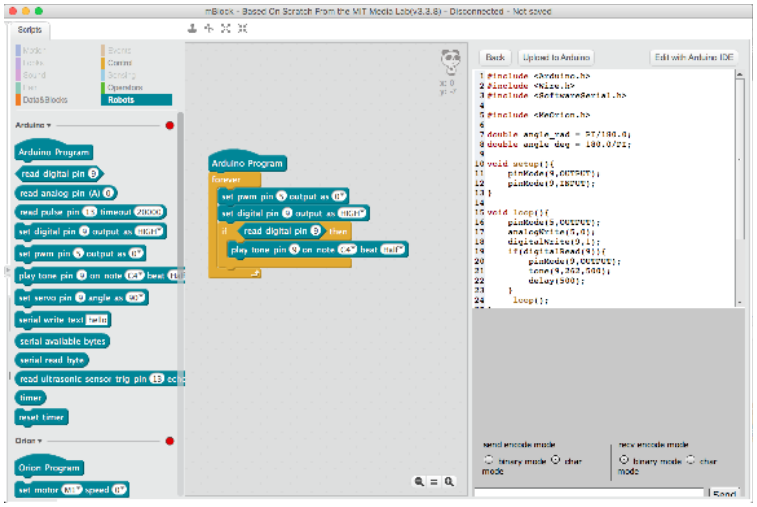
Создание расширений для mBlock

by Wang Yu, Product Manager at Makeblock

Как работают расширения

Расширения позволяют создавать свои блоки для mBlock. Вы можете использовать расширения для подключения сторонних датчиков или других робототехнических продуктов, таких как LEGO или LittleBits.

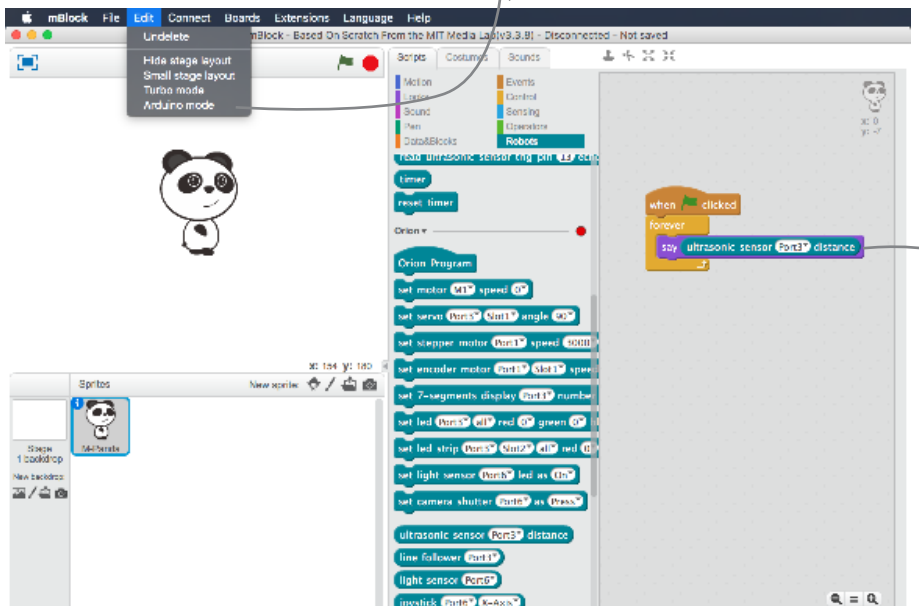
Писать расширения для mBlock может кто угодно. Это позволяет mBlock быть прекрасной платформой для любого типа аппаратно-зависимого программирования.



Режим Scratch и режим Arduino

Каждый блок в mBlock работает в двух режимах: **режим Scratch** и **режим Arduino**. Важно знать разницу между ними, прежде чем писать расширения.

используйте пункт меню **Редактирование \Режим Arduino** для переключения между режимами Scratch и Arduino.



▸ Режим **Scratch**

В режиме Scratch, робот или плата Arduino **должны быть подключены к компьютеру** для того, чтобы блоки подраздела Arduino\Робот работали. Вы **можете использовать** блоки Scratch для работы с графикой или для создания игр **в данном режиме**.

▴ Режим **Arduino**

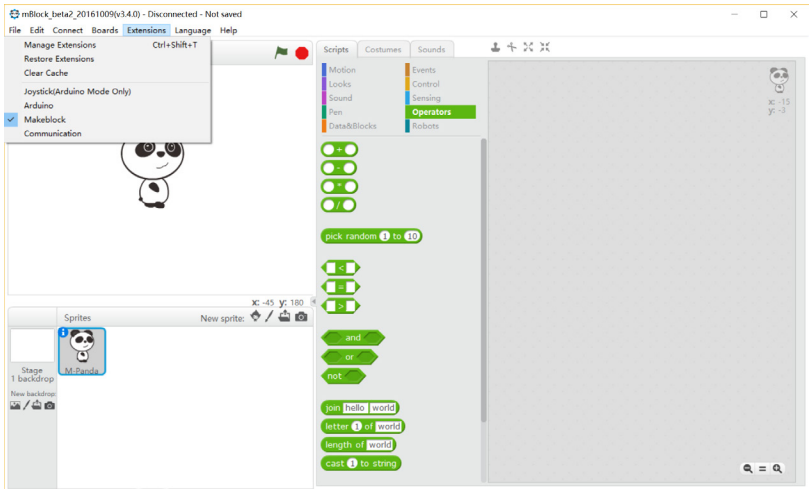
В режиме Arduino, **программа загружается в робота** или плату Arduino, после чего устройство может **работать само** по себе, без компьютера. Однако блоки Scratch для работы с графикой **уже нельзя будет использовать**, поскольку не будет связи с компьютером.

блок **"Say/Говорить"** может быть использован только в режиме Scratch; в то время как блок **"Forver/Всегда"** можно использовать в обоих режимах.

Использование расширения

Через менеджер расширений, Вы можете просматривать расширения, найти тот, который Вам нужен и получить описанные в найденном расширении блоки в один клик.
Следующие инструкции показывают, как добавить одно из расширений в mBlock.

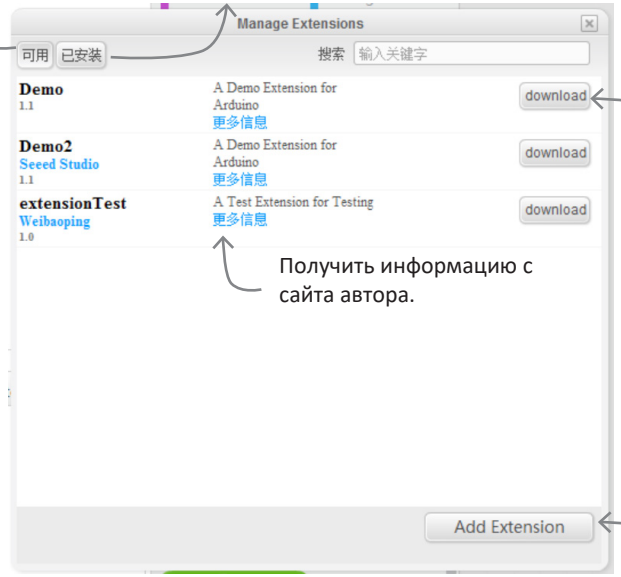
1 Используйте пункт меню "Расширения", "Управление расширениями" чтобы открыть Менеджер расширений.



2 Вы можете искать расширения используя поле поиск. Нажмите кнопку “Загрузить” справа от элемента таблицы, чтобы скачать расширение. Т**ребуется подключение к интернету.**

Расширения, которые Вы можете загрузить через интернет

Расширения, установленные на этом компьютере. Вы можете обновлять или удалять эти расширения.

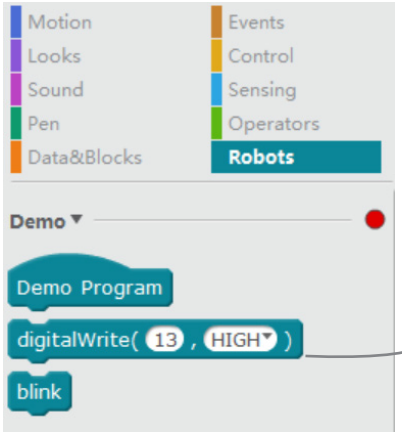


Кликните сюда, чтобы загрузить.

Получить информацию с сайта автора.

Установить расширение из .zip файла, расположенного на компьютере

3 Скачанные/установленные Вами расширения будут отображаться в подгруппе "Робот/Arduino".



Эти блоки из расширения "Demo", которое можно просто скачать.

Написание своего расширения

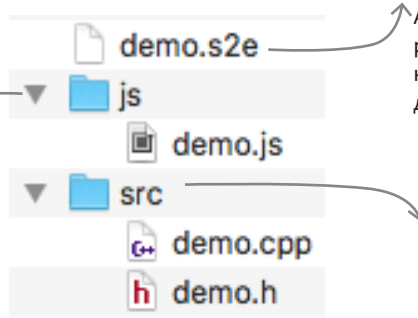
Писать расширения не так сложно, как кажется. Обычно это делается "в виде" редактирования текстовых файлов. Знания **Arduino IDE** необходимы для реализации режима Arduino, а знания **JavaScript** необходимы для реализации режима Scratch. Вы можете пропустить один из режимов, и блок просто не будет работать в данном режиме.

Вот перечень требований при написании расширений:

- Записать основную информацию, такую как название и автор
- Определиться с тем, как блок должен выглядеть
- Описать для mBlock как генерировать код в Arduino IDE
- Описать работу функций в режиме Scratch
- Включить дополнительный Arduino заголовок / C файлов.

Структура файлов расширения

Каждое расширение - это папка с файлами сжатая в .zip файл. Это пример папки, полученной при разархивировании расширения "Demo":



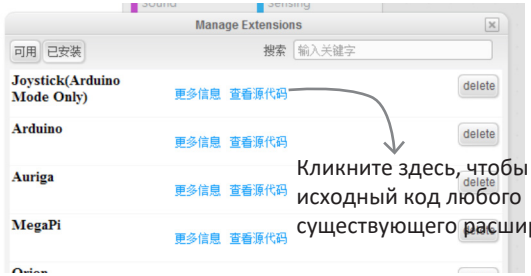
файл .s2e: основное описание и определение блоков расширения, код Arduino IDE генерируемый расширением в режиме Arduino. "S2E" расшифровывается как "Scratch 2 Extension" т.е. "файл скрэтч для расширений".

в папке "j s" находится код JavaScr ipt , который выполняется в режиме Scratch

в папке "src" дополнительные файлы библиотек, включаемые в режиме Arduino. Вы можете пропустить их, если не используете дополнительные библиотеки.

Советы

Исходный код существующих расширений всегда лучший ресурс при написании нового расширенияю В mBlock Вы всегда можете посмотреть исходный код существующих расширений.



Кликните здесь, чтобы посмотреть исходный код любого существующего расширения.

"Смотреть исходник" вызывает окно с папкой этого расширения. Вы можете попробовать изменить код расширения и изменения вступят в силу при следующем запуске mBlock (это один из способов отладки). Однако, **любые изменения, сделанные сдесь, будут потеряны при обновлении mBlock или выполнении команды "Очистить кэш".**

1 Заполнение базовой информации

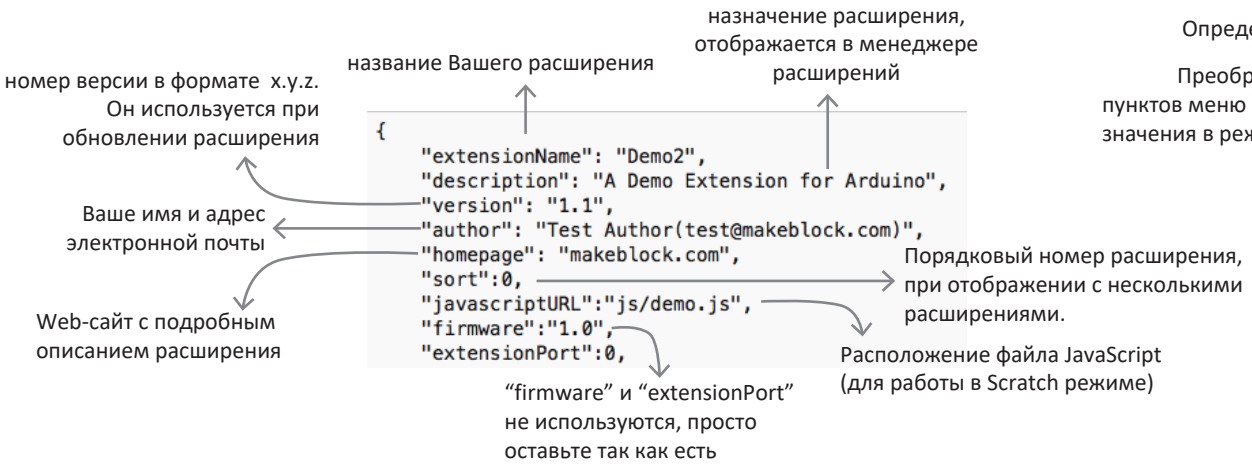
Для начала скачайте расширение "Demo" отсюда:

http://www.mblock.cc/extensions/

https://yadi.sk/d/doRccpBS3Kmy3J

Следующий шаг - редактирование s2e файла. Вам необходим текстовый редактор. Стандартный "Блокнот" подойдет, но рекомендуется использовать специальный редактор, предназначенный для редактирования кода, например: Github Atom, Visual Studio Code, Notepad++ и т.п.

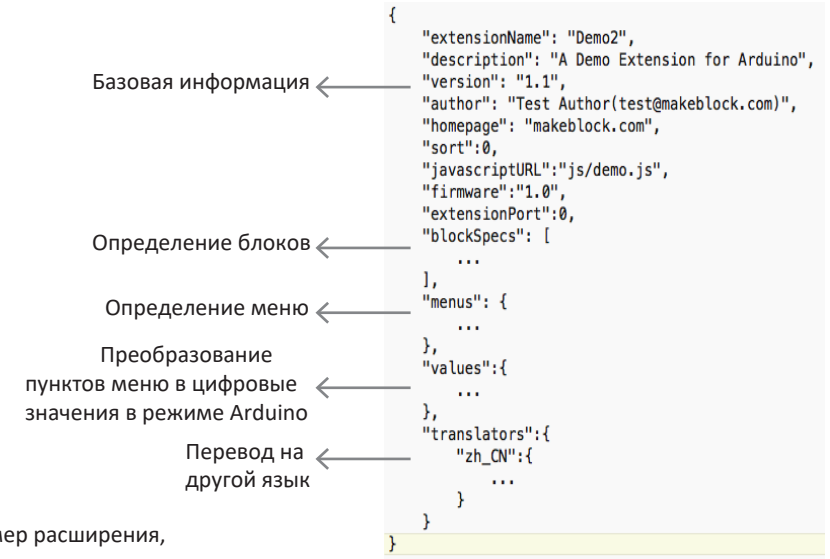
В начале файла .s2e находится базовая информация. Вы должны сообщить пользователю что делает данное расширение и кто написал его. Для примера смотрим первые строки файла demo.s2e:



Советы

Файл .s2e является главным файлом расширения. Помимо базовой информации, в файле определяются блоки, описываются для mBlock выпадающие меню блоков и приводится перевод расширения на другие языки (по желанию).

Если Вы знакомы с JavaScript, Вы можете заметить что используется простой формат JSON (JavaScript Object Notation) объектов.



2 Определение блоков

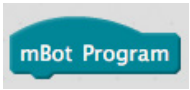
Секция "blockSpecs" описывает то как блоки выглядят и указывает как блоки ведут себя в режимах Scrath и Arduino.

```
"blockSpecs": [
    ["h","Demo Program","runArduino"],
    [
        "w",
        "digitalWrite( %n , %d.digital )",
        "digitalWrite",
        "13",
        "HIGH",
        {
            "setup":"pinMode({0},OUTPUT); \n",
            "inc":"",
            "def":"",
            "work":"digitalWrite({0},{1});\n",
            "loop":""
        }
    ],
    [
        "w",
        "blink",
        "blink",
        {
            "setup":"",
            "inc":"#include \"demo.h\"",
            "def":"DemoClass demo; \n",
            "work":"demo.blink(); \n",
            "loop":""
        }
    ]
],
```

Каждый блок описывается в виде массива JavaScript

▾ Типы блоков обозначаются одной буквой:

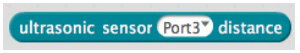
“h” означает **"Блок Заголовок"**; редко используется в пользовательских расширениях



“w” означает **"Блок Запись"**; блоки посылают команды оборудованию и не ожидают ответа

“r” означает **"Блок Чтение"**; в режиме Scratch блок ждет пока функция не ответит (ждет значение от функции); **"R"** - это асинхронное чтение - значение не возвращается функцией, но будет получено позже ("callback")через функцию в JavaScript

("r" или "R" в режиме Arduino не различаются)



“b” означает **"Бинарный Блок"**; он возвращает двоичное значение - "да" или "нет". Аналогично **"B" - это асинхронное чтение двоичного значения**

▾ Текст, отображающийся на блоках

слова, начинающиеся с символа "%" это параметры - "слоты", которые пользователь может заполнить сам или вписать в него другой блок. Этот блок (“digitalWrite(%n , %d.digital)”) выглядит так:

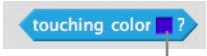
%n дает круглый слот для чисел и передает число в режиме Arduino.

%d.name дает круглое поле с выпадающим меню и передает число в режиме Arduino. Содержание выпадающего меню определяется в секции "menu" и идентифицируется по имени.

Другие типы параметров:

%s дает прямоугольный слот для ввода текста (строки)

%m.name дает прямоугольнъое выпадающее меню со строками



%c дает палитру цветов

3 Создание меню

блок“digitalWrite” имеет удобное меню для выбора HIGH или LOW выходного значения. Оно определяется следующим образом:


```
"blockSpecs": [
    ["h","Demo Program","runArduino"],
    [
        "w",
        "digitalWrite( %n , %d.digital )",
        "digitalWrite",
        "13",
        "HIGH",
        {
            "setup":"pinMode({0},OUTPUT); \n",
            "inc":"",
            "def":"",
            "work":"digitalWrite({0},{1});\n",
            "loop":""
        }
    ],
    [
        "w",
        "blink",
        "blink",
        {
            "setup":"",
            "inc":"#include \"demo.h\"",
            "def":"DemoClass demo; \n",
            "work":"demo.blink(); \n",
            "loop":""
        }
    ]
],
"menus": {
    "digital":["HIGH","LOW"]
},
"values":{
    "HIGH":1,
    "LOW":0
},
```

“%d.digital” - означает что информация для данного меню хранится в списке “digital” в разделе всех меню

символы \" в описании кода блока - дают символ кавычек в генерируемом коде **"inc":"#include \"demo.h\""** в результате даёт **#include "demo.h"**

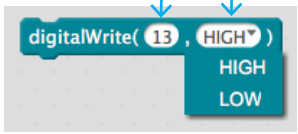
это означает что меню имеет 2 пункта (строки): **“HIGH”** и **“LOW”**

в **режиме Arduino**, **“HIGH”** принимает значение 1, а **“LOW”** значение 0. Это не касается режима Scratch.

4 Заканчиванием с определением блоков

задайте имя функции **Javascript** используемой в режиме **Scratch**

```
"blockSpecs": [
    ["h","Demo Program","runArduino"],
    [
        "w",
        "digitalWrite( %n , %d.digital )",
        "digitalWrite",
        "13",
        "HIGH",
        {
            "setup":"pinMode({0},OUTPUT); \r
            "inc":"",
            "def":"",
            "work":"digitalWrite({0},{1});\r
            "loop":""
        }
    ],
```



задайте **значения по умолчанию** для блока, который вы создали

5 Генерация кода для Arduino

При определении каждого блока, в .s2e файле также определяется то, как будет генерироваться код для Arduino, по нижеприведенным правилам:


```
[
    "w",
    "set servo pin %n angle as %d.servovalue",
    "runServoArduino",
    "9",
    90,
    {
        "inc":"#include <Servo.h>\n",
        "def":"Servo servo_{0};\n",
        "setup":"servo_{0}.attach({0});\n",
        "work":"servo_{0}.write({1});\n",
        "loop":""
    }
],
```

вставка директив и инструкций

определение **переменных**

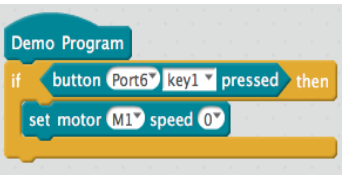
однократная вставка в функцию setup()

может вставляться **много раз** при каждом использовании блока (в том месте где используется блок)

{0} означает первый параметр блока, "9" в данном случае

{1} означает второй параметр блока, здесь - "90"

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include <Servo.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
Servo servo_9;

void setup(){
    servo_9.attach(9);
    for(int __i__=0;__i__<10;++__i__)
    {
        servo_9.write(90);
    }
}

void loop(){
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
}
```

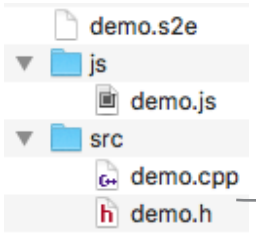



```
[
    "B",
    "button %d.blackPorts %m.button_keypressed",
    "getButton",
    "Port6",
    "key1",
    {
        "setup":"",
        "inc":"",
        "def":"Me4Button buttonSensor_{0}({0});\n",
        "work":"(buttonSensor_{0}.pressed()=={1})",
        "loop":"buttonSensor_{0}.pressed();"
    }
]
```

\n - означает конец строки и переход на новую в генерируемом коде

однократная вставка в функцию loop()

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
Me4Button buttonSensor_6(6);
MeDCMotor motor_9(9);

void setup(){
    if((buttonSensor_6.pressed()==1)){
        motor_9.run(0);
    }
}

void loop(){
    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
    buttonSensor_6.pressed();
}
```



если вы включаете какие-либо файлы с ресурсами (библиотеки например), они должны быть помещены в директорию **"src"**

6 Программирование режима Scratch

Файл Javascript довольно большой. Рекомендуется брать файл demo.js и при необходимости вносить изменения. Если Ваше расширение не поддерживает режим Scratch, пропустите этот раздел.

demo.s2e

```
{
    "extensionName": "Demo2",
    "description": "A Demo Extension for Arduino",
    "version": "1.1",
    "author": "Test Author(test@makeblock.com)",
    "homepage": "makeblock.com",
    "sort":0,
    "javascriptURL":"js/demo.js",
```

должно быть соответствие в имени файла

demo.js

```
// demo.js

(function(ext) {
    var device = null;

    var levels = {
        HIGH:1,
        LOW:0
    };

    ext.resetAll = function(){};

    ext.runArduino = function(){};

    ext.digitalWrite = function(pin,level) {
        device.send([pin, levels[level]])
    };

    ext.blink = function(){
        device.send([0x22, 0x23])
    }

    function processData(bytes) {
        trace(bytes);
    }

    // Extension API interactions
    var potentialDevices = [];
```

не изменять

здесь определяются Ваши переменные

setup code for the extension

здесь код для Вашего блока

при отсылке байтов через последовательный порт, используются массивы байтов

этот код вызывается каждый раз, когда компьютер принимает байты из последовательного порта

trace(string) это очень полезная для отладки функция, потому что она пишет логи в соответствующей панели режима Arduino.

demo.s2e

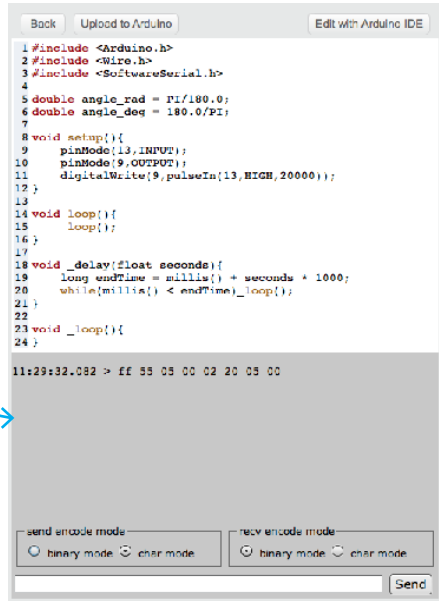
```
"blockSpecs": [
    ["h","Demo Program","runArduino"],
    [
        "w",
        "digitalWrite( %n , %d.digital )",
        "digitalWrite",
        "13",
        "HIGH",
        {
            "setup":"pinMode({0},OUTPUT); \n",
            "inc":"",
            "def":"",
            "work":"digitalWrite({0},{1});\n",
            "loop":""
        }
    ],
```

имя функции определенное здесь

(и в самом конце) demo.js

```
    ext._getStatus = function() {
        if(!device) return {status: 1, msg: 'demo disconnected'};
        return {status: 2, msg: 'demo connected'};
    }

    var descriptor = {};
    ScratchExtensions.register('demo', descriptor, ext, {type: 'serial'});
})({});
```

впишите здесь название Вашего расширения.



Публикация Ваших расширений

Если Вы удовлетворены Вашим расширением, заархивируйте его в .zip файл. В MacOS, кликните правой кнопкой на дирректории и выберите **“Compress xxx...”**; в Windows, кликните правой кнопкой на папке и выберите **“Отправить”**, затем **“Сжатая ZIP-папка”**

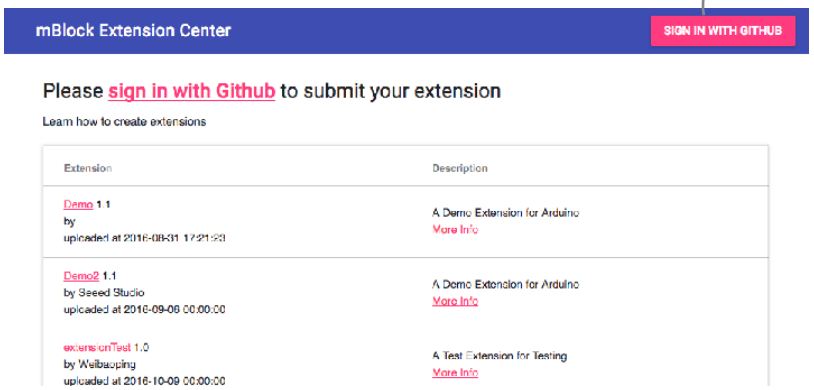
Далее Вы можете импортировать расширение кнопкой “Add Extension” в Менеджере расширений. Но для пользователей будет проще если Вы загрузите расширение в Онлан Центр (Online Extension Center).

1 Залогиньтесь с помощью Github аккаунта

Зайдите на сайт Онлайн Центра расширений:
http://www.mblock.cc/extensions/

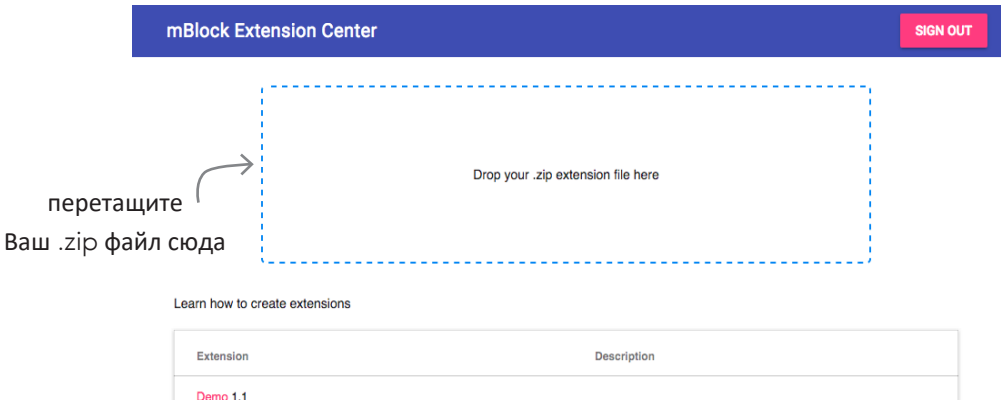
И кликните “Sign-in with Github”. Если у Вас нет Github аккаунта, Вы должны создать его (зарегистрироваться).

Кликните чтобы войти



2 Загрузка расширения в Онлайн Центр

После того как Вы залогинетесь, перетащите Ваш .zip файл в указанную область страницы (или кликните по данной области чтобы выбрать расширение через проводник)



Поздравляем! Вы загрузили своё расширение и внесли свой вклад в развитие мирового сообщества mBlock!

Совет

Если вы хотите обновить ваше расширение, просто загрузите .zip-файл с более высоким номером версии. И всё!

Совет

Makeblock, компания владеющая mBlock, оставляет за собой полное право удалять любое расширение, без объяснения причин. Но мы приветствуем любые расширения для любого продукта и надеемся, что достаточным будет просто модерировать спам (например названия вида “test123”) и незаконный контент. Надеемся на Ваш здравый смысл.

Совет ++

Как ни странно, но нельзя использовать комментарии в коде. В противном случае Вы не сможете загрузить расширение на сайт. Либо выполняйте очистку комментов перед загрузкой на сайте.