Содержание

Разработка модуля с динамическими SubDevice's.

---

1. Определения.

Модуль - это компонент, который позволяет вам контролировать один тип оборудования по его уникальному ID. Модуль расположен в облачном хранилище. **Под-устройство (SubDevice)** - это часть устройства, которая выполняет определенный ряд задач и может быть выделена как отдельная сущность для управления. Например: у Вас есть камера с датчиком движения - это одно устройство, но здесь можно выделить две основные части - под-устройства, одно для передачи изображения (камера) и другое датчик движения. У каждого под-устройства можно выделить набор действий которые можно комбинировать в макросы (меж машинное взаимодействие):

1. У камеры получение потока изображения.
2. У датчика статус наличия движения.

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:

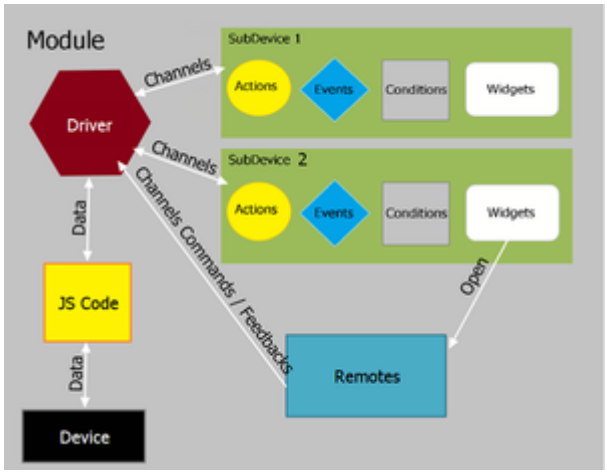
- Сбор требований (Поиск документации для разработки драйвера)
- Разработка графической части в iRidium Studio
- Разработка драйверной части (скрипты, устройства)
- Тестирование
- Публикация в облаке
- Использование (Загрузка модуля через приложение i3 из облака и добавление в проект автоматизации)

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.

Рассмотрим создание динамических под-устройств через скрипт на данном примере:

```
// Получаем уникальный идентификатор модуля и шины при запуске

IR.AddListener(IR.EVENT_MODULE_START, 0, function(moduleID, busID){
  // Получение копии модуля
  // moduleID - уникальный идентификатор модуля
  // Присваивание переменной module
  var module = B.getModule(moduleID);
  // Формируем имя подсети и присваиваем переменной netWorkName
  var netWorkName = "HDL-BUS Pro Network (UDP)" + busID;
  // Берем существующее устройство и присваиваем переменной device
```

```
  var device = IR.GetDevice(netWorkName);
  // Создаем новое под-устройство с параметрами:
  // Device - объект драйвера устройства iRidium
  // Имя создаваемого под-устройства, через которое будем к нему обращаться и
это же имя увидит пользователь
  var NewSubDevice = module.addSubDevice({
      Device: device,
      DeviceName: "HDL Dimmer 1"
   });
});
```

При успешном создании под-устройства, в переменной *NewSubDevice* будет объект созданного под-устройства, у которого мы можем выполнить следующие операции:

- Добавить зону влияния.

Зона влияния определяет на какую область влияет устройство, в отличии от позиции (комнаты) где располагается пульт управления, а само устройство может влиять на другие зоны. Для добавления зоны влияния можно использовать следующий метод:
`SubDevice.addLocation(`**`in_Location`**`)`
Входным параметром метода является: **in_Location** - объект комнаты.
Выходным параметром метода является: успешно или нет (**True**, **False**).

- Удалить зону влияния.

Для удаления зоны влияние можно использовать следующий метод:
`SubDevice.removeLocation(`**`in_Location`**`)`
Входным параметром метода является: **in_Location** - объект комнаты.
Выходным параметром метода является: успешно или нет (**True**, **False**).

- Получить данные от под-устройстве (загрузка кэша).

Данный метод служит для получения данных сохраненные в течении предыдущих сессий работы приложения. Вы можете использовать эту информацию для загрузки под-устройств, виджетов и любой другой информации, которая может вам потребоваться для визуализации или управления устройством. Для получения данных от под-устройства можно использовать следующий метод:
`SubDevice.getData()`
Выходным параметром метода является: массив или объект сохраненный в предыдущих сессиях.

- Установить данные об под-устройстве (сохранение кэша).

Данный метод служит для загрузки контейнера данных который вы можете использовать для сохранения на устройство, это массив или объект. Для отправки данных на под-устройство можно использовать следующий метод:
`SubDevice.setData(`**`in_data`**`)`
Входным параметром метода является: **in_data** - массив данных, либо объект.
Выходным параметром метода является: успешно или нет (**True**, **False**).
ВАЖНО!!! Для очистки кэша можно использовать следующий метод:
`B.clearEmulatorCahce(`**`bool`**`)`
Входным параметром метода является: **bool** - **true** или **false** (выполнять или нет)..

Выходным параметром метода является: успешно или нет (**True**, **False**).

- Добавить действие для под-устройства.

Действия (Actions), другими словами небольшой макрос, позволяющий отправить значение в канал команды драйвера. Механизм действий схож с графической связкой элементов и каналов драйвера, с единственной разницей, что на графический элемент обязательно надо нажать для того что бы драйвер выполнил команду. В случае действий, заданные значения, например 100 для канала Dimmer 1, можно будет активировать программно. Это образует начало межпрограммного взаимодействия различных драйверов без участия человека, что полностью соответствует концепции интернета вещей, как идеи меж машинного взаимодействия. Помимо этого сохраненные действия можно комбинировать между собой и образовывать динамические сценарии во время использования приложения, которые помимо машины, можно так же поместить на графический элемент и активировать пальцем.
Существует два вида Actions: **Simple** - простое и **Advanced** - расширенное. Различаются они тем, что у Simple вся информация необходимая для функционирования действия известна в момент создания Action. Например, чтобы включить диммер на полную яркость необходимо активировать канал setDimmer и отправить туда значение 255. Значение известно заранее. Advanced Action создается на основе не полной информации о предстоящем действии, в случае с диммером, можно привести пример установки диммера в любое допустимое значение от 0 до 255, но какое значение надо будет записать не известно, причем не известен даже момент, в какой требуется получить это значение.
Для добавления действия на под-устройство можно использовать следующий метод:

```
// Пример функции для действия

function changeLight(in_data){
// Выводи сообщение о запуске метода
IR.Log("changeLight run");
// Выводим передаваемые параметры
// lamp: 1, set: 100 - сообщение выводимое в логе
IR.JSONLog(in_data);
// Передаем на устройство полученные параметры
device.Send(in_data); }

// Добавить действие для лампы

NewSubDevice.addAction({
ActionType: "Simple",  // Указываем вид действия
Type: "script_call",   // Указываем тип действия
Param: {lamp: 1, set: 100},   // Передаем параметры, для вызова функции
-необязательно
Space: this,  // Передаем пространство для вызова функции      -необязательно
Method: changeLight, // Передаем метод
Name: "Turn On"    // Указываем имя метода
})
```

Входным параметром метода является:
ActionType - указываем вид действия (**Simple** - простое или **Advanced** - расширенное).
Type - указываем тип действия.
Param - необязательно, передаем параметры, для вызова функции (метода).
Space - необязательно, передаем пространство для вызова функции.
Method - передаем метод (функцию).

Name - указываем имя метода.
Выходным параметром метода является: успешно или нет (**True**, **False**).
Пример действия типа отправки числа на канала:

```
NewSubDevice.addAction({

  ActionType: "Simple",  // Указываем вид действия
  Type: "send_number"   // Указываем тип действия
  Param: "0",   // Передаем параметры, для вызова функции
  Name: Turn Off,   // Указываем имя метода
  ChannelName: channelName, // Указываем имя канала
});
```

Входным параметром метода является:
ActionType - указываем вид действия (**Simple** - простое или **Advanced** - расширенное).
Type - указываем тип действия.
Param - передаем параметры, для вызова функции (метода).
ChannelName - указываем имя канала.
Name - указываем имя метода.
Выходным параметром метода является: успешно или нет (**True**, **False**).

- Удалить действие у под-устройства.

Для добавления действия на под-устройство можно использовать следующий метод:
`SubDevice.removeAction(in_ActionName)`
Входным параметром метода является: **in_ActionName** - Имя действия (**Name**).
Выходным параметром метода является: успешно или нет (**True**, **False**).

- Получение под-устройства.

Данный метод используется для получения данных об под-устройстве (обертку):
`module.getSubDevice(in_SubDeviceName)`
Входным параметром метода является: **in_SubDeviceName** - Имя под-устройства.
Выходным параметром метода является: объект под-устройства.

- Удаление под-устройства.

Для удаления под-устройств можно использовать следующий метод:
`module.removeSubDevice(in_SubDevice)`
Входным параметром метода является: **in_SubDevice** - Объект под-устройства.
Выходным параметром метода является: успешно или нет (**True**, **False**).

2.2. Создание Commands и Feedback.

Существует так же два способа создания **Commands** (Команд) и **Feedback** (Фидбеков):

- Используя функции редактора iRidium Studio.
- Через код.

Рассмотрим подробнее способ создания команд и фидбеков через код, но перед этим следует подключить модуль в дереве **PROJECT DEVICE PANEL**:

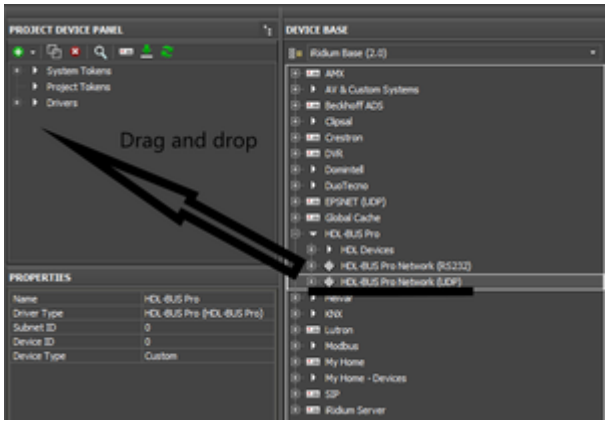


Рис. Подключение модуля

Для подключения достаточно перетащить и отпустить как показано на рисунке. Далее рассмотрим код:

```
// Получение адреса подсети

var subnetID = module.FromSubnetID;
// Получение номера устройства
var deviceID = module.FromDeviceID;
// Формируем имя подсети
var netWorkName = "HDL-BUS Pro Network (UDP)" + busID;
// Формируем имя устройства
var deviceName = "HDL-MC48IPDMX.231";
// Формируем имя канала
var statusOnStartName = 'deviceName + ":" + "statusOnStart" + moduleID';
// Разделитель, используемый в командах
var separator = 0x0;
// Таблица HDL кодов команд
var HDLCodes = {
   singleChannelLigtning: 0x31,
   singleChannelReadTarget: 0x33,
   singleSceneControl: 0x02,
   singleSequenceControl: 0x1A
};
// Формируем параметры канала
var Parametrs = [subnetID, deviceID, HDLCodes.singleChannelReadTarget,
separator, 0x0, 0x0, 0x0, 0x0, 0x1, 0x0, 0x1, 0xE8, 0x3, 0x0, 0x0]
// Добавляем канал StatusOnStart
device.AddChannel(statusOnStartName, Parametrs);
```

Входным параметром метода является:
statusOnStartName - Имя канала.
Parametrs - Параметры канала.
Выходным параметром метода является: успешно или нет (**True**, **False**).
Для данного канала не требуется создание Feedback, так как он осуществляет получение данных при старте модуля, в связи с этим я покажу пример создания Feedback для другого канала:

```
device.AddTag(channelName, Parametrs);
```

Входным параметром метода является:
channelName - Имя канала.

Parametrs - Параметры канала.
Выходным параметром метода является: успешно или нет (**True**, **False**).
ВАЖНО!!! Созданные команды и фидбеки через код **не показываются** в дереве PROJECT DEVICE PANEL.

2.3. Создание Widget через код.

Для создания виджета через код нам понадобиться уже созданный заранее пример виджета либо создать его через код. Для примера создадим виджет димера. Виджет ограничен размерами:

1. Ширина **640**
2. Высота от **300**

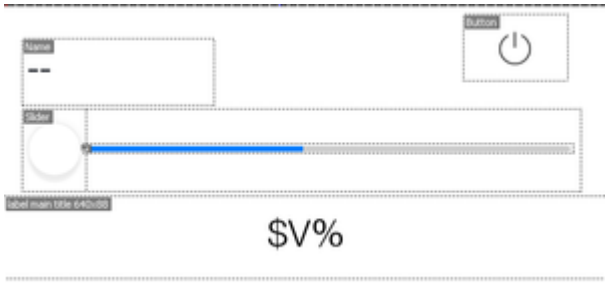


Рис. Пример виджета

Что бы указать программе, что данное всплывающее окно - не виджет, установите значение **False** в поле **Widget** в **Object Properties**.
`SubDevice.addWidget(in_Widget)`
Входным параметром метода является: **in_Widget** - объект всплывающее окно.
Выходным параметром метода является: успешно или нет (**True**, **False**).
К примеру у нас 3 лампочки и для того что бы не создавать для каждой лампочки отдельный виджет мы воспользуемся уже готовым примером нашего всплывающего окна и просто будем клонировать. Напишем пример создания виджета с помощью кода по уже существующему образцу.
Метод Clone позволяет клонировать уже существующее всплывающее окно.
`Widget.Clone (in_Name)`
Входным параметром метода является: **in_Name** - имя нового виджета.
Выходным параметром метода является: успешно или нет (**True**, **False**).

```
// Обращаемся к виджету который собираемся скопировать

var popup = IR.GetItem("Dimmer" + moduleID);
for (var i = 1; i<=3; i++)
{  // Создаем виджет методом клонирования
var widget = NewSubDevice.addWidget(popup.Clone("Dimmer"+ i + moduleID));
}
```

Для нашего димера добавим функцию которая позволяет двигать слайдер, подробнее её разбирать не будем:

```
// пользовательский слайдер для уровня

function UserSlider(Level, Slider)
{
```

```
   Property = "X";
   Len = "Width";
   // Функция вычисления позиции слайдера относительно уровня
   function Move(){
      Slider[Property] = Level.Value * (Level[Len] -50) / 100;
   }
   // Подписка на события
   IR.AddListener(IR.EVENT_ITEM_PRESS, Level, Move); // нажатие на уровень
   IR.AddListener(IR.EVENT_MOUSE_MOVE, Level, Move); // движение мыши по
уровню
   IR.AddListener(IR.EVENT_TOUCH_MOVE, Level, Move); // движение пальца по
уровню
   IR.SetInterval(600, Move);  // автообновление через 600 мс
};
```

ВАЖНО!!! Имена новых виджетов не должны повторяться (не должны совпадать), а так же **обязательно** наличие **moduleID**.

2.4. Создание связи между графическим элементом виджета и каналом драйвера.

Связь между графическим элементов и каналом драйвера образуется при помощи связи канала обратной связи драйвера (Feedback) и свойство графического элемента на созданном виджете. Напишем пример создания связей между графическими элементами созданного виджета и каналов драйверов с помощью кода.
`IR.AddRelation(`**`in_Feedback, in_Graphic`**`);`
Входным параметром метода является:
in_Feedback - полный путь к каналу обратной связи в строковом типе (пример: "Drivers.HDL-BUS Pro Network (UDP)" + busID + ".HDL-MC48IPDMX.231:channel" + moduleID).
in_Graphic - полный путь к свойству графического элемента в строковом типе (пример: "UI.Dimmer" + moduleID + ".level.Value").
Выходным параметром метода является: успешно или нет (**True**, **False**).
Рассмотрим полный код модуля которая создает 3 виджета для 3 разных под-устройств и разными закрепленными каналами, для этого Вам понадобиться ранее созданное всплывающее окно:

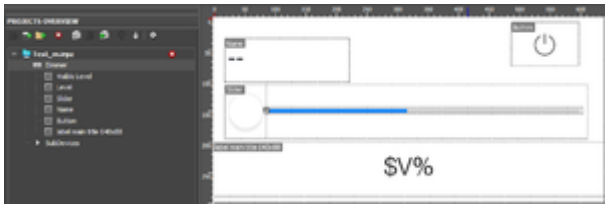


Рис. Пример результата работы в iRidium Studio

и код модуля представленный ниже:

```
// Получаем уникальный идентификатор модуля и шины при запуске

IR.AddListener(IR.EVENT_MODULE_START, 0, function(moduleID, busID){
  // Получение копии модуля
  // moduleID - уникальный идентификатор модуля
  // Присваивание переменной module
```

```
  var module = B.getModule(moduleID);
  // Формируем имя подсети и присваиваем переменной netWorkName
  var netWorkName = "HDL-BUS Pro Network (UDP)" + busID;
  // Берем существующее устройство и присваиваем переменной device
  var device = IR.GetDevice(netWorkName);
  // Получение адреса подсети
  var subnetID = module.FromSubnetID;
  // Получение номера устройства
  var deviceID = module.FromDeviceID;
  // Формируем имя устройства
  var deviceName = "HDL-MC48IPDMX.231";
  // Формируем имя канала
  var statusOnStartName = 'deviceName + ":" + "statusOnStart" + moduleID';
  // Разделитель, используемый в командах
  var separator = 0x0;
  // Таблица HDL кодов команд
  var HDLCodes = {
     singleChannelLigtning: 0x31,
     singleChannelReadTarget: 0x33,
     singleSceneControl: 0x02,
     singleSequenceControl: 0x1A
  };
  // Неполное имя каналов для простоты использования
  var drivers = "Drivers." + netWorkName + "." + deviceName + ":" +
"channel";
  // Формируем параметры канала
  var Parametrs = [subnetID, deviceID, HDLCodes.singleChannelReadTarget,
separator, 0x0, 0x0, 0x0, 0x0, 0x1, 0x0, 0x1, 0xE8, 0x3, 0x0, 0x0]
  // Если устройство подключено то
  if(device) {
     // Добавляем канал StatusOnStart
     device.AddChannel(statusOnStartName, Parametrs);
     // Очистка кеша
     B.clearEmulatorCahce(true);
     // Цикл для создания виджетов
     for (var i = 0; i<3; i++){
        // Обращаемся к виджету который собираемся скопировать
        var popup = IR.GetItem("Dimmer" + moduleID);
        // Присваиваем оригинальное имя
        var name = "Dimmer" + i;
        // Создаем новое под-устройство с параметрами:
        // Device - объект драйвера устройства iRidium
        // Имя создаваемого под-устройства, через которое будем к нему
обращаться и это же имя увидит пользователь
        var NewSubDevice = module.addSubDevice({
                Device: device,
                DeviceName: name
        });
        // Создание виджета
        var widget = popup.Clone("Dimmer" + i + moduleID)
        // Добавляем виджет на экран
```

```
        NewSubDevice.addWidget(widget);
        // Объявления переменных необходимых для создания виджета
        var channelName = deviceName + ":" + "channel" + i + moduleID;
        var Dimmer = IR.GetItem("Dimmer" + i + moduleID);
        var Dimmer_level = Dimmer.GetItem("Level");
        var Dimmer_button = Dimmer.GetItem("Button");
        var uiDimmer = "UI.Dimmer" + i + moduleID;
        // Добавление каналов
        device.AddChannel(channelName, [subnetID, deviceID,
HDLCodes.singleChannelLigtning, separator, parseInt(i+1, 16), 0x0, 0x0, 0x0,
0x01, 0x0, 0x0]);
        device.AddTag(channelName, [subnetID, deviceID,
HDLCodes.singleChannelLigtning, separator, parseInt(i+1, 16), 0x0, 0x0]);
        // Вызов функции привязки слайдера
        var UserSlider_1 = new UserSlider(Dimmer_level,
Dimmer.GetItem("Slider"));
        // Отправка команды с уровня на HDL канал диммера
        IR.AddListener(IR.EVENT_ITEM_RELEASE, Dimmer_level, function ()
        {
           var data = this;
           device.Set(data.channel, data.level.Value);
        },{level: Dimmer_level, channel: channelName});
        // Кнопка для включения и отключения (0 или 100)
        IR.AddListener(IR.EVENT_ITEM_PRESS, Dimmer_button, function()
        {
           var data = this;
           device.Set (data.channel, data.button.Value * 100);
        }, {button: Dimmer_button, channel: channelName});
        // Привязка графических элементов к фидбекам
        Dimmer.GetItem("Name").Text = "Dimmer" + i;
        IR.AddRelation(drivers + i + moduleID, uiDimmer + ".Visible
Level.Value");
        IR.AddRelation(drivers + i + moduleID, uiDimmer + ".Level.Value");
        IR.AddRelation(drivers + i + moduleID, uiDimmer + ".label main title
640x88.Value");
        IR.AddRelation(drivers + i + moduleID, uiDimmer + ".Button.Value");
     }
  };
  // Пользовательский слайдер для уровня
  function UserSlider(Level, Slider)
  {
     Property = "X";
     Len = "Width";
     // Функция вычисления позиции слайдера относительно уровня
     function Move(){
        Slider[Property] = Level.Value * (Level[Len] -50) / 100;
     }
     // Подписка на события
     IR.AddListener(IR.EVENT_ITEM_PRESS, Level, Move); // нажатие на уровень
     IR.AddListener(IR.EVENT_MOUSE_MOVE, Level, Move); // движение мыши по
уровню
```

```
    IR.AddListener(IR.EVENT_TOUCH_MOVE, Level, Move); // движение пальца по
уровню
    IR.SetInterval(600, Move);  // автообновление через 600 мс
  };
});
```

Результатом данного модуля у Вас должны появиться 3 димера с разными каналами как показано на рисунке:

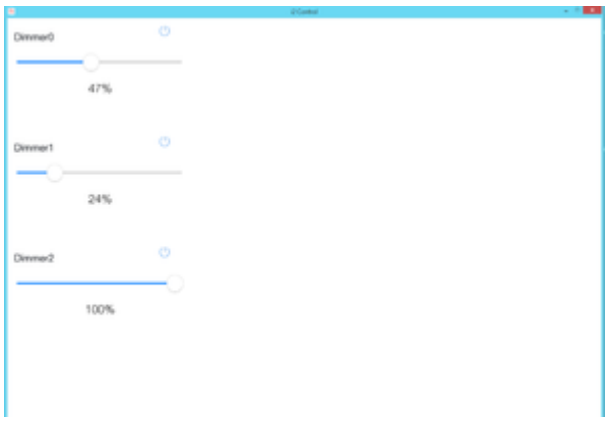


Рис. Результат

Скачать пример модуля:[1]

---

3. Адаптация модуля для мобильных устройств (New).

Если Вы хотите создать модуля для нескольких типов устройств (планшетов и смартфонов), то Вам понадобиться метод:
`B.GetResolutionType()`
Входных параметров у данного метода нет.
Выходным параметром метода является:

- **B.RESOLUTION_TYPE.TABLET** (планшетный тип)
- **B.RESOLUTION_TYPE.PHONE** (телефонный тип)

Для разных типов устройств используется разное расширение(для планшеты 2048x1536, а для телефона 640x1136), в связи с этим Вам понадобиться два типа виджетов с разными размерами для лучшего отображения. Рассмотрим пример загрузки виджетов разных размеров:

```
// Определение типа устройства (телефон, планшет)

var my_resolution = B.GetResolutionType();
// Условие выбора по типу устройства
if (my_resolution == B.RESOLUTION_TYPE.PHONE)
{
   // Подключение телефонных виджетов
   var Error = "ErrorChanel_Phone" + moduleID;
}
else
{
```

```
   // Подключение планшетных виджетов
   var Error = "ErrorChanel_Tablet" + moduleID;
};
```

В данном примере, переменной **my_resolution** мы присваиваем тип нашего устройства, далее проводим выбор по типу и в соответствии типа присваиваем переменной **Error** виджет нашего типа (телефон, планшет).
Так же для тестирования того или иного типа можно использовать следующее:

```
// Присваивание переменной my_resolution телефонного тип

var my_resolution = B.RESOLUTION_TYPE.PHONE;
// Условие выбора по типу устройства
if (my_resolution == B.RESOLUTION_TYPE.PHONE)
{
   // Подключение телефонных виджетов
   var Error = "ErrorChanel_Phone" + moduleID;
}
```

Данным способом Вы можете протестировать как у Вас будут работать виджеты для двух типов устройств.
Так же Вы можете не создавать по 2 типа виджета, а изменять их размеры и параметры элементов (размеры и координаты) через код.

```
// Определение типа устройства (телефон, планшет)

var my_resolution = B.GetResolutionType();
// Условие выбора по типу устройства
if (my_resolution == B.RESOLUTION_TYPE.PHONE)
{
   // Изменение размера виджета под телефон
  IR.GetItem (""ErrorChanel" + moduleID).Width =   IR.GetPage(2).Width * 50 /
100;
  IR.GetItem (""ErrorChanel" + moduleID).Height = IR.GetPage(2).Height  *
0.5;
}
else
{
   // Изменение размера виджета под планшет
   IR.GetItem (""ErrorChanel" + moduleID).Width =   IR.GetPage(2).Width * 1;
  IR.GetItem (""ErrorChanel" + moduleID).Height = IR.GetPage(2).Height  *
0.75;
};
```

ВАЖНО!!! Следует помнить что в случае изменения размера виджета, координаты и размеры элементов виджетов не изменяются, данный способ актуален если у Вас на виджете 1-2 элемента.

---

4. Полный пример программы с адаптацией к телефону.

Рассмотрим полный пример программы с комментариями.
Ваше дерево проекта должно выглядеть следующим образом:

Рис. Дерево Вашего проекта

Пример кода полной программы:

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

  // Получение копии модуля
  // moduleID - уникальный идентификатор модуля
  // Присваивание переменной module
  var module = B.getModule(moduleID);
  // Формируем имя подсети и присваиваем переменной netWorkName
  var netWorkName = "HDL-BUS Pro Network (UDP)" + busID;
  // Берем существующее устройство и присваиваем переменной device
  var device = IR.GetDevice(netWorkName);
  // Получение адреса подсети
  var subnetID = module.FromSubnetID;
  // Получение номера устройства
  var deviceID = module.FromDeviceID;
  // Формируем имя устройства
  var deviceName = "HDL-MC48IPDMX.231";
  // Формируем имя канала
  var statusOnStartName = 'deviceName + ":" + "statusOnStart" + moduleID';
  // Разделитель, используемый в командах
  var separator = 0x0;
  // Таблица HDL кодов команд
  var HDLCodes = {
    singleChannelLigtning: 0x31,
    singleChannelReadTarget: 0x33,
    singleSceneControl: 0x02,
    singleSequenceControl: 0x1A
```

```
  };
  // Обращение к существующему под-устройству SubDevice1
  var settingPopup = module.getSubDevice("SubDevice1");
  // Определение типа устройства (телефон, планшет)
  var myResolution = B.GetResolutionType();
  // Условие выбора по типу устройства
  if (myResolution == B.RESOLUTION_TYPE.PHONE) {
     // Подключение телефонных всплывающих окон
     var error = "ErrorChanel_phone" + moduleID;
     var addDeviceName = "AddDevice_phone" + moduleID;
     var addDimmerName = "AddDimmerChannel_phone" + moduleID;
     var listFreeChannel = IR.GetItem("ListFreeChannel_phone" + moduleID);
     var chooseDevice = IR.GetItem("ChooseDevice_phone" + moduleID);
  }
  else {
     // Подключение планшетных всплывающих окон
     var error = "ErrorChanel" + moduleID;
     var addDeviceName = "AddDevice" + moduleID;
     var addDimmerName = "AddDimmerChannel" + moduleID;
     var listFreeChannel = IR.GetItem("ListFreeChannel" + moduleID);
     var chooseDevice = IR.GetItem("ChooseDevice" + moduleID);
  };
  // Обращаемся к виджету Setup
  var setup = IR.GetItem("Setup" + moduleID);
  // Добавляем виджет Setup в под-устройство
        var widget = settingPopup.addWidget(setup);
  // Обращаемся к всплывающему окну Добавления под-устройств
  var addDevice = IR.GetItem(addDeviceName);
  // Обращаемся к всплывающему окну Добавления диммера
  var addDimmer = IR.GetItem(addDimmerName);
  // Обращаемся к элементу List на всплывающем окне свободных каналов
  var list = listFreeChannel.GetItem("listAddchannel");
  // Выставляем шаблон элементов списка листа свободных каналов
  list.Template = 'Channel' + moduleID;
  // Обращаемся к всплывающему окну Ошибка
  var messageError = IR.GetItem(error);
  // Обращаемся к элементу List на всплывающем коне Добавления под-устройств
  var listDevice = addDevice.GetItem("listAddDevice");
  // Выставляем шаблон элементов списка листа Добавления под-устройств
  listDevice.Template = 'Device_template' + moduleID;
  // Объявление необходимых переменных
  var idSub;          // Переменная для обращения к под-устройства
  var tName;           // Переменная для сравнения запретных символов
  var duplicateName;  // Переменная для сравнения совпадающих имен
  // Объявление массива запретных символов
  var simvoli = ["\\", "@", "*", "/", "|", '&', '%', '#', '"', '^', ':', ';',
'$', '<', '<<', '>', '>>', '!', '?', '{', '}', '[', ']', '(', ')', '=', '\,
'.', ',', '``];

  // Добавляем канал StatusOnStart
  device.AddChannel(deviceName + ":" + "statusOnStart" + moduleID, [subnetID,
```

```
deviceID, HDLCodes.singleChannelReadTarget, separator, 0x0, 0x0, 0x0, 0x0,
0x1, 0x0, 0x1, 0xE8, 0x3, 0x0, 0x0]);
  // Если устройство подключено то
  if(device) {
     var cache = settingPopup.getData();
     //B.clearEmulatorCahce(true);      // Чистка кеша
     // Загрузка кеша если он не равен нулю, а иначе создание переменных
     if (cache == null) var cache = [];
     if (cache[1] == null) {
        // Заполнение массива свободных каналов если кеш пуст
        var m = [];
        for (var j = 0; j<48; j++)
        m[j] = j+1;
     }
     else
        var m = cache[1];
     // Заполнение листа свободных каналов
     fillListFreeChannel();
     if (cache[2] == null) {
        var id = 0;
     }
     else
        var id = cache[2];
     if (cache[0] == null)
        var md = []
     else
        var md = cache[0];
     for (var i = 0; i < md.length; i++) {
        // Обращаемся к существующему под-устройству с именем из кеша
        var create = module.getSubDevice(md[i].data);
        // Присваиваем существующее под-устройство массиву
        md[i].subDevice = create;
        // Вызов функции создания виджетов диммера
        new CreateDimmer (md[i]);
        // заполнение листа под-устройств
        fillListSubDevice();
     };
     // Чистка текста при нажатии кнопки Cancel  и возвращение элементов в
массив на попапе AddDimmer
     IR.AddListener(IR.EVENT_ITEM_RELEASE, addDimmer.GetItem("cancel1"),
function() {
        // Очистка имени
        addDimmer.GetItem("NameDevice").Text = ;
        // Если канал выбран то
        if (addDimmer.GetItem("Channelnumber").Text != ) {
           // Возвращаем канала в список свободных каналов
           m.push (addDimmer.GetItem("Channelnumber").Text);
           // Очищаем текст каналов
           addDimmer.GetItem("Channelnumber").Text = ;
        };
        // Заполнение листа свободных каналов
```

```
        fillListFreeChannel();
     });
     //Создание Dimmer виджета
     IR.AddListener(IR.EVENT_ITEM_RELEASE, addDimmer.GetItem("ok1"),
function() {
        name = addDimmer.GetItem("NameDevice").Text;
        duplicateName=0;
        tName = 0;
        // Совпадение с запретными символами
        // Цикл для проверки массива запрещенных символов
        for (var k=0; k<simvoli.length; k++) {
           // Если символ из имени совпадает с символом из массива запретных
символов то выводим ошибку
           if (name.indexOf(simvoli[k].toString()) != -1) {
              tName = 1;
              break;
           };
        };
        if (tName != 1) {
           // Проверка на повторяющиеся имена
           for (var j=0; j<md.length; j++)
           // Если имя совпадает с именами из массива то присваиваем
переменной duplicateName = 1
           if (name == md[j].data) duplicateName=1;
           // Если переменная duplicateName = 0 то тогда
           if (duplicateName == 0) {
              // Проверка на ненулевые значения каналов
              if (addDimmer.GetItem("Channelnumber").Text == ) {
                 IR.GetItem(error).GetItem("Button").Text = 'Не правильное
значение!';
                 // Вывод всплывающего окна ошибки
                 IR.ShowPopup(error);
              }
              else {
                 // Очистка имени при создании
                 addDimmer.GetItem("NameDevice").Text = ;
                 i = parseInt(addDimmer.GetItem("Channelnumber").Text);
                 // Очистка EditBox (канала) при создании
                 addDimmer.GetItem("Channelnumber").Text = ;
                 // Создание под-устройства
                 idSub = module.addSubDevice({
                              Device: device, // Передаем устройство, как
основной источник информации
                              DeviceName: name  // Имя устройства, которое
видит пользователь
                 });
                 // Добавление данных в массив
                 md.push ({
                    index: id,
                    data: name,
                    subDevice: idSub,
```

```
                  W: i
               });
               // Вызов функции создания виджета
               new CreateDimmer (md[md.length-1]);
               // Добавление уникального кода
               id++;
               // Сохранение в кеш
               cache[0] = md;
               cache[1] = m;
               cache[2] = id;
               // Отправка нового кеша на под-устройство
               settingPopup.setData(cache);
               // Заполнение листа под-устройств
               fillListSubDevice();
               //  Открытие всплывающего окна Добавления под-устройств
               IR.ShowPopup(addDeviceName);
               // закрытие всплывающего окна Добавления Диммера
               IR.HidePopup(addDimmerName);
            };
         }
         else {
            IR.GetItem(error).GetItem("Button").Text = 'Совпадение имени!';
            // Открываем всплывающее окно ошибки
            IR.ShowPopup(error);
         };
      }
      else {
         IR.GetItem(error).GetItem("Button").Text = 'Запретные символы!';
         // Открываем всплывающее окно ошибки
         IR.ShowPopup(error);
      };
   });
};
// Функция удаления под-устройств
IR.AddListener (IR.EVENT_LIST_ITEM_CHANGE, listDevice, function (Item,
Subitem, TypeEvent, object) {
   // Если нажали то
   if (TypeEvent == 12) {
      // Если нажали на кнопку удаления то
      if (object.Name == 'button close 1') {
         // Удаление виджета
         var removeWidget = md[Item].subDevice.removeWidget(md[Item].type +
md[Item].index+ moduleID);
         // Удаление под-устройства
         removeSub = module.removeSubDevice(md[Item].subDevice);
         // Возвращение свободного канала
         m.push (md[Item].W.toString());
         // Удаление под-устройства из массива под-устройств
         md.splice(Item,1);
         // Сохранение в кеш
         cache[0] = md;
```

```
            cache[1] = m;
            settingPopup.setData(cache);    // Отправка измененного кеша на
под-устройство
            fillListFreeChannel();   // Вызываем функцию заполнения листа
свободных каналов
            fillListSubDevice();     // Вызываем функцию заполнения под-
устройств
        };
    };
  });

  // Создание затемнения на заднем фоне для всплывающих окон
  messageError.GetState(0).FillColor = 0x00000066;
  addDimmer.GetState(0).FillColor = 0x00000066;
  listFreeChannel.GetState(0).FillColor = 0x00000066;
  addDevice.GetState(0).FillColor = 0x00000066;
  // Изменение размеров элементов всплывающих окон через код
  addDevice.GetItem("listAddDevice").Height = 650;
  listFreeChannel.GetItem("listAddchannel").Height = 550;
  addDimmer.GetItem("Channelnumber").Width = 200;
  // Создание виджета
  function CreateDimmer (md) {
     // Обращаемся к всплывающему окну который хотим клонировать
     var popup = IR.GetItem("Dimmer" + moduleID);
     // Присваиваем переменной имя канала
     var channelName = deviceName + ":" + "channel" + md.W + moduleID;
     widget = md.subDevice.addWidget(popup.Clone("Dimmer" + md.index +
moduleID));
     // Объявления переменных необходимых для создания виджета
     var dimmer = IR.GetItem("Dimmer" + md.index + moduleID);
     var dimmerLevel = dimmer.GetItem("Level");
     var dimmerButton = dimmer.GetItem("Button");
     // Добавление каналов выбранные пользователем
     device.AddChannel(channelName, [subnetID, deviceID,
HDLCodes.singleChannelLigtning, separator, parseInt(md.W, 16), 0x0, 0x0, 0x0,
0x01, 0x0, 0x0]);
     device.AddTag(channelName, [subnetID, deviceID,
HDLCodes.singleChannelLigtning, separator, parseInt(md.W, 16), 0x0, 0x0]);

     // Вызов функции привязки слайдера
     var userSlider5 = new UserSlider(dimmerLevel, dimmer.GetItem("Slider"));
     // Отправка команды с уровня на HDL канал диммера
     IR.AddListener (IR.EVENT_ITEM_RELEASE, dimmerLevel, function () {
        device.Set(channelName, this.Value);
     }, dimmerLevel);
     // Кнопка для включения и отключения (0 или 100)
     IR.AddListener (IR.EVENT_ITEM_PRESS, dimmerButton, function() {
        device.Set (channelName, dimmerButton.Value * 100);
     });
     // Привязка графических элементов к фидбекам
     dimmer.GetItem("Name").Text = md.data;
```

```
     IR.AddRelation("Drivers." + netWorkName + "." + deviceName + ":" +
"channel" + md.W + moduleID, "UI.Dimmer" + md.index + moduleID + ".Visible
Level.Value");
     IR.AddRelation("Drivers." + netWorkName + "." + deviceName + ":" +
"channel" + md.W + moduleID, "UI.Dimmer" + md.index + moduleID +
".Level.Value");
     IR.AddRelation("Drivers." + netWorkName + "." + deviceName + ":" +
"channel" + md.W + moduleID, "UI.Dimmer" + md.index + moduleID + ".label main
title 640x88.Value");
     IR.AddRelation("Drivers." + netWorkName + "." + deviceName + ":" +
"channel" + md.W + moduleID, "UI.Dimmer" + md.index + moduleID +
".Button.Value");
     // Добавить действие для лампы
     md.subDevice.addAction ({
        ActionType: "Simple",  // Указываем вид действия
        Type: "script_call",   // Указываем тип действия
        Param: {lamp: 1, set: 100},   // Передаем параметры, для вызова
функции      -необязательно
        Space: this,  // Передаем пространство для вызова функции      -
необязательно
        Method: changeLight, // Передаем метод
        Name: "Turn On"    // Указываем имя метода
     });
  };
  // Функция для действия
  function changeLight (in_data) {
     // Выводи сообщение о запуске метода
     IR.Log("changeLight run");
     // Выводим передаваемые параметры
     // lamp: 1, set: 100 - сообщение выводимое в логе
     IR.JSONLog(in_data);
     // Передаем на устройство полученные параметры
     device.Send(in_data);
  };
  // Функция заполнения листа под-устройств
  function fillListSubDevice () {
     listDevice.Clear();
     for (var i=0; i<md.length; i++) {
        listDevice.CreateItem(i, 0, {Text: md[i].data});
     };
  };
  // Функция заполнение листа свободных каналов
  function fillListFreeChannel () {
     m.sort(function(a,b){return a-b;}); // Функция сортировки элементов по
возрастанию
     list.Clear();   // Очистка листа
     for (var i=0; i<m.length; i++) {
       list.CreateItem(i, 1, {Text: m[i].toString()});  // Заполнение листа
     };
  };
  // Вызываем функцию для начального построения списка свободных каналов
```

```
  fillListFreeChannel();
  // Удаление выбранного канала со списка свободных каналов
  IR.AddListener(IR.EVENT_LIST_ITEM_CHANGE, list, function(Item, Subitem,
TypeEvent, object) {
     // Если нажали то
     if (TypeEvent == 12) {
        // Если ранее был выбран канал то возвращаем его
        if (addDimmer.GetItem("Channelnumber").Text != ) {
           m.push (addDimmer.GetItem("Channelnumber").Text);
        };
        // Присваиваем EditBox текст (канал) который выбрали
        addDimmer.GetItem("Channelnumber").Text = object.Text;
        m.splice(Item,1);   //  Удаление выбранного канала из списка
свободных каналов
        IR.HidePopup('ListFreeChannel' + moduleID);  // Закрытие всплывающего
окна
        fillListFreeChannel();    // Перезаполняем список свободных каналов
     };
  });
  // Открытие окна AddDevice  при нажатии кнопки на виджете Setup
  IR.AddListener(IR.EVENT_ITEM_RELEASE, setup.GetItem("button next 120x120"),
function () {
     IR.ShowPopup(addDeviceName);
  });
});
// Пользовательский слайдер для уровня
function UserSlider (in_Level, in_Slider) {
  Property = "X";
  Len = "Width";
  // Функция вычисления позиции слайдера относительно уровня
  function Move(){
     in_Slider[Property] = in_Level.Value * (in_Level[Len] -50) / 100;
  };
  // Подписка на события
  IR.AddListener(IR.EVENT_ITEM_PRESS, in_Level, Move); // нажатие на уровень
  IR.AddListener(IR.EVENT_MOUSE_MOVE, in_Level, Move); // движение мыши по
уровню
  IR.AddListener(IR.EVENT_TOUCH_MOVE, in_Level, Move); // движение пальца по
уровню
  IR.SetInterval(600, Move);
};
```

Полный рабочий пример Вы можете скачать через данную ссылку: [[2]]