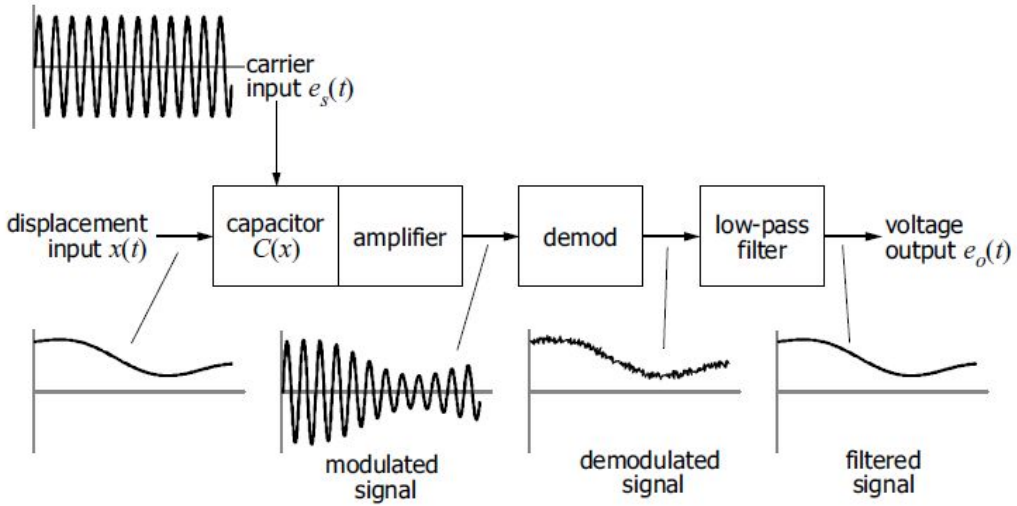
Принцип работы МЭМС акселерометров основан на изменении емкости конденсаторов при изменении ускорения



Источник

Функциональная схема МЭМС акселерометра

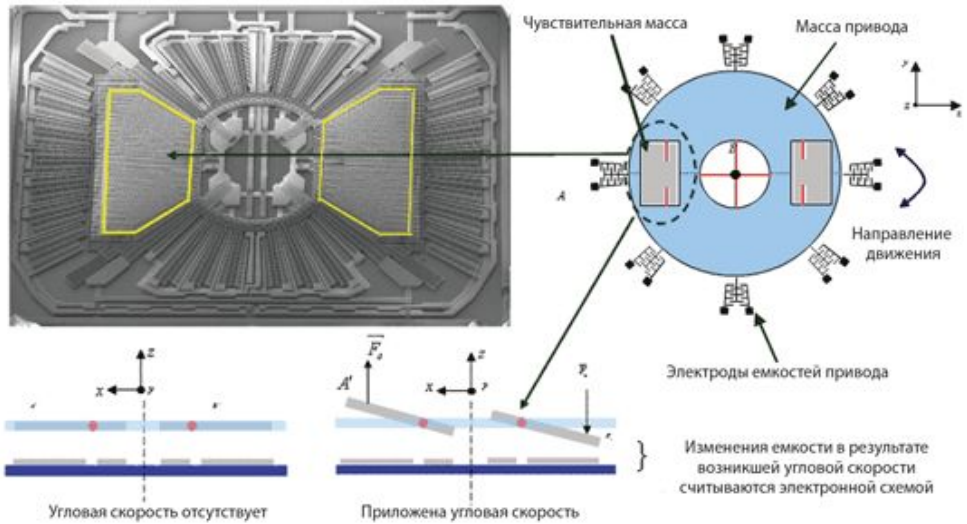
Сигнал x(t) отражает изменение ускорения. Далее он перемножается его с несущим сигналом $e_s(t)$ и усиливается. Затем происходит демодуляция. После чего остатки шума фильтруются с помощью фильтра низких частот.



Источник

МЭМС гироскоп

С появлением угловой скорости сила Кориолиса прикладывается в противоположных направлениях к чувствительной массе. Измеряемая дифференциальная емкостная составляющая пропорциональна углу перемещения



Источник

Область применения МЭМС датчиков движения

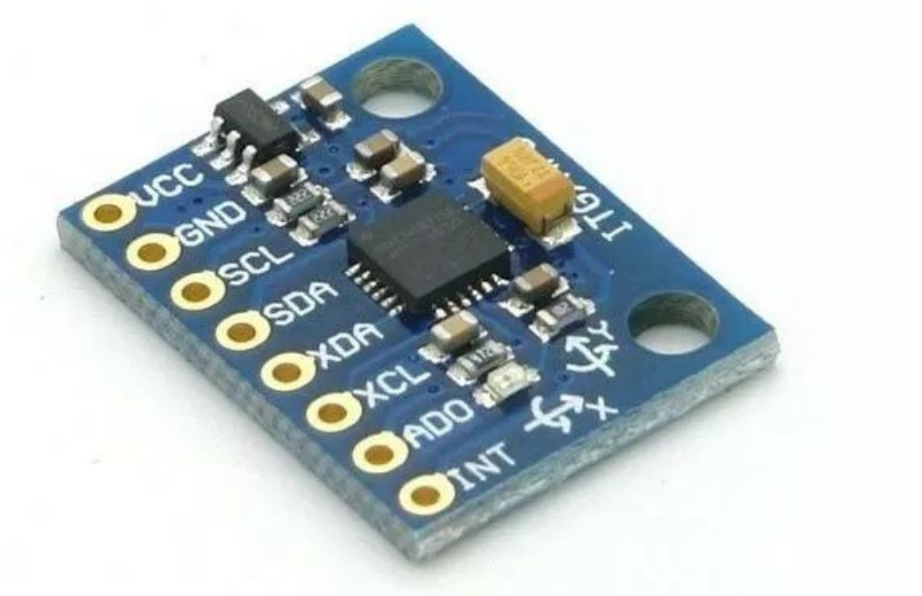
- автомобильные системы стабилизации и курсовой устойчивости;
- дистанционно управляемые модели вертолетов;
- измерительное оборудование;
- ручной и стационарный электроинструмент;
- системы распознавания жестов;
- персональные системы навигации и ориентирования в сочетании с глобальными навигационными спутниковыми системами (ГНСС);
- фитнес браслеты и т.д.

Как подключить МЭМС датчики движения

2

Модуль MPU6050

MPU6050 представляет собой 3-х осевой гироскоп и 3-х осевой акселерометр в одном корпусе

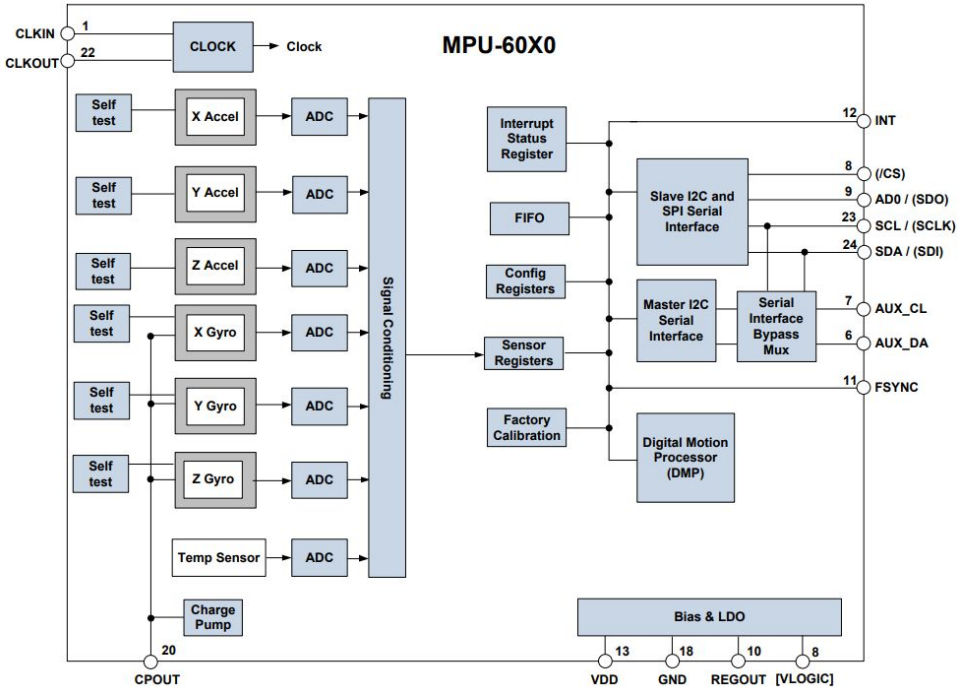


Источник

Структура модуля MPU6050

Характеристики модуля:

- Питание: 3,5 – 6 В;
- Ток потребления: 500 мкА;
- Акселерометр диапазон измерений: ± 2, ± 4, ± 8, ± 16g,
- Гироскоп диапазон измерений: ± 250, ± 500, ± 1000, ± 2000 ° / s,
- Интерфейс: I2C



Источник

Регистры модуля MPU6050

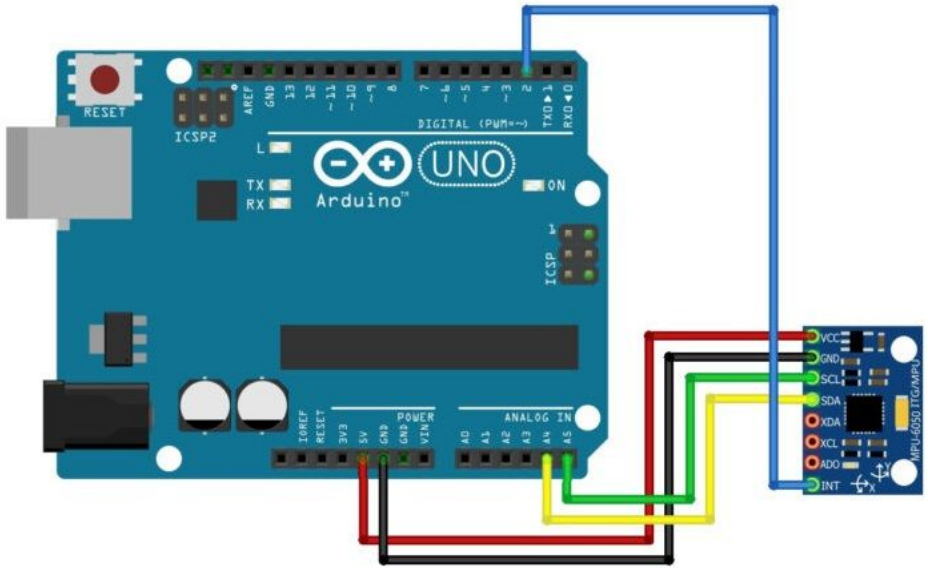
Для управления работой модуля и считывания с него результатов имеется более 50 регистров (на рисунке показана только часть)

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
23	35	FIFO_EN	R/W	TEMP _FIFO_EN	XG _FIFO_EN	YG _FIFO_EN	ZG _FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0 _RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0 _EN	I2C_SLV0 _BYTE_SW	I2C_SLV0 _REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LEN[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1 _RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1 _EN	I2C_SLV1 _BYTE_SW	I2C_SLV1 _REG_DIS	I2C_SLV1 _GRP	I2C_SLV1_LEN[3:0]			
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2 _RW	I2C_SLV2_ADDR[6:0]						
2C	44	I2C_SLV2_REG	R/W	I2C_SLV2_REG[7:0]							
2D	45	I2C_SLV2_CTRL	R/W	I2C_SLV2 _EN	I2C_SLV2 _BYTE_SW	I2C_SLV2 _REG_DIS	I2C_SLV2 _GRP	I2C_SLV2_LEN[3:0]			
2E	46	I2C_SLV3_ADDR	R/W	I2C_SLV3 _RW	I2C_SLV3_ADDR[6:0]						
2F	47	I2C_SLV3_REG	R/W	I2C_SLV3_REG[7:0]							
30	48	I2C_SLV3_CTRL	R/W	I2C_SLV3 _EN	I2C_SLV3 _BYTE_SW	I2C_SLV3 _REG_DIS	I2C_SLV3 _GRP	I2C_SLV3_LEN[3:0]			
31	49	I2C_SLV4_ADDR	R/W	I2C_SLV4 _RW	I2C_SLV4_ADDR[6:0]						
32	50	I2C_SLV4_REG	R/W	I2C_SLV4_REG[7:0]							
33	51	I2C_SLV4_DO	R/W	I2C_SLV4_DO[7:0]							
34	52	I2C_SLV4_CTRL	R/W	I2C_SLV4 _EN	I2C_SLV4 _INT_EN	I2C_SLV4 _REG_DIS	I2C_MST_DLY[4:0]				
35	53	I2C_SLV4_DI	R	I2C_SLV4_DI[7:0]							
36	54	I2C_MST_STATUS	R	PASS_ THROUGH	I2C_SLV4 _DONE	I2C_LOST _ARB	I2C_SLV4 _NACK	I2C_SLV3 _NACK	I2C_SLV2 _NACK	I2C_SLV1 _NACK	I2C_SLV0 _NACK
37	55	INT_PIN_CFG	R/W	INT_LEVEL	INT_OPEN	LATCH _INT_EN	INT_RD _CLEAR	FSYNC_ INT_LEVEL	FSYNC _INT_EN	I2C _BYPASS _EN	-
38	56	INT_ENABLE	R/W	-	-	-	FIFO _OFLOW _EN	I2C_MST _INT_EN	-	-	DATA _RDY_EN
3A	58	INT_STATUS	R	-	-	-	FIFO _OFLOW _INT	I2C_MST _INT	-	-	DATA _RDY_INT

Источник

Подключение модуля MPU6050 к плате Arduino UNO

Для подключения используется интерфейс I2C, который заведен на выводы A4 и A5.
Дополнительно можно использовать выход INT, который подключается к любому выводу платы Arduino



Источник

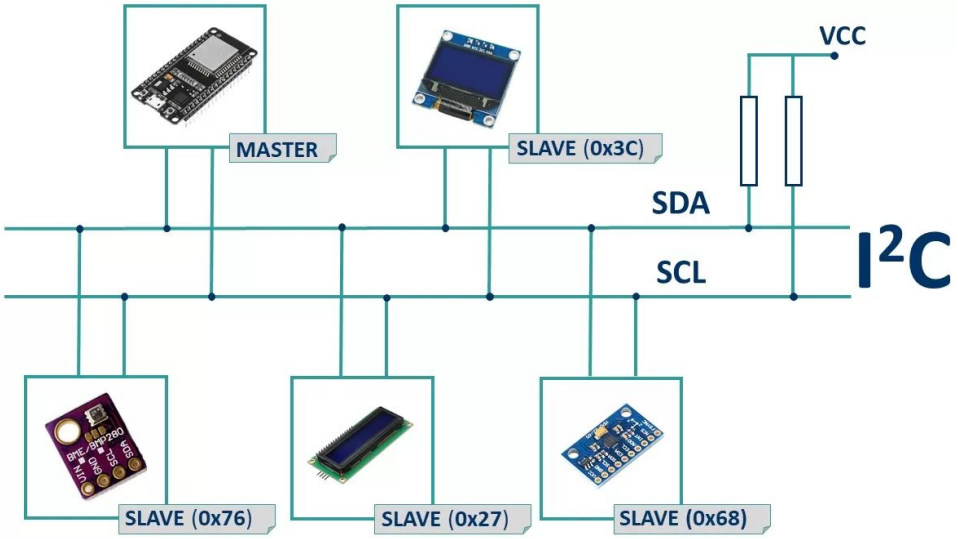
Как использовать интерфейс I2C

3

Параметры интерфейса I2C

Разработан фирмой Philips Semiconductors в начале 1980-х для внутренней связи между микросхемами.

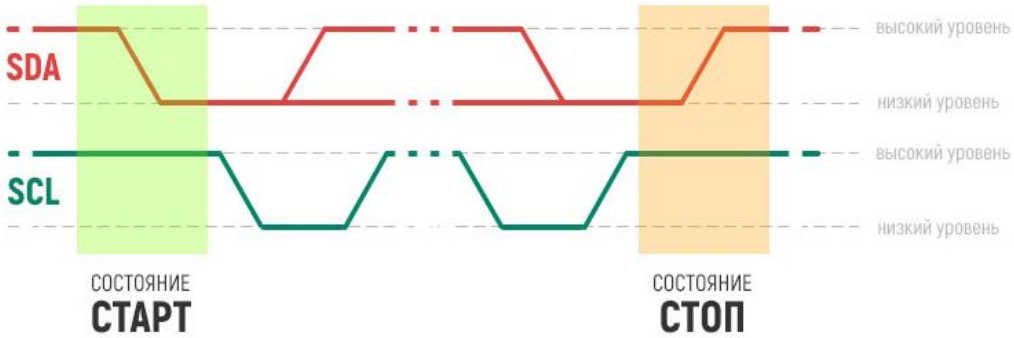
Изначально поддерживал скорость до 100 кбит/с, сейчас содержит режим со скоростью до 3,4 Мбит/с



Источник

Принцип работы интерфейса I2C

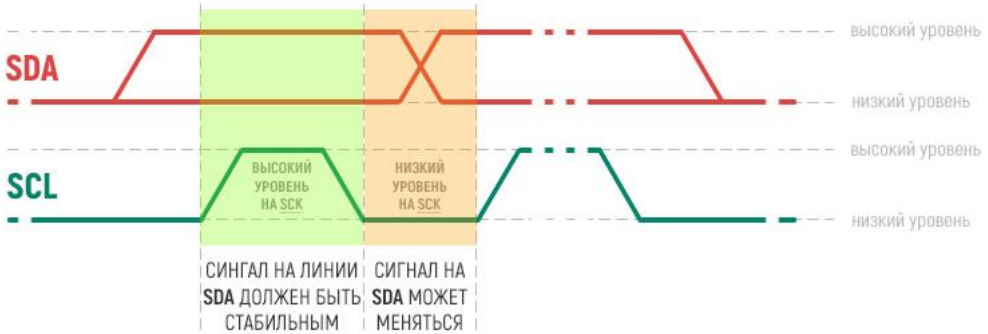
Обмен данными инициируется ведущим устройством.
В исходном состоянии обе линии SDA и SCL находятся в высоком состоянии.
Условие СТАРТ: переход состояния линии SDA с высокого на низкое при высоком состоянии на линии SCL.
Условие СТОП: переход состояния линии SDA с низкого на высокое при высоком состоянии на линии SCL.



Источник

Обмен данными по интерфейсу I2C

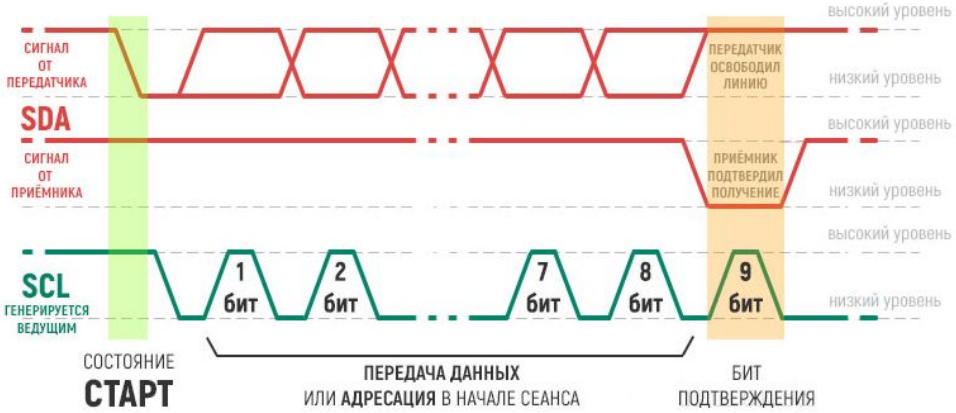
Данные передаются изменением уровня сигнала на линии SDA в определенной последовательности. Обмен данными происходит байтами, каждый из которых состоит из 8-ми бит. Изменять уровень на линии SDA можно только при низком состоянии на тактирующей линии SCL. Бит информации «читается» приёмником только когда на линии SCL установлен высокий уровень сигнала.



Источник

Подтверждение приема данных

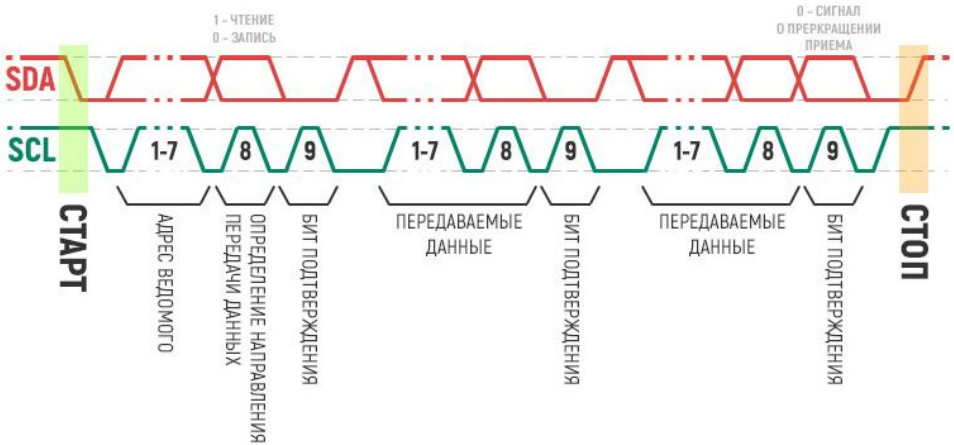
Для корректной передачи данных, приемник должен подтверждать прием каждого байта от передатчика. Для этого вводится специальный бит подтверждения, выставляемый приёмником на шину SDA после приема каждого 8-го бита данных. Передача каждых 8-ми бит данных считается успешной, если приемник отправляет бит подтверждения с низкий уровень сигнала по линии SDA.



Источник

Адресация в интерфейсе I2C

Ведущее устройство, сразу после установки состояния СТАРТ, отправляет по шине специальный байт информации, первые 7 бит которого содержат адрес ведомого (в обычном режиме), а 8-й бит (бит направления) — информацию о направлении передачи: низкий уровень («0») означает, что ведущий будет отправлять информацию ведомому
высокий («1») означает, что ведущий будет получать информацию от ведомого.



Источник

Типичный сеанс обмена по интерфейсу I2C

Из спецификации шины следует, что допускаются не только простые форматы обмена, но и комбинированные, когда в промежутке от состояния СТАРТ до состояния СТОП ведущий и ведомый могут выступать и как приемник и как передатчик данных.

Источник

Библиотека Wire для работы с интерфейсом I2C

Является наследником класса Stream. При подключении библиотеки автоматически вызывается конструктор, который создает объект с именем Wire. Поэтому в пользовательском коде вызывать конструктор не нужно.

void begin()
void begin(uint8_t address) — инициализирует объект Wire и подключается к шине I2C как ведущий или ведомый

Параметры:

- address: 7-битный адрес ведомого устройства; если не задан, плата подключается к шине как мастер

Возвращаемое значение: нет

Библиотека Wire для работы с интерфейсом I2C

uint8_t requestFrom(uint8_t address, uint8_t quantity, uint8_t sendStop) — отправляет запрос на определенное количество байтов от ведущего устройства к ведомому
Параметры:

- address: 7-ми битный адрес устройства, к которому посылается запрос
- quantity: количество запрашиваемых байт
- sendStop: (необязательный параметр) ненулевое значение — после запроса отправляет STOP, освобождая шину I2C; ноль — после запроса отправляет RESTART, шина не освобождается и можно отправлять дополнительные запросы. По умолчанию — ненулевое значение.

Возвращаемое значение: количество байт, возвращенных от устройства

void beginTransmission(uint8_t address) — открывает канал связи по шине I2C с ведомым устройством
Параметры:

- address: 7-ми битный адрес устройства к которому посылается запрос

Возвращаемое значение: нет

Библиотека Wire для работы с интерфейсом I2C

uint8_t endTransmission(uint8_t sendStop) — отправляет данные, которые были поставлены в очередь методом write(), и завершает передачу
Параметры:

- sendStop: (необязательный параметр) ненулевое значение — после запроса отправляет STOP, освобождая шину I2C; ноль — после запроса отправляет RESTART, шина не освобождается и можно отправлять дополнительные запросы. По умолчанию — ненулевое значение.

Возвращаемое значение: состояние передачи:

- 0: успешная передача;
- 1: Объем данных для передачи слишком велик;
- 2: принят NACK при передаче адреса;
- 3: принят NACK при передаче данных;
- 4: другие ошибки

Библиотека Wire для работы с интерфейсом I2C

size_t write(const uint8_t *data, size_t quantity) — ставит данные в очередь для передачи
Параметры:

- data: один байт данных или указатель на массив передаваемых данных
- quantity: (необязательный параметр) длина передаваемого массива данных; используется только при передаче массива в первом параметре.

Возвращаемое значение: количество записанных байт

int available(void) — возвращает количество байт, доступных для чтения
Параметры: нет
Возвращаемое значение: количество байт доступных для считывания

Библиотека Wire для работы с интерфейсом I2C

int read(void) — считывает байт переданной информации
Параметры: нет
Возвращаемое значение: принятый байт

void setClock(uint32_t clock) — устанавливает тактовую частоту обмена данными по I2C интерфейсу
Параметры:

- clock: новое значение частоты обмена данными в герцах. Доступные значения: 10000 — медленный режим, 100000 — стандартное значение, 400000 — быстрый режим, 1000000 — быстрый режим плюс, 3400000 — высокоскоростной режим. Необходимо убедится, что выбранный режим поддерживается платой.

Возвращаемое значение: нет

Считывание данных с модуля MPU6050 с помощью библиотеки WIRE

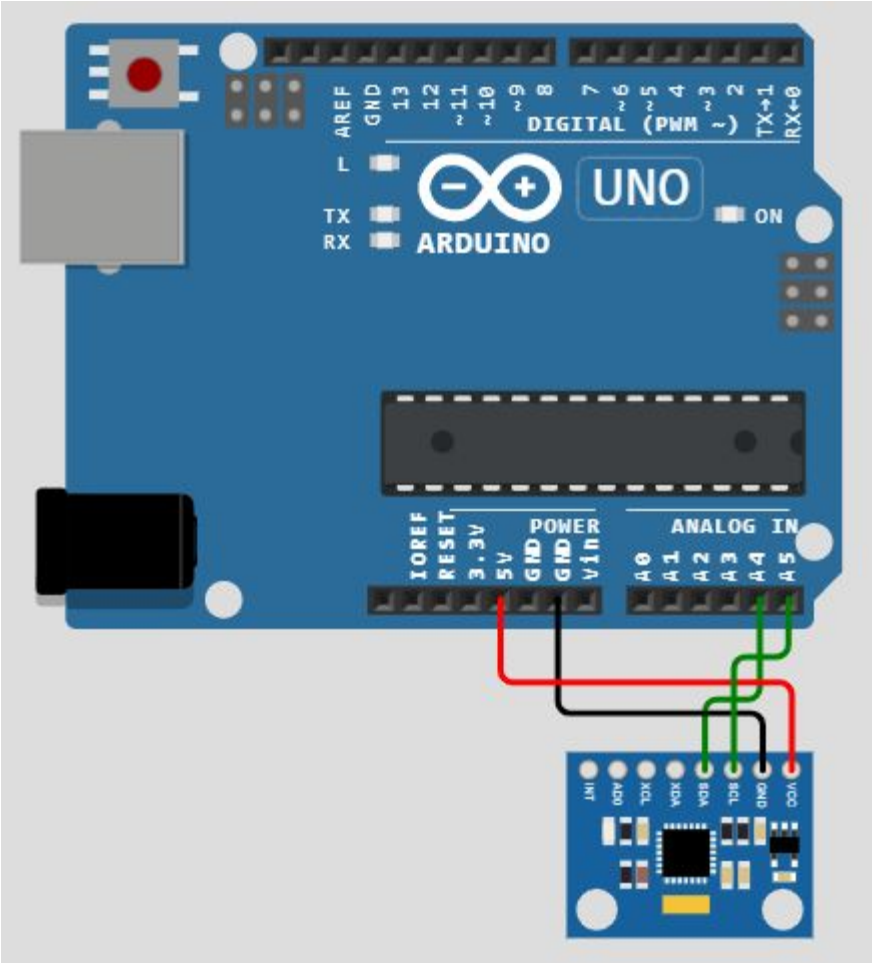
```
#include "Wire.h" //подключение библиотеки для работы с I2C
const int MPU_addr = 0x68; // адрес датчика
// массив данных
// [accX, accY, accZ, temp, gyrX, gyrY, gyrZ]
// acc - ускорение, gyr - угловая скорость, temp - температура (raw)
int16_t data[7];

void setup()
{
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // адрес регистра PWR_MGMT_1
  Wire.write(0);     // 0 - разрешение работы
  Wire.endTransmission(true);
  Serial.begin(9600); // для отладки
}
```

Считывание данных с модуля MPU6050 с помощью библиотеки WIRE

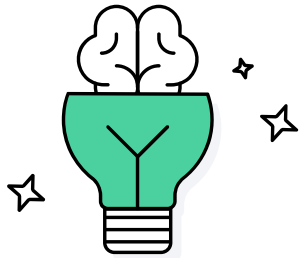
```
void loop()
{
  getData();  // получаем данные
  for (byte i = 0; i < 7; i++) // выводим данные в отладку
  {
    Serial.print(data[i]);
    Serial.print('\t');
  }
  Serial.println();
  delay(1000);
}
// функция чтения данных по I2C
void getData()
{
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // установка начального регистра ACCEL_XOUT_H
  Wire.endTransmission(false); //передача данных без установки состояния STOP
  Wire.requestFrom(MPU_addr, 14, true); // запрос 14 байтов
  for (byte i = 0; i < 7; i++) // считывание данных
  {
    data[i] = Wire.read() << 8 | Wire.read();
  }
}
```

Считывание данных с модуля MPU6050 с помощью библиотеки WIRE

Имитация изменения показаний модуля MPU6050 происходит в процессе симуляции с помощью ползунков



Практическое задание №1



Практика: считывание данных с модуля MPU6050 с помощью библиотеки WIRE

Задание:

1) соберите схему в симуляторе WOKWI, подключив выводы SDA и SCL модуля MPU6050 к выводам A4 и A5 соответственно;
2) создайте скетч с текстом, приведенным выше;
3) проведите моделирование работы

Как выполнять: напишите в чат об удачной работе схемы

Время выполнения: 5 минут

Как использовать специальную библиотеку для MPU6050

4

Библиотека MPU6050 для работы с модулем MPU6050

Библиотека содержит более сотни функций для настройки и работы с модулем. Здесь будут приведены только основные из них

MPU6050(uint8_t address, void *wireObj):devAddr(address), wireObj(wireObj) — конструктор, создающий объект MPU6050

Параметры:

- address: (необязательный параметр) 7-битный адрес ведомого устройства; если не задан, то берется значение по умолчанию MPU6050_DEFAULT_ADDRESS (0x68)
- wireObj: (необязательный параметр) указатель на объект, реализующий интерфейс I2C; если не задан, то берется значение по умолчанию из библиотеки I2Cdev

Возвращаемое значение: нет

Библиотека MPU6050 для работы с модулем MPU6050

void initialize() — инициализирует объект MPU6050

Параметры: нет
Возвращаемое значение: нет

bool testConnection() — тестирует связь с модулем, считывая регистр ID и сравнивая со значением 0x34
Параметры: нет
Возвращаемое значение: true - связь есть, false - связи нет

Библиотека MPU6050 для работы с модулем MPU6050

void getRotation(int16_t* x, int16_t* y, int16_t* z) — считывает значение датчика угловой скорости по трем осям
Параметры:

- x, y, z: указатели на буферы, куда запишется результат считывания

Возвращаемое значение: нет

int16_t getRotationX()
int16_t getRotationY()
int16_t getRotationZ() — считывает значение датчика угловой скорости по одной оси
Параметры: нет
Возвращаемое значение: значение угловой скорости

Библиотека MPU6050 для работы с модулем MPU6050

uint8_t getFullScaleGyroRange() — получить диапазон измерения датчика угловой скорости
Параметры: нет
Возвращаемое значение: 0 = +/- 250 degrees/sec; 1 = +/- 500 degrees/sec; 2 = +/- 1000 degrees/sec; 3 = +/- 2000 degrees/sec

void setFullScaleGyroRange(uint8_t range) — установить диапазон измерения датчика угловой скорости
Параметры:

- range: значение диапазона: 0 = +/- 250 degrees/sec; 1 = +/- 500 degrees/sec; 2 = +/- 1000 degrees/sec; 3 = +/- 2000 degrees/sec

Возвращаемое значение: нет

Библиотека MPU6050 для работы с модулем MPU6050

void getAcceleration(int16_t* x, int16_t* y, int16_t* z) — считывает значение акселерометра по трем осям
Параметры:

- x, y, z: указатели на буферы, куда запишется результат считывания

Возвращаемое значение: нет

int16_t getAccelerationX()
int16_t getAccelerationY()
int16_t getAccelerationZ() — считывает значение акселерометра по одной оси
Параметры: нет
Возвращаемое значение: значение ускорения

Библиотека MPU6050 для работы с модулем MPU6050

uint8_t getFullScaleAccelRange() — получить диапазон измерения акселерометра
Параметры: нет
Возвращаемое значение: 0 = +/- 2 g; 1 = +/- 4 g; 2 = +/- 8 g; 3 = +/- 16 g

void setFullScaleAccelRange(uint8_t range) — установить диапазон измерения гироскопа
Параметры:

- range: 0 = +/- 2 g; 1 = +/- 4 g; 2 = +/- 8 g; 3 = +/- 16 g

Возвращаемое значение: нет

Библиотека MPU6050 для работы с модулем MPU6050

void getMotion6(int16_t* ax, int16_t* ay, int16_t* az, int16_t* gx, int16_t* gy, int16_t* gz) — получить значение по всем 6-и осям
Параметры:

- ax, ay, az, gx, gy, gz: указатели на буферы, куда запишется результат считывания

Возвращаемое значение: нет

int16_t getTemperature() — получить значение температуры
Параметры: нет
Возвращаемое значение: значение температуры

Расшифровка “сырых” данных

По каждой оси и параметру датчик выдаёт 16-битное знаковое значение (-32768.. 32767). При стандартных настройках это значение отражает:

- ускорение в диапазоне -2.. 2 g (9.82 м/с/с).
- угловую скорость в диапазоне -250.. 250 градусов/секунду.

Таким образом, для перевода сырых данных в физические величины (если это нужно) можно сделать по каждой оси:

- ускорение в м/с/с при чувствительности 2: float accX_f = accX / 32768 * 2
- угловая скорость в град/с при чувствительности 250: float gyrX_f = gyrX / 32768 * 250

Считывание данных с модуля MPU6050 с помощью библиотеки MPU6050

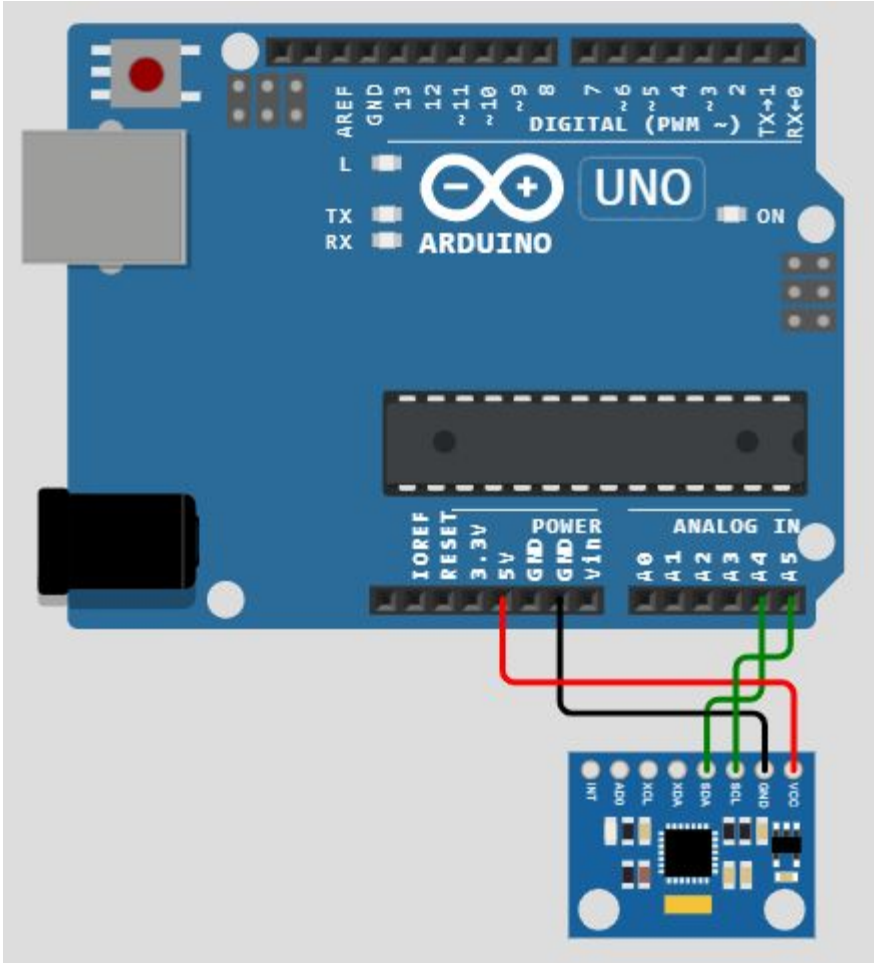
```
#include "MPU6050.h" // подключение библиотеки
MPU6050 mpu; //датчик
int16_t ax, ay, az; // результаты измерения
int16_t gx, gy, gz;

void setup()
{
  Wire.begin();
  Serial.begin(9600);
  mpu.initialize();
  Serial.println(mpu.testConnection() ? "MPU6050 OK" : "MPU6050 FAIL"); // состояние соединения
  delay(1000);
}
```

Считывание данных с модуля MPU6050 с помощью библиотеки MPU6050

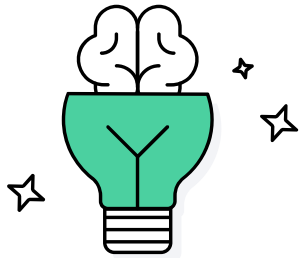
```
void loop()
{
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); //четние всех 6-и осей
  Serial.print((float)ax/32768*2); Serial.print('\t'); //вывод в физических величинах
  Serial.print((float)ay/32768*2); Serial.print('\t');
  Serial.print((float)az/32768*2); Serial.print('\t');
  Serial.print((float)gx/32768*250); Serial.print('\t');
  Serial.print((float)gy/32768*250); Serial.print('\t');
  Serial.println((float)gz/32768*250);
  delay(1000);
}
```

Считывание данных с модуля MPU6050 с помощью библиотеки MPU6050

Имитация изменения показаний модуля MPU6050 происходит в процессе симуляции с помощью ползунков



Практическое задание №2



Практика: считывание данных с модуля MPU6050 с помощью библиотеки MPU6050

Задание:

1) соберите схему в симуляторе WOKWI, подключив выводы SDA и SCL модуля MPU6050 к выводам A4 и A5 соответственно;
2) создайте скетч с текстом, приведенным выше;
3) проведите моделирование работы

Как выполнять: напишите в чат об удачной работе схемы

Время выполнения: 5 минут

Итоги

5

Итоги занятия

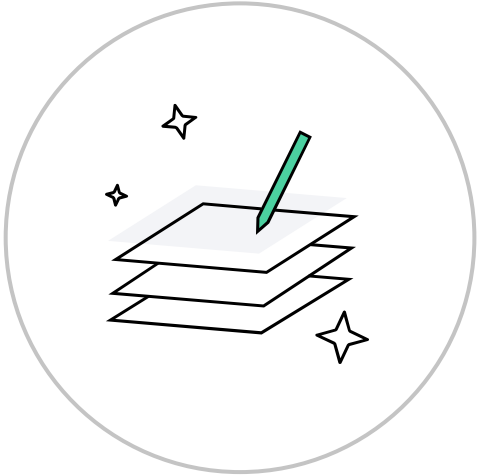
Сегодня мы

1. Узнали особенности технологии МЭМС
2. Научились подключать МЭМС датчики движения
3. Узнали особенности использования интерфейса I2C при работе с датчиками
4. Научились применять специальную библиотеку для работы с датчиком MPU6050

Домашнее задание

Давайте посмотрим ваше домашнее задание.

1. Вопросы по домашней работе задавайте в чате группы
2. Задачи можно сдавать по частям
3. Зачёт по домашней работе ставят после того, как приняты все задачи



Задавайте вопросы и пишите отзыв о лекции

Павел Пронин
C++ разработчик