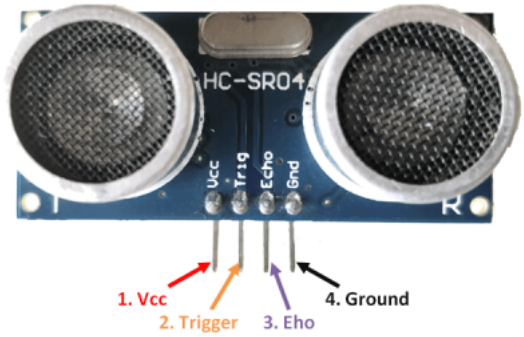
Конструктор спутника "ОрбиКрафт"

Дистанционное сканирование с помощью УЗ дальномера.

Знакомство с ультразвуковым дальномером HC-SR04.

Ультразвуковой дальномер HC-SR04 определяет расстояние до объектов с использованием ультразвука частотой 40 кГц. Таким же способом это делают летучие мыши и дельфины. Он излучает звук на частоте 40 кГц и слушает отраженное эхо. По времени движения звуковой волны туда и обратно рассчитывается расстояние до предмета.

На показания ультразвукового дальномера не влияет засветка от солнца и цвет предмета. Он позволяет обнаружить даже прозрачную поверхность предмета. Испытывает сложности с измерением расстояний до пушистых предметов.



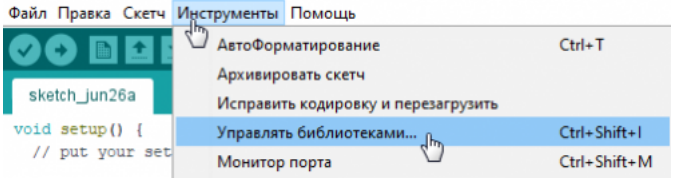
Конфигурация выводов дальномера HC-SR04

Номер вывода	Имя вывода	Описание
1	Vcc	Подключение питания +5V
2	Trigger	Триггер – это входной вывод. Для запуска измерения необходимо подать на этот вход логическую единицу на 10 микросекунд. Следующее измерение рекомендуется выполнять не ранее чем через 50 микросекунд.
3	Echo	Эхо – это выходной вывод. После завершения измерения, на этот выход будет подана логическая единица на время, пропорциональное расстоянию до объекта.
4	Ground	Подключения земли питания.

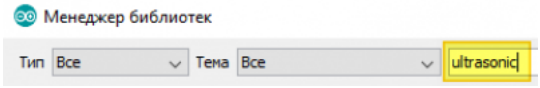
Параметры HC-SR04

- Напряжение питания: 5 В
- Потребление в режиме тишины: 2 мА
- Потребление при работе: 15 мА
- Диапазон измерения расстояний: от 5 до 400 см
- Угол наблюдения: 30°

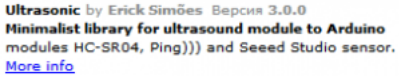
Для работы с ультразвуковым дальномером необходимо установить библиотеку Ultrasonic от разработчика Erick Simões. Откройте меню Инструменты и выберите раздел Управления библиотеками.



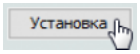
Введите в строку поиска слово «ultrasonic».



Прокрутите перечень библиотек и найдите библиотеку Ultrasonic by Erick Simões



Нажмите на кнопку Установки.



Теперь можно открыть тестовый скетч в разделе меню Файл – Примеры – Ultrasonic и протестировать работу датчика.

Пример кода программы для Arduino

Ultrasonic.ino

```
#include <OrbicraftBus.h>
#include <Ultrasonic.h>

 * Module HR-SC04 (four pins)
 * ----------------------
 * | HC-SC04 | Arduino |
 * ----------------------
 * |   Vcc   |   5V    |
 * |   Trig  |   6     |
 * |   Echo  |   7     |
 * |   Gnd   |   GND   |
 * ----------------------

Message msg;
OrbicraftBus bus;
Ultrasonic ultrasonic(6, 7);  // подключаем  HC-SR04 к пинам 6 (Trig) и 7 (Echo)
int distance;
int16_t msgSize = 0;

void setup() {
  Serial1.begin(9600); // задаем скорость обмена информацией по Serial1   !!!
}

void loop() {
  distance = ultrasonic.read();  // считываем расстояние
  msgSize = bus.takeMessage(msg); // пробуем прочитать сообщение с помощью метода takeMessage

  if (msgSize > 0){ //если сообщение есть
    switch (msg.id){//в зависимости от идентификатора сообщения выполняем те или иные действия

      // Рассмотрим случай с идентификатором 2
        case 0x02:{
        String data = String(distance); // записываем показания датчика расстояния в переменную data
        bus.sendMessage(bus.obcAddress, 0, data); // передаем содержимое переменной data на БКУ
        break;
      }
    }
  }
}
// Следующий блок кода необходимо всегда добавлять в конец программы
// Функция вызывается автоматически и необходима для обработки сообщения
void serialEvent2() {
  bus.serialEventProcess();
}
```

Пример кода программы для Орбикрафт

Scan.c

```
#include "libschsat.h"
const float kd = 200.0;

// Временной шаг работы алгоритма, с
const float time_step = 0.1;

// Целевая угловая скорость спутника, град/с.
// Для режима стабилизации равна 0.0.
const float omega_goal = 5.0;

// Номер маховика
const int mtr_num = 1;

// Максимально допустимая скорость маховика, об/мин
const int mtr_max_speed = 3000;

// Номер ДУС (датчика угловой скорости)
const  uint16_t hyr_num = 1;
int16_t mtr_new_speed;
int motor_new_speed_PD(int mtr_speed, int omega, float omega_goal){
 /* Функция для определения новой скорости маховика.
 Новая скорость маховика складывается из
 текущей скорости маховика и приращения скорости.
 Приращение скорости пропорционально ошибке по углу и ошибке по угловой скорости.
 mtr_speed - текущая угловая скорость маховика, об/мин
 omega - текущая угловая скорость спутника, град/с
 omega_goal - целевая угловая скорость спутника, град/с
 mtr_new_speed - требуемая угловая скорость маховика, об/мин*/
```

```
        mtr_new_speed = (int)(mtr_speed + kd * (omega - omega_goal));

        if (mtr_new_speed > mtr_max_speed)
        {
                mtr_new_speed = mtr_max_speed;
        }
        else if (mtr_new_speed < -mtr_max_speed)
        {
                mtr_new_speed = -mtr_max_speed;
        }
        return mtr_new_speed;
}

void initialize_all(void){
/*Функция включает все приборы,которые будут использоваться в основной программе.*/
        printf("Enable motor №%d\n", mtr_num);
        motor_turn_on(mtr_num);
        Sleep(1);
        printf("Enable angular velocity sensor №%d\n", hyr_num);
        hyro_turn_on(hyr_num);
        Sleep(1);
 }

void switch_off_all(void){
/* Функция отключает все приборы,которые будут использоваться в основной программе.*/
        printf("Finishing...");
        int16_t new_speed = mtr_new_speed; //
        printf("\nDisable angular velocity sensor №%d\n", hyr_num);
        hyro_turn_off(hyr_num);
        motor_set_speed(mtr_num, 0, &new_speed);
        Sleep (1);
        motor_turn_off(mtr_num);
        printf("Finish program\n");
}

void control(void){
        char answer[255]; // Создаем массив для сохранения ответа
        //nt32_t count = 100; // Устанавливаем счетчик на 5 шагов

        initialize_all();
        // Инициализируем статус маховика
        int16_t mtr_state;
        // Инициализируем статус ДУС
        int16_t hyro_state = 0;
        int16_t pRAW_dataX = 0;
        int16_t pRAW_dataY = 0;
        int16_t pRAW_dataZ = 0;
        int16_t *gx_raw = &pRAW_dataX;
        int16_t *gy_raw = &pRAW_dataY;
        int16_t *gz_raw = &pRAW_dataZ;
        int16_t speed = 0;
        int16_t *mtr_speed = &speed;
        int16_t omega;
        int i;
        for(i = 0; i < 500; i++){

                int status = arduino_send(0, 2, NULL, answer, 100);
                if (status == 0){
                        printf("%s\r\n", answer);
                        }
                else{
                                printf("ErrorSend\r\n");
                        }

                //printf("i = %d\n", i);
                // Опрос датчика угловой скорости и маховика.
                hyro_state = hyro_request_raw(hyr_num, gx_raw, gy_raw, gz_raw);
                mtr_state = motor_request_speed(mtr_num, mtr_speed);

                // Обработка показаний датчика угловой скорости,
                // вычисление угловой скорости спутника по показаниям ДУС.
                // Если код ошибки ДУС равен 0, т.е. ошибки нет
                if (!hyro_state){
                        float gz_degs = *gz_raw * 0.00875;
                        omega = gz_degs;
                }
                else if (hyro_state == 1){
                        printf("Fail because of access error, check the connection\n");
                }
                else if (hyro_state == 2){
                        printf("Fail because of interface error, check your code\n");
                }

                int mtr_new_speed;
                //Обработка показаний маховика и установка требуемой угловой скорости.
                if (!mtr_state) {
                        // если код ошибки 0, т.е. ошибки нет
                        int16_t mtr_speed=0;
                        motor_request_speed(mtr_num, &mtr_speed);
                        //printf("Motor_speed: %d\n", mtr_speed);
                // установка новой скорости маховика
                        mtr_new_speed = motor_new_speed_PD(mtr_speed,omega,omega_goal);
                        motor_set_speed(mtr_num, mtr_new_speed, &omega);
                }
        }
        Sleep(time_step);
        switch_off_all();
}
```

Анализ полученных данных

Результатом работы программы будет большой массив данных, выведенных Орбикрафтом на экран. Для обработки этих данных следует использовать Excel. Скопируйте данные, и вставьте их в один столбец Excel. Постройте график высот на основе полученных данных.