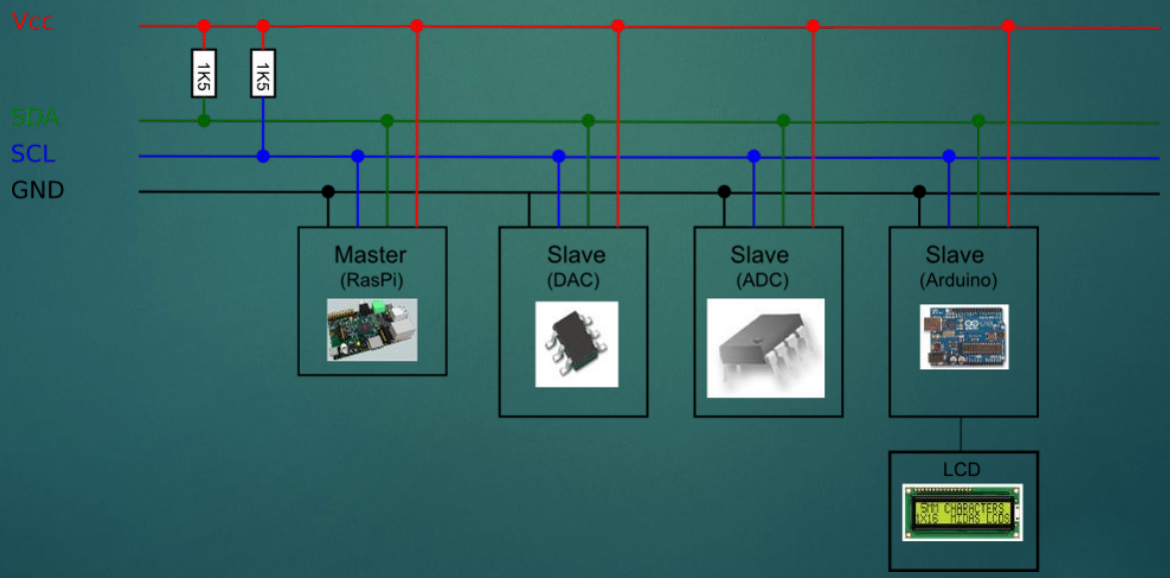
Интерфейс I2C и датчики на нем

ИГОРЬ ХУДЯКОВ
СМИРНОВ ИВАН

I2C

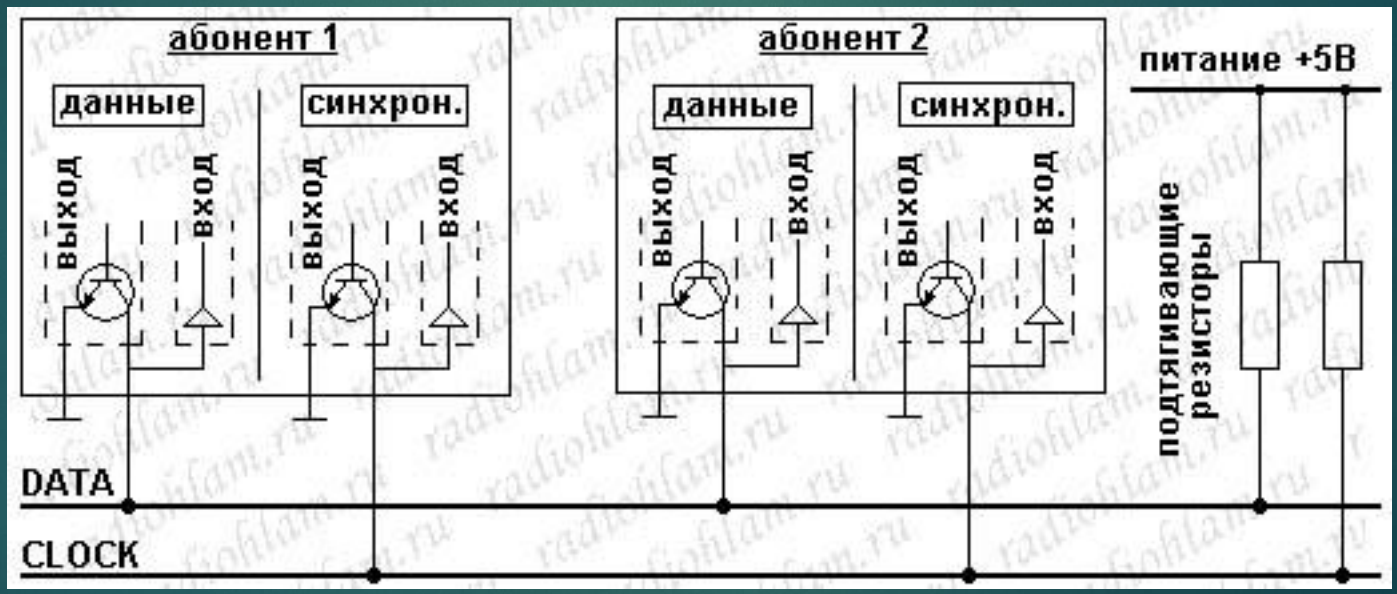
I2C - это достаточно широко распространённый сетевой последовательный интерфейс, придуманный фирмой Philips и завоевавший популярность относительно высокой скоростью передачи данных (обычно до 100 кбит/с, в современных микросхемах до 400 кбит/с), дешевизной и простотой реализации.

Одновременно можно подключить 127 устройств.

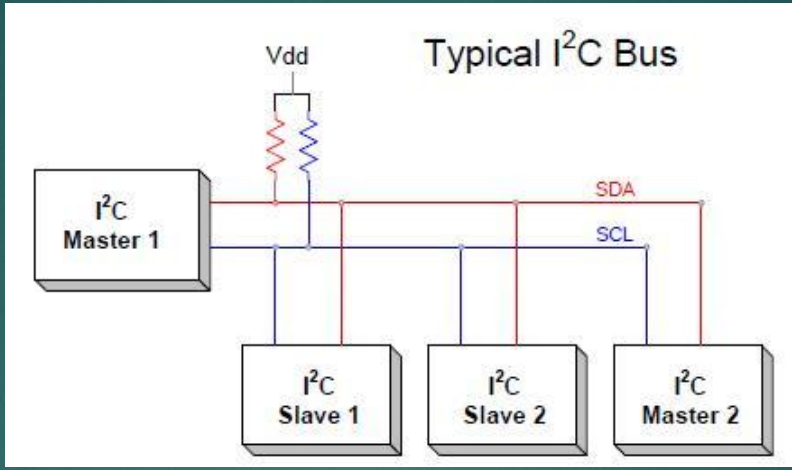


I2C - ФИЗИКА

Устройства имеют выход с "открытым коллектором" - когда выходной ранзистор закрыт на линии через подтягивающий резистор устанавливается высокий уровень, а когда открыт - низкий. Номинал резисторов влияет на скорость передачи данных(чем выше номинал, тем ниже скорость)



I2C - ЛОГИКА



Любое устройство на шине I2C может быть одного из двух типов:

- Master(ведущий) – управляет сеансом
- Slave(ведомый) – «поддакивает» при приеме байта

Master и Slave могут работать в двух режимах:

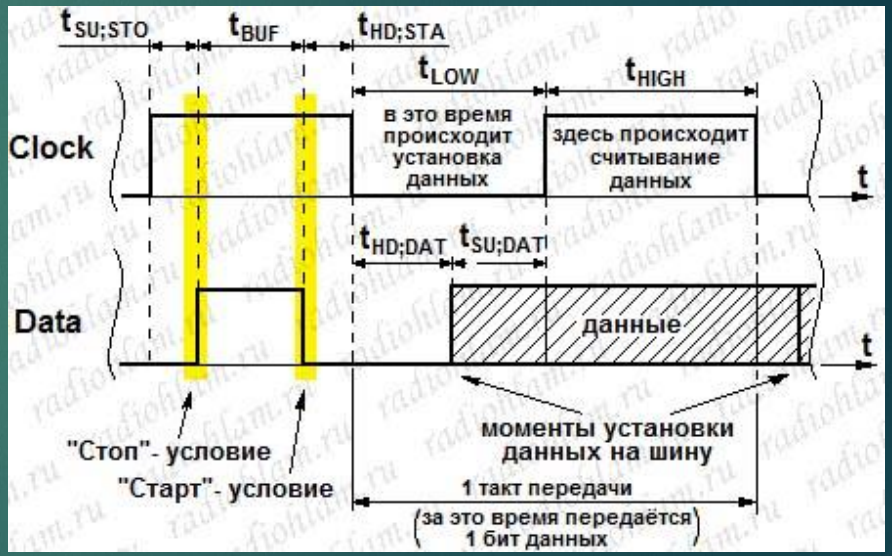
- Приемник
- Передатчик

I2C - ЛОГИКА

Каждый сеанс начинается и заканчивается с подачи «Мастером» так называемых **START** и **STOP** условий.

«Start-условие» — это изменение уровня на линии Data с высокого на низкий при наличии высокого уровня на линии Clock.

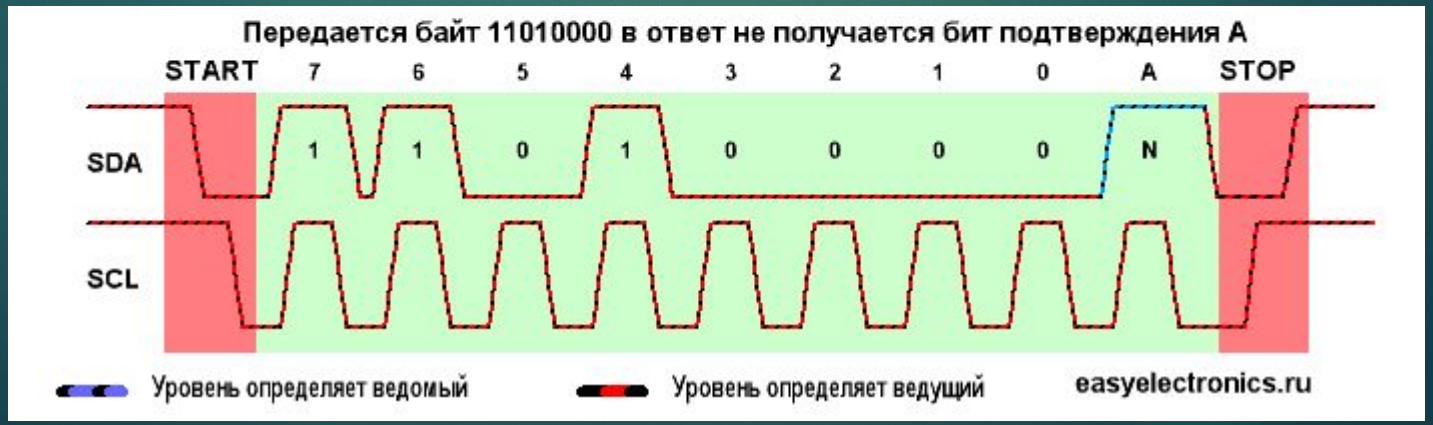
«Stop-условие» - это изменение уровня на линии Data с низкого на высокий при наличии высокого уровня на линии Clock.



Внутри сеанса любые изменения на линии Data при наличии высокого уровня на линии Clock запрещены!!!

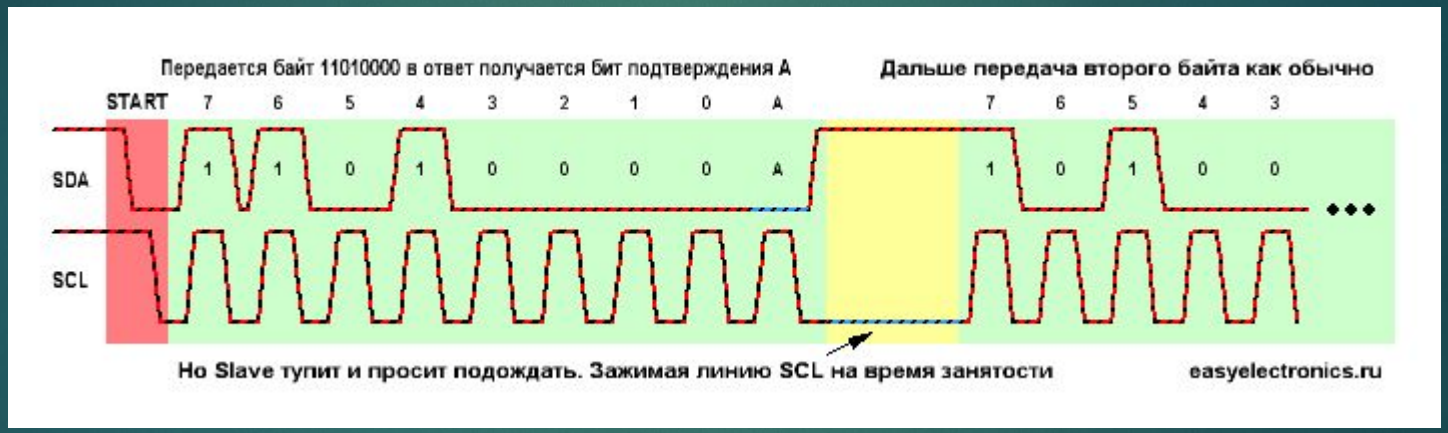
I2C - ЛОГИКА

Внутри сеанса передача состоит из пакетов по девять бит, передаваемых в обычной положительной логике (то есть высокий уровень — это 1, а низкий уровень — это 0). Из них 8 бит передаёт "Передатчик" "Приёмнику", а последний девятый бит передаёт "Приёмник" "Передатчику". Биты в пакете передаются старшим битом вперёд. Последний, девятый бит называется битом подтверждения ACK (от английского слова acknowledge — подтверждение). Он передаётся в инвертированном виде, то есть 0 на линии соответствует наличию бита подтверждения, а 1 — его отсутствию.



I2C - ЛОГИКА

Если **Slave** торомоз и не успевает, то он может **насильно положить линию SCL в землю и не давать ведущему генерировать новые такты**.



I2C - ЛОГИКА

Как передаются отдельные биты понятно, теперь о том что эти биты значат. В отличии от **SPI** тут умная адресная структура. Данные шлются пакетами, каждый пакет состоит из девяти бит. 8 данных и 1 бит подтверждения/не подтверждения приема.

Первый пакет шлется от ведущего к ведомому это **физический адрес устройства и бит направления**.

I2C - ЛОГИКА

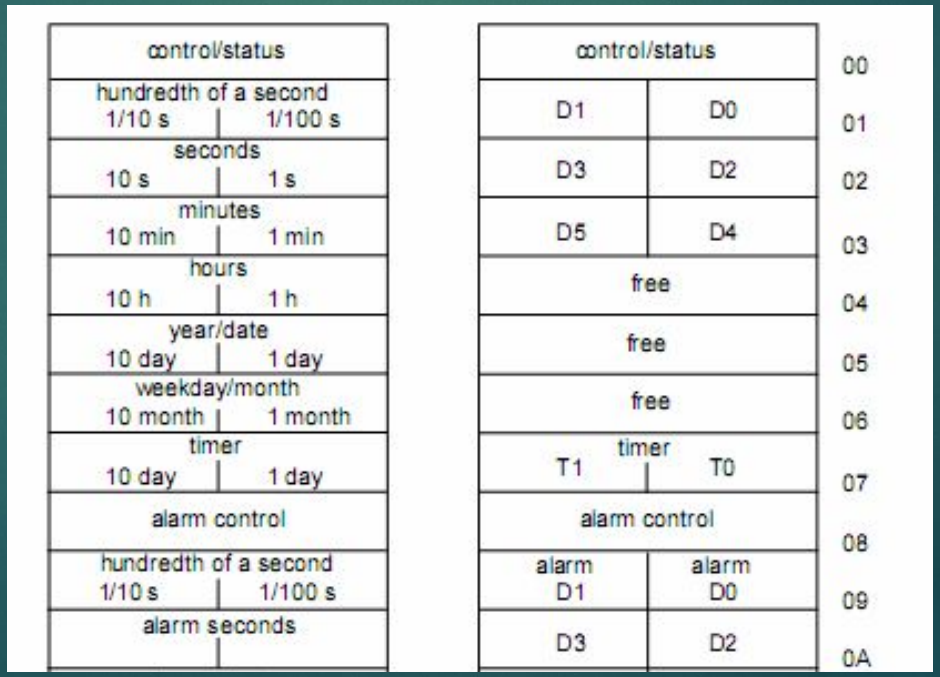
После адресного пакета идут **пакеты с данными** в ту или другую сторону, в зависимости от **бита RW** в заголовочном пакете.

I2C - ЛОГИКА

Чтение практически также, но тут есть одна тонкость. При приеме последнего байта надо дать ведомому понять, что в его услугах больше не нуждаемся и **отослать NACK** на последнем байте. Если отослать ACK то после стопа Master не отпустит линию.

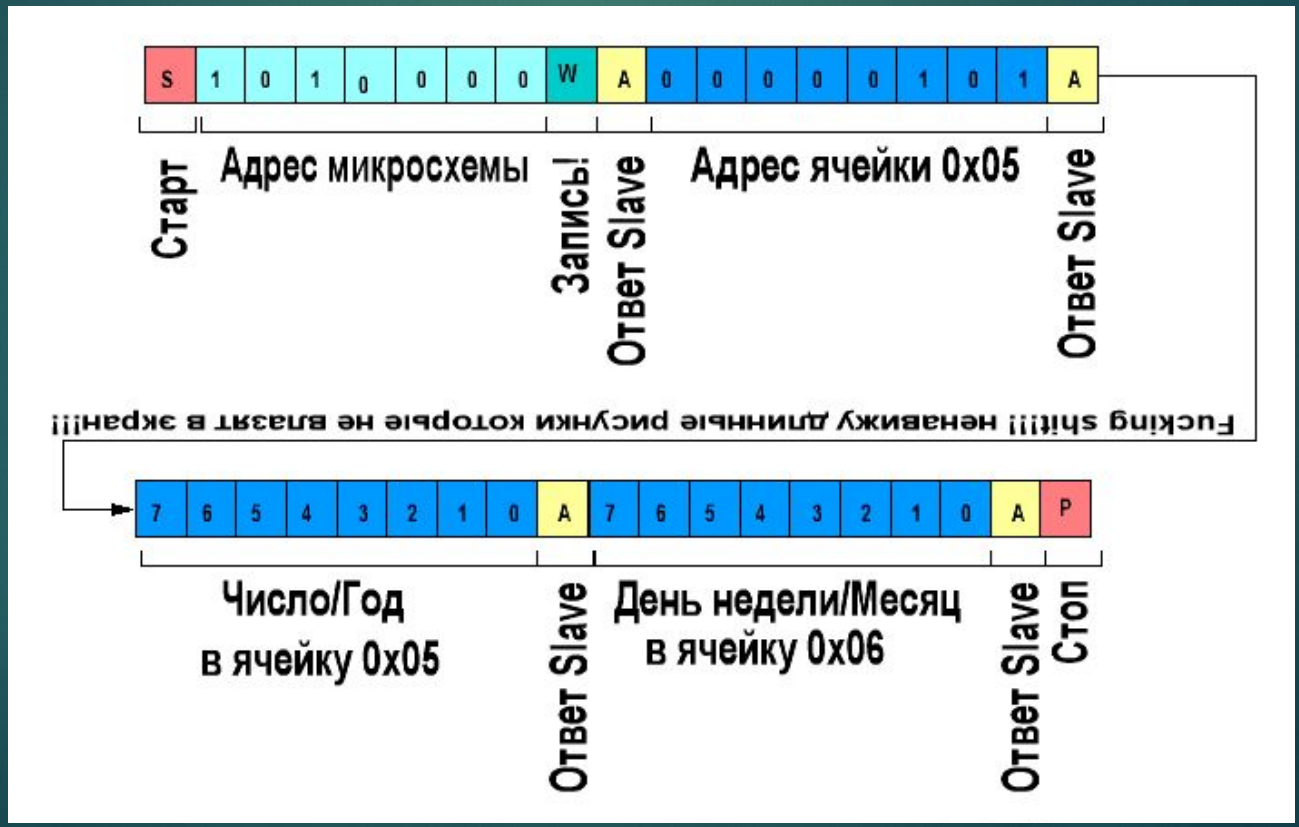
I2C - ЛОГИКА

Как видно из протокола, в первом байте мы **адресовываем само устройство**, а их может быть до 127 штук. Но в самом устройстве вполне может быть очень сложная структура, с кучей ячеек. Как обращаться с этими данными? Не считывать же все по очереди от нуля до конца — это долго. Поэтому приняли хитрый формат. Это не стандарт, но юзается повсеместно.



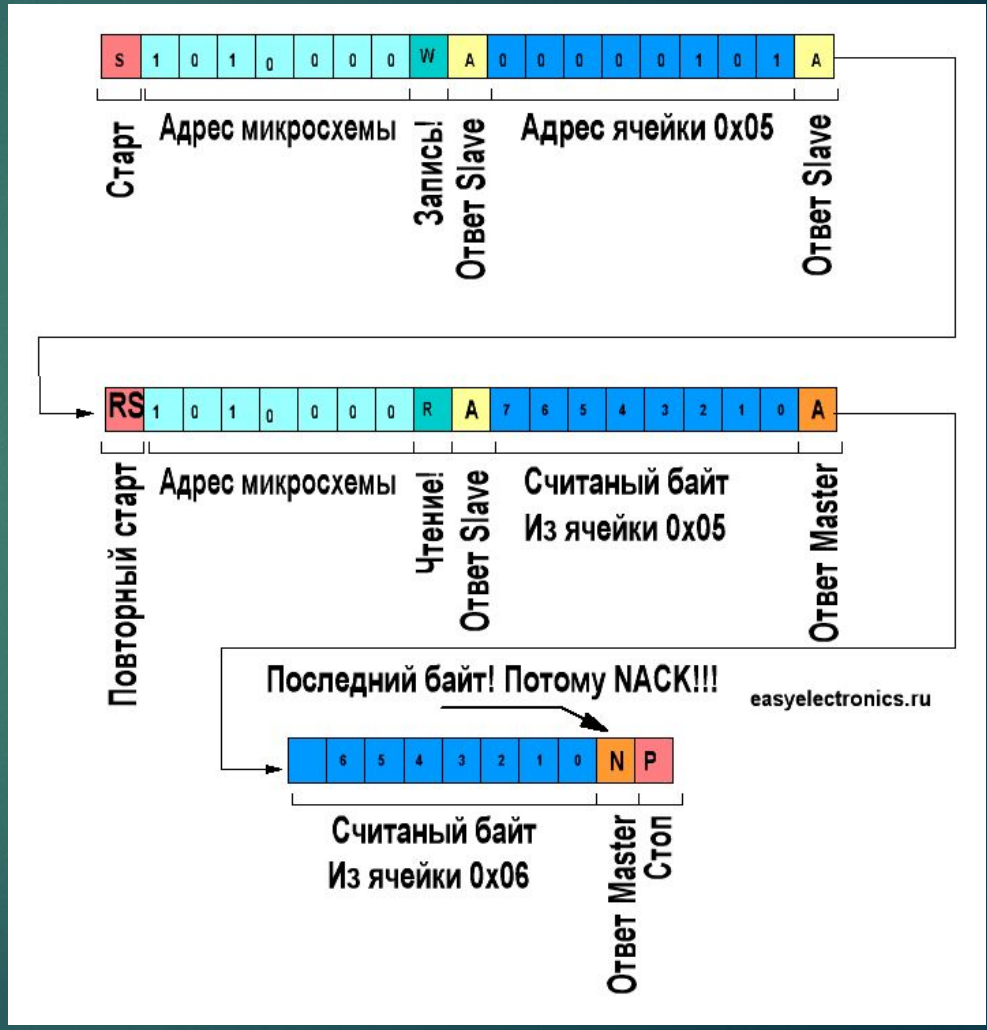
I2C - ЛОГИКА

Надо мне записать две ячейки памяти с адресами **0x05 и 0x06**. Как это сделать? Из даташита я узнаю, что **первый байт данных это адрес куда** мы будем обращаться, а потом уже идут данные и со следующим байтом счетчик адреса увеличивается на 1.



I2C - ЛОГИКА

Надо нам **считать** те же данные, а вдруг изменились? С записью все понятно. А с чтением как? А с чтением все через запись.
То есть, мы, вначале, записываем один байт — адрес. Потом делаем повторный старт, затем снова обращаемся к часам по ее **Slave-адресу**, **но уже с битом R**, на чтение. И умная микруха выдает нам байты с адреса который мы в нее вот только что записали. Выглядит это так:



ВОТ И ВСЕ

Датчики

Датчик — собирательный термин, который может означать: измерительный преобразователь; первичный измерительный преобразователь; чувствительный элемент.

В российских рамках стандартизации **датчик** является средством измерений

Они бывают:

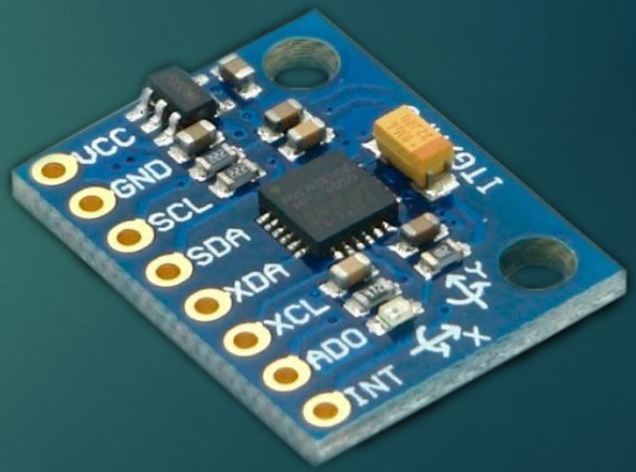
- Газа
- Пространства (Акселерометр, гироскоп, магнитометр, датчик линии, датчик наклона и тд)
- Климатические (Барометр, датчик влажности, датчик температуры)
- Датчики света (Освещенности, пламени, жестов, отражения и тд)
- Датчики шума

GY-521

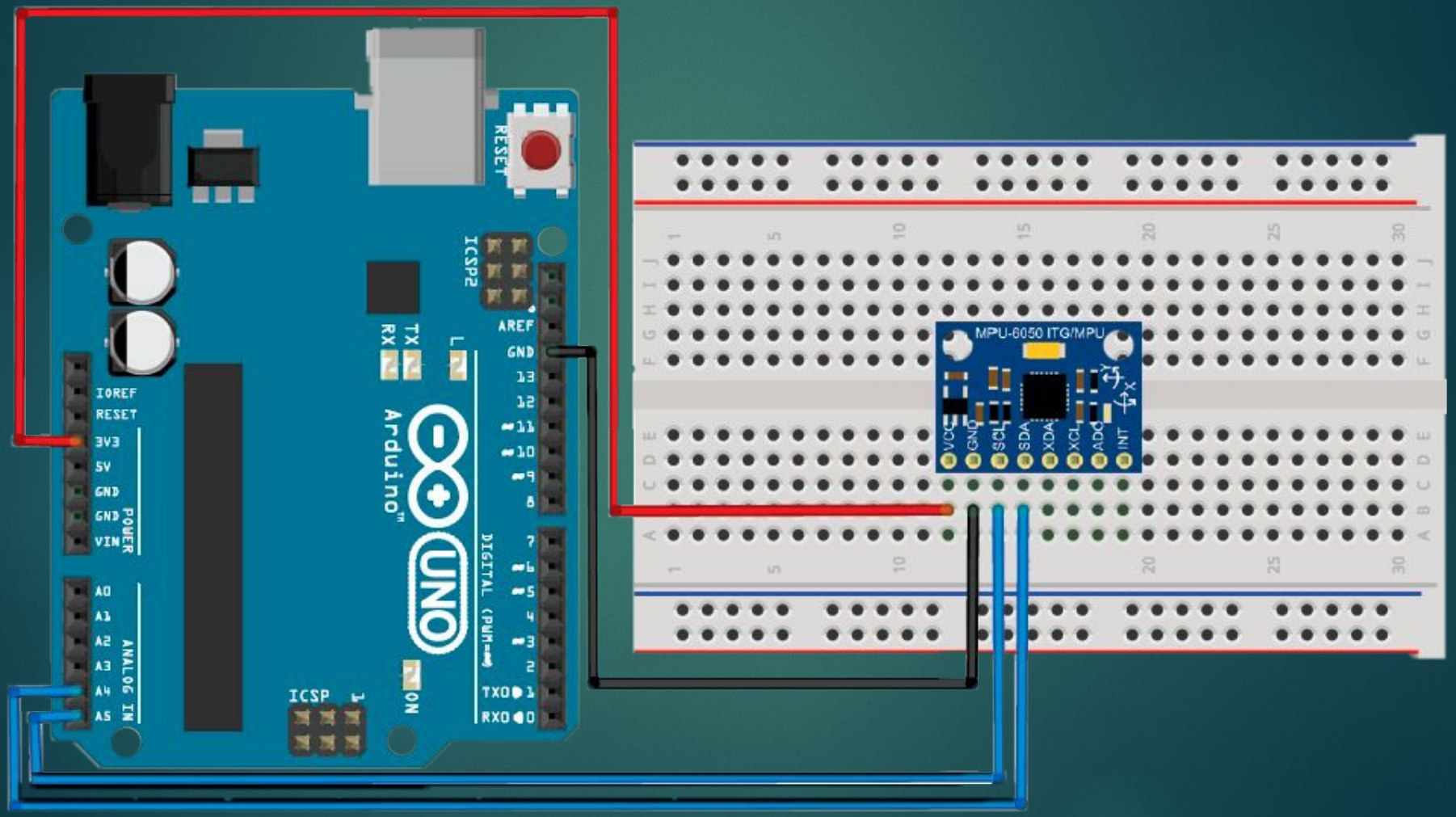
Модуль 3-х осевого гироскопа + 3-х осевого акселерометра GY-521 на чипе MPU-6050. Позволяет определить положение и перемещение объекта в пространстве, угловую скорость при вращении. Так же имеет встроенный датчик температуры. Используется в различных коптерах и авиамоделях, так же на основе этих датчиков можно собрать систему захвата движений.

Характеристики:

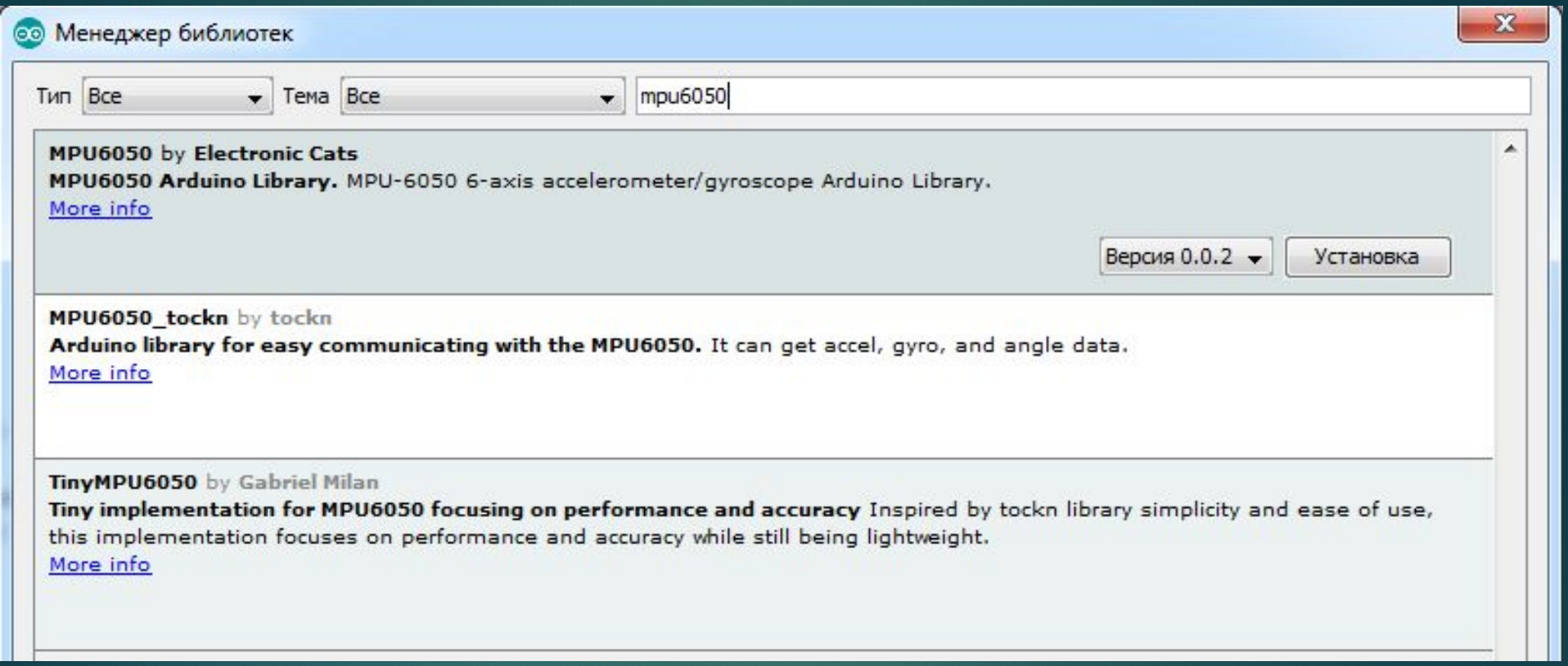
Микросхема: MPU-6050
Напряжение питания: от 3,5V до 6V (DC);
Диапазон гироскопа: ± 250 500 1000 2000 ° / с
Диапазон акселерометра: ± 2 ± 4 ± 8 ± 16g
Интерфейс связи: I2C
Размер: 15x20 мм.
Вес: 5 г



Подключение



Подключение



Написание кода

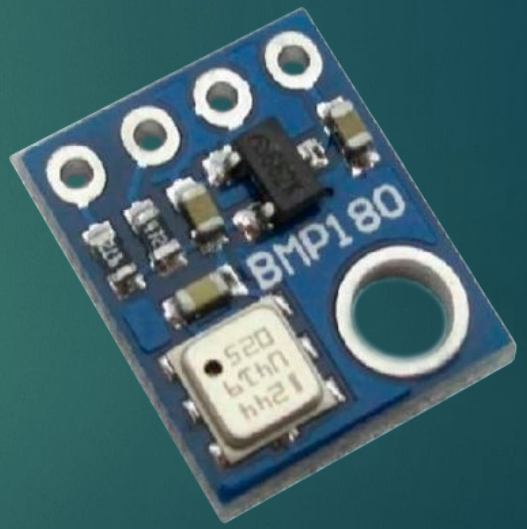
MPU6050

```
#include "Wire.h"
#include "MPU6050.h"

MPU6050 accelgyro;

int16_t ax, ay, az;
int16_t gx, gy, gz;

void setup() {
    Wire.begin();
    Serial.begin(9600);
    accelgyro.initialize();
    Serial.println("Testing device connections...");
    Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");
    accelgyro.setFullScaleAccelRange(3);
}

void loop() {

    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    Serial.print("a/g:\t");
    Serial.print(ax); Serial.print("\t");
    Serial.print(ay); Serial.print("\t");
    Serial.print(az); Serial.print("\t");
    Serial.print(gx); Serial.print("\t");
    Serial.print(gy); Serial.print("\t");
    Serial.println(gz);
}
```

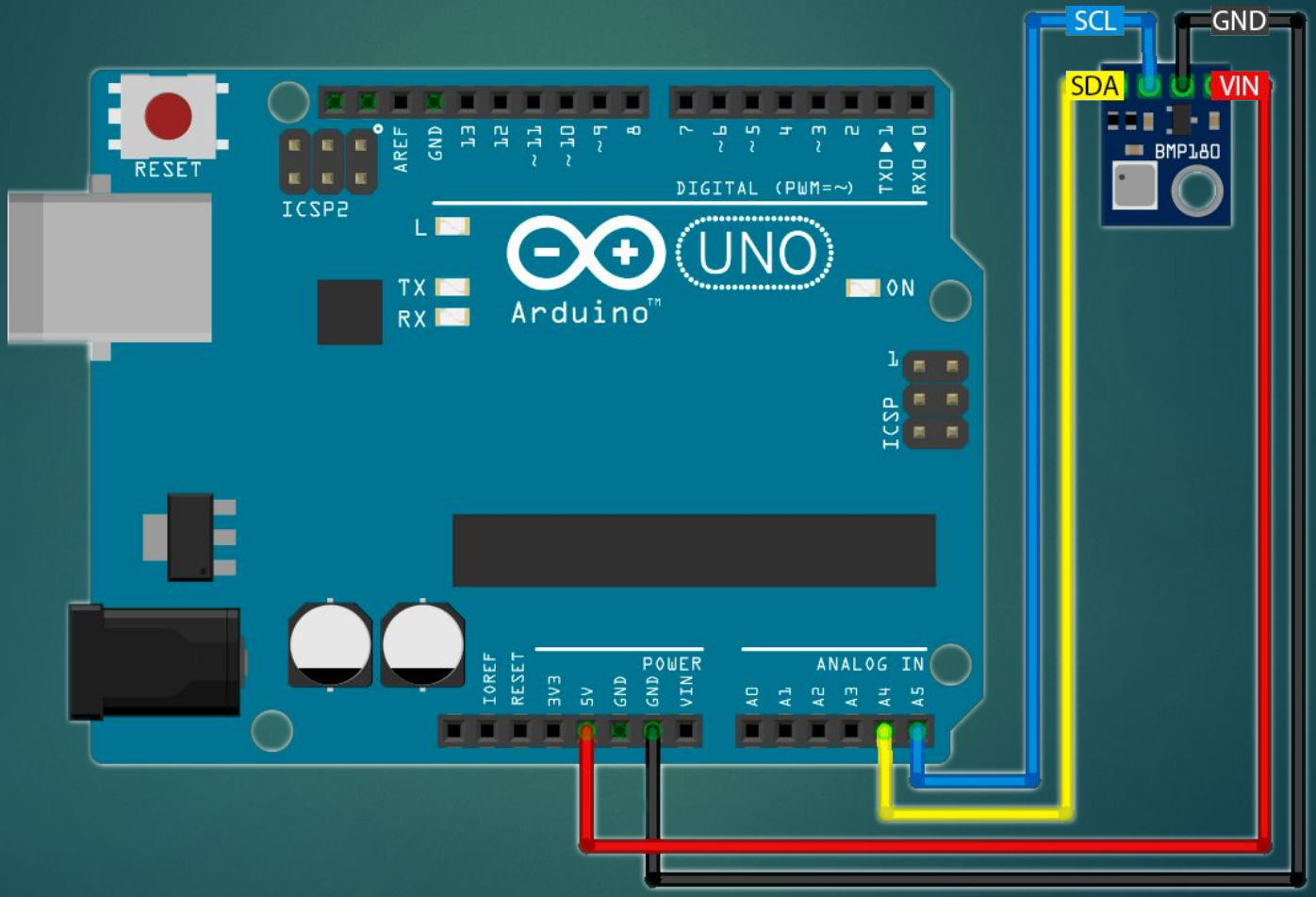
GY-68

Барометр GY-68 (датчик атмосферного давления и температуры на чипе BMP180). Отлично подходит для создания погодных станций и систем "умный дом"

Характеристики:
Чип: Bosh BMP180
Питание: 1.8-3.6V
Потребляемый ток:0.5uA на 1Hz
Интерфейс: I2C
Максимальная частота I2C: 3.5Mhz
Точность давления до 0.02hPa (17см)
Диапазон измерения давления: от 300hPa до 1100hPa (от +9000м до -500м)
Диапазон измерения температуры: от -40 до 85 градусов по Цельсию
Вес: 1.18г



Подключение



```
#include <SFE_BMP180.h>
#include <Wire.h>
SFE_BMP180 pressure;
double baseline;
void setup()
{
  Serial.begin(9600);
  if (pressure.begin())
    Serial.println("BMP180 init success");
  else
  {
    Serial.println("BMP180 init fail (disconnected?)\n\n");
    while(1);
  }
  baseline = getPressure();
  Serial.print("baseline pressure: ");
  Serial.print(baseline);
  Serial.println(" mb");
}
void loop()
{
  double a,P;
  P = getPressure();
  a = pressure.altitude(P,baseline);
  Serial.print("relative altitude: ");
  if (a >= 0.0) Serial.print(" ");
  Serial.print(a,1);
  Serial.println(" meters, ");
  delay(500);
}
```

```
double getPressure()
{
  char status;
  double T,P,p0,a;
  status = pressure.startTemperature();
  if (status != 0)
  {
    delay(status);
    status = pressure.getTemperature(T);
    if (status != 0)
    {
      status = pressure.startPressure(3);
      if (status != 0)
      {
        delay(status);
        status = pressure.getPressure(P,T);
        if (status != 0)
        {
          return(P);
        }
        else Serial.println("error retrieving pressure measurement\n");
      }
      else Serial.println("error starting pressure measurement\n");
    }
    else Serial.println("error retrieving temperature measurement\n");
  }
  else Serial.println("error starting temperature measurement\n");
}
```