Содержание

Разработка модуля для устройства с BLE

---

1. Определения.

Модуль - это компонент, который позволяет вам контролировать один тип оборудования по его уникальному ID. Модуль расположен в облачном хранилище.
BLE - Bluetooth low energy, спецификация ядра беспроводной технологии Bluetooth, наиболее существенным достоинством которой является сверхмалое пиковое энергопотребление, среднее энергопотребление и энергопотребление в режиме простоя.
Виджет - визуальный компонент модуля, который будет отображен в комнате. Выделяют несколько основных способов создания виджетов:

- Создание в редакторе iRidium Studio.
- Динамическое создание при помощи скрипта.

В случае работы с виджетом и под-устройством через скрипт, Вы получаете полный контроль над жизненным циклом модуля и можете изменять его содержимое (удалять, создавать, редактировать и загружать).

Облачное хранилище - это управляемый сервер для загрузки готовых модулей.

Редактор iRidium Studio - графический редактор, который позволяет вам создавать модуль управления устройством для приложения i3 lite.

i3 lite - это приложение для создания и использования проектов i3 lite.

Проект i3 lite – проект автоматизации составленный из готовых модулей управления оборудования. Отличие i3 lite проекта от обычного проекта i2 Control в том, что проект i3 lite может изменяться динамически без предварительного редактирования на устройствах с ОС Windows. i3 lite проект состоит из небольших частей и при помощи встроенного конструктора в приложении i3 lite может собираться в единое целое и легко редактироваться. Отпадает ряд промежуточных процессов, таких как установка программ на ПК, загрузка проекта на панель, все происходит в одном месте на панели.

---

2. Как приступить к созданию модуля.

Для начала рассмотрим из каких частей состоит жизненный цикл модуля:

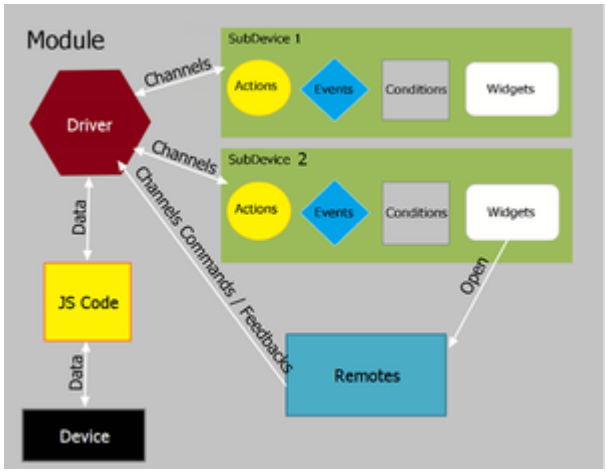
- Сбор требований (Поиск документации для разработки драйвера)
- Разработка графической части в iRidium Studio
- Разработка драйверной части (скрипты, устройства)
- Тестирование
- Публикация в облаке
- Использование (Загрузка модуля через приложение i3 из облака и добавление в проект автоматизации)

Модуль состоит из нескольких частей:

- Графической
- Драйверной
- Скриптовой

Создание модуля происходит в редакторе iRidium Studio. Что бы создать модуль, запустите редактор, выберите пункт *File / New /New Bongo Module*. После этого будет создан проект модуля, и можно приступать к разработке. При создании модуля указывается: имя проекта и тип ОС, на которую он (модуль) ориентирован (iOS, Android). Следует учесть что модуль может работать как на смартфоне, так и на планшетном компьютере. О настройке этого параметра речь пойдет в разделе "Разработка скриптовой части".

2.1. Создание SubDevice.



Модуль может содержать в себе несколько **SubDevice**. SubDevice могут быть однотипными или нет. Например 6 канальный диммер HDL, содержит в себе 6 диммируемых каналов, которые можно применить к 6 разным группам света для управления. Поэтому целесообразно выделить в модуле 6 SubDevice, для возможности создания макросов, событий и условий для каждой группы по отдельности. Создавая больше SubDevice, вы даете большую власть пользователю в управлении, но тем самым можете усложнить использования модуля.

Главная задача **SubDevice** - выделить управляемую сущность.

Рассмотрим из чего состоит **SubDevice**:

1. Условий (conditions)
2. Действий (Actions)
3. Событий (Events)
4. Виджетов (Widgets)

Для работы с модулями **обязательно** используется специальное событие **IR.EVENT_MODULE_START** которая позволяет разработчику получить уникальный идентификатор модуля и шины, которая использует модуль.

```
IR.AddListener(IR.EVENT_MODULE_START, 0, function(moduleID, busID){

//Тело модуля
}
```

Входящие параметры функции:
moduleID - уникальный идентификатор модуля.
busID - уникальный идентификатор шины используемый модулем.

---

3. BLE Модуль.

ВАЖНО отметить что BLE модуль будет работать только на устройстве поддерживающем BLE, т.е. не работает на WIndows. Протокол BLE строго структурирован по принципу своей коммуникации с другими устройствами. Вначале девайсы изучают доступные сервисы для отправки/принятия данных; неотъемлемая часть этих сервисов – их характеристики (characteristics), определяющие тип данных для будущей передачи. Характеристики, из соображений наглядности, могут иметь в своём составе описания-дескрипторы (descriptors), которые помогают определить тип данных.

Большинство API для Bluetooth LE позволяют искать локальные устройства и определять доступные в них сервисы, характеристики и дескрипторы.

3.1. BLE сканер.

BLE устройства можно найти сканером, для работы со сканером нужно создать соответствующий девайс и подключить его

```
var scanner= IR.CreateDevice(IR.DEVICE_BLE_SCANNER)
scanner.Connect();
```

Создадим лист на котором будем отображать найденные устройства и их данные

```
template = IR.CreateItem(IR.ITEM_POPUP, "template", 0, 0, 880, 32);
list = IR.CreateItem(IR.ITEM_LISTBOX, "list", 50, 50, 900, 500);
list.Template = "template";
```

Сканер получает uuid, rssi и имена найденных устройств.При нахождении сканером устройства срабатывает событие IR.EVENT_DEVICE_BLE_FOUND, это позволяет нам создать слушатель который будет выводить на наш лист все BLE устройства и их данные.

```
IR.AddListener(IR.EVENT_DEVICE_BLE_FOUND, 0, function(uuid, name, rssi,
scanRecord){

     list.CreateItem(list.ItemsCount, 0, {Text: "Device found uuid = " + uuid
```

```
+ " name = " + name});
     scanner.Disconnect();
  });
```

3.2. BLE девайс.

Чтобы работать с определенным BLE устройством нужно создать девайс и подключиться к нему
```
var device = IR.CreateDevice(IR.DEVICE_BLE_DEVICE, name, uuid);
device.Connect();
```
Входным параметром метода является:
Device_Type - Тип создаваемого девайса.
name - Имя устройства.
uuid - uuid устройства.
Имя и uuid должны полностью совпадать с данными устройства, их получение существенно упрощает сканер.

```
Так же полезно подписаться на некоторые события устройства

IR.AddListener(IR.EVENT_ONLINE, device, online, device); // подключение
устройства
IR.AddListener(IR.EVENT_OFFLINE, device, offline, device); // отключение
устройства
IR.AddListener(IR.EVENT_SERVICES_DISCOVERED, device, service_discovered,
device); // нахождение сервисов


IR.AddListener(EVENT, device, function, device); // подключение устройства
```
Входными параметрами слушателя являются:
EVENT - Событие после которого произойдёт вызов функция .
device - Устройство.
function - Вызываемая функция.

3.3. Сервисы, характеристики, дескрипторы.

Как было написано выше BLE устройства взаимодействуют через сервисы, чтобы узнать какие сервисы есть у устройства используется функция `DiscoverServices();`
получим сервисы устройства как только оно будет онлайн
```
function online(){

  this.DiscoverServices();

}
```
У нас есть слушатель получения сервиса, напишем вызываемую им функцию
```
function service_discovered(){

       IR.Log("service is descovered");
clearData();
```

```
var device = this;
// Получаем сервисы устройства
var services = device.GetServices();                          // out - array
of service objects
fillServiceList(services, "Uuid");
// Сохраняю сервисы
var IRdevice = findDevicebyName(device.Name);
IRdevice.services = services;
CurrentIR_Device = IRdevice;
CurrentIR_Device.TransportDevice = device;
IR.Log("Service length (" + services.length + ")");
for(var i = 0; i < services.length; i++){
IR.Log("Service Found (" + i + ")");
IR.Log("Service UUID: " + services[i].Uuid); // Печатаем Uuid
var Characteristics = services[i].GetCharacteristics(); // out - array of
Characteristi
// прилинковываю характеристики
IRdevice.services[i].Characteristics = Characteristics;
// Получаем характеристики сервиса
IR.Log("Characteristics length (" + Characteristics.length + ")");
for (var j = 0; j < Characteristics.length; j++){
IR.Log("Characteristics Found (" + j + ")");
IR.Log("Characteristics UUID" + Characteristics[j].Uuid); // Печатаем Uuid
IR.Log(Characteristics[j].Permissions); // Печатаем уровень разрешений
IR.Log(Characteristics[j].Properties); // Печатаем свойтство
IR.Log(Characteristics[j].Value); // Печатаем значения
// Подписываемся на уведомление об изменении характеристики
if (checkNotifyCharPermissions(Characteristics[j])){
device.SetCharacteristicNotification(Characteristics[j], true); // in -
object Characteristic, bool    | out - bool
}
// Если можно читать
if (checkReadCharPermissions(Characteristics[j])){
// Отправляем запрос на чтение характеристики
device.ReadCharacteristic(Characteristics[j]);        // in - object
Characteristic   | out - bool
}
                        // Получаем дескрипторы характеристики
var Descriptors = Characteristics[j].GetDescriptors(); // out - array of
Descriptors
// прилинковываю дескрипторы
IRdevice.services[i].Characteristics[j].Descriptors = Descriptors;
IR.Log("Descriptors length (" + Descriptors.length + ")");
for (var d = 0; d < Descriptors.length; d++){
// Отправляем запрос на чтение дескриптора
device.ReadDescriptor(Descriptors[d]); // in - object Descriptor  | out -
bool
IR.Log("Descriptors Found (" + d + ")");
IR.Log("Descriptors UUID (" + Descriptors[d].Uuid + ")");
IR.Log(Descriptors[d].Permissions); // Печатаем уровень разрешений
IR.Log(Descriptors[d].Value); // Печатаем значения
```

```
}
}
}
}
```

В результате мы получаем в лог количество сервисов устройства и их uuid, у каждого сервиса находим все характеристики выводим их количество и свойства в лог, так же для каждой характеристики находим дескрипторы выводим их количество и их свойства.
