## Подключаемся к Intel Edison через Android с Bluetooth LE (BLE) [tutorial]Вступление

Технология Bluetooth с низким энергопотреблением (англоязычный вариант: Bluetooth LE - low energy; для краткости - BLE), нашла применение как в коммерческих, так и в сугубо любительских продуктах благодаря и низкой стоимости, и столь же низкому аппетиту к ресурсам аккумуляторов. Эти факторы делают BLE отличным выбором для связи смартфона или планшета под ОС Android с проектами на базе плат Intel для разработчиков - Edison или Galileo.

Задача предложенной ниже статьи - показать, как нужно писать код для соединения “железа” посредством протокола BLE на примере платформы Intel Edison и Android-устройства с Bluetooth версии 4.0. При этом используется исключительно бесплатное ПО и недорогие “железные” компоненты, уже готовое к работе.

## Что такое BLE?

Знакомимся: BLE, он же Bluetooth LE; полное “имя” - Bluetooth Low Energy (другое название - Bluetooth Smart). Технология беспроводной персональной сети передачи данных, разработанная и распространяемая организацией Bluetooth Special Interest Group (Bluetooth SIG). Предназначение Bluetooth с низким энергопотреблением - применение в сферах здравоохранения и домашних развлечений, в задачах спорта и физкультуры, в автомобилях.

Bluetooth LE впервые был представлен в 2006 году компанией Nokia под кодовым названием *Wibree*. А в стандарте Bluetooth он появился в 2010 году в версии Bluetooth 4.0.

BLE позволяет устройствам довольствоваться значительно меньшим количеством энергии по сравнению со стандартными соединениями Bluetooth. При этом он предоставляет бОльшую часть обычной функциональности на расстоянии, приблизительно вдвое меньшем по сравнению с обычным - примерно 15 метров. Устройства, работающие по протоколу Bluetooth LE, способны годами работать без необходимости менять или подзаряжать батарею. Маячки, наподобие устройств фирмы Estimote, уже способны проработать от батарейки до 3 (трёх!) лет: [www.estimote.com](http://www.estimote.com/).

## Железо

Главным героем поста будет Intel Edison, хотя многие вещи применимы и для Intel Galileo. В нашем IoT-проекта мы будем использовать физические сенсоры и органы управления системы Grove производства фирмы Seeed Studio. А именно:

* плата Intel Galileo с Arduino;
* начальный комплект системы Seeed Grove (Starter Kit Plus) для Intel IoT Edition для Galileo (второго поколения);
* Seeed Grove BLE;
* Android-девайс с версией ОС 4.3 или выше (я использовал планшет Lenovo TAB S8-50);
* компьютер с Windows 7 или 8 (я использовал Dell XPS 12).

Несколько нюансов по железу:

1. Хотя комплект **Grove** начального уровня (**Starter Kit)** предназначен для платформы Intel Galileo, он вполне успешно работает и с Edison. Можно, конечно, приобрести компоненты Grove по отдельности, однако использование готового комплекта - более экономичный путь.
2. В проекте использовался мой планшет **Lenovo** на **Android**, однако должно подойти и любое другое устройство с поддержкой Bluetooth 4.0, работающее под управлением этой ОС версии не ниже 4.3.
3. Написание кода для Intel Edison, Android-устройства (и написания этой статьи) происходило на ноутбуке Dell XPS12. Возможна и разработка на системах Mac или Linux.

## Софт

Я использовал несколько бесплатных программ – для воспроизведения данного примера вам тоже придётся их поставить:

* Arduino IDE для Intel Galileo: <https://communities.intel.com/docs/DOC-22226>
* Android Studio: <https://developer.android.com/sdk/installing/studio.html>
* Android SDK: <https://developer.android.com/sdk/index.html?hl=i>

Весь перечисленный выше софт доступен для Windows, Mac и Linux, но в статье я буду отталкиваться от примера “Винды”.

## Знакомимся поближе с Intel Edison

Intel Edison - первое предложение в серии недорогих многоцелевых компьютерных платформ. Эта платформа создана для простого и быстрого создания прототипов проектов IoT, одновременно обеспечивая создание производственных продуктов для коммерциализации проекта.

“Сердце” платы Intel Edison - 22-нм “система-на-чипе” Intel, включающая в себя двухядерный процессор Atom, работающий на частоте 500 МГц. Крохотный форм-фактор этой SoC включает в себя 40-пиновые интерфейсы GPIO, 1 Гб ОЗУ класса LPDDR3, 4 Гб на накопителе eMMC и двухфункциональный модуль Wi-Fi + Bluetooth.

Заправляет “железом” платформы Edison полновесное ядро Linux. А чтобы выжать все соки, заложенные в Edison, потребуется написание низкоуровневого кода под Linux.

Но Linux на Edison содержит ещё и возможность выполнения Linux-программ на мощностях Arduino. Проще говоря, это означает возможность написания привычных Arduino-скетчей и их запуска на Galileo. Именно эти мы и займёмся.

Подробнее о системе Edison рассказывается на сайте Intel: <http://www.intel.com/content/www/us/en/do-it-yourself/edison.html>

### Плата Arduino

Плата Arduino для Intel Galileo предназначен для двух целей. Во-первых, это более крупная платформа для прототипирования - с облегчённым доступом к интерфейсам ввода-вывода. А во-вторых, перед нами вычислительная платформа, полностью совместимая с Arduino. На практике это значит, что мы можем использовать на Intel Edison (или на Intel Galileo) стандартные платы расширения Arduino. На Рисунке 1 показан модуль Edison, установленный на плате Arduino.



***Фото 1.*** *Intel Edison установлен в плату Arduino*

### Комплект разработчика Grove Starter Kit Plus

Полное название этого комплекта -”Grove Starter Kit Plus - Intel IoT Edition for Intel Galileo Gen 2 Developer Kit”. Или, по-русски, “Комплект разработчика Grove ‘Начального с плюсом’ уровня; издание Intel IoT для платформы Intel Galileo 2 поколения”. Изначально набор предназначался для платы Intel Galileo 2 поколения. К счастью, в плате Arduino предусмотрена полная совместимость и с Intel Edison.

Комплект (он изображён на Рисунке 2) предназначен для упрощения макетирования и работы с датчиками, механизмами и модулями-”шилдами”. В комплекте присутствует совместимый с Arduino модуль на стандартных 4-пиновых интерфейсах. Интерфейсы питаются от портов ввода-вывода посредством включённых в комплект кабелей. Подключение кабелей к датчикам продумано весьма удобно: можно заниматься самими проектами, а не разбираться во всех этих мелких проводках, постоянно потягивать/ослабевать резисторы и беспокоиться о полярности подключения.

Больше информации о комплекте (и его приобретении) находится по этой ссылке: <http://www.seeedstudio.com/depot/Grove-starter-kit-plus-Intel-IoT-Edition-for-Intel-Galileo-Gen-2-p-1978.html>

Создатель набора Grove, компания Seeed Studios, предлагает коллекцию полезных онлайн-ресурсов.

Я в особенности рекомендую изучить или скачать материалы серии “Sketchbook для начинающих” (англ.): [http://Github.com/Seeed-Studio/Sketchbook\_Starter\_Kit\_V2.0](http://github.com/Seeed-Studio/Sketchbook_Starter_Kit_V2.0)

Всячески достойна помещения в Избранное и Wiki-страница Grove: <http://www.seeedstudio.com/wiki/index.php?title=Main_Page#Grove>



***Рисунок 2.*** *Комплект разработчика “Grove Starter Kit Plus - Intel IoT Edition for Intel Galileo Gen 2”*

### Grove-модуль Bluetooth LE (V1)

Мы будем использовать модуль Grove Bluetooth Low Energy v1, который не включён в комплект начального уровня, зато “пин-в-пин” совместим с модулем Grove и соединительными проводами. Также это сравнительно недорогой модуль BLE - на момент написания этой статьи я так и не нашёл ничего дешевле.

Основа “шилда” Grove BLE v1 - стандартный для устройств этого класса чип CC2540 от фирмы Texas Instruments. Этот же микрочип используется и в некоторых других продуктах. Поэтому, если под рукой есть другой BLE-модуль на базе TI CC2540, образец кода можно адаптировать и под него, притом с минимальными усилиями.

Больше информации про модуль Grove BLE v1 на сайте производителя: <http://www.seeedstudio.com/wiki/index.php?title=Grove_BLE_v1&uselang=en>

Отмечу, что на плате Intel Edison уже присутствует встроенный модуль беспроводной связи с поддержкой Wi-Fi и Bluetooth 4.0/BLE. Однако Grove-модуль BLE в разы упрощает как аппаратную, так и программную настройку. Использование модуля Grove (Рисунок 3) также означает, что проект легко адаптируется и для платформы Intel Galileo.



***Рисунок 3.*** *модуль Grove BLE V1*

### Отладка на Android-девайсах

Поддержка BLE в Android появилась с версии 4.3 (API level 18). Соответственно, для работы с BLE версия Android на девайсе должна быть не ниже 4.3.

Более подробно про Android BLE можно узнать тут: <https://developer.android.com/guide/topics/connectivity/bluetooth-le.html>

Если в освоении разработки под Android вы делаете первые шаги, то на планшете/телефоне вначале нужно активировать функции разработчика - они позволяют запускать и отлаживать код. Активация режима разработчика происходит так: открываем Настройки, пролистываем меню до самого низа, находим пункт “Об устройстве” и щёлкаем по номеру версии сборки 7 (семь) раз.

Теперь включение режима разработчика (Developer Options) отображается в нижней части “Настроек”; не забудьте отметить галочкой пункт ”USB debugging” (“Отладка USB”).

О режиме разработчика можно узнать здесь: <http://developer.android.com/tools/device.html>

## Установка софта. Готовимся кодить!

Для развёртывания скетчей на Intel Edison или Galileo нам понадобится скачать специально подготовленную версию Arduino IDE. На момент написания статьи актуальная версия - 1.5.3, а источник находится здесь:

<https://communities.intel.com/docs/DOC-22226>

### Драйверы для Intel Edison

По ссылке выше расположены и драйверы для Intel Edison: их тоже нужно скачать и установить. Расположены они в нижней части страницы, в категории “ПО и драйверы” - “Driver Software”. Текущая (на момент создания статьи) версия - 1.0.0.

Если возникнут дополнительные вопросы, весьма рекомендую превосходно написанную на английском языке инструкцию - “Getting Started Guide”:

<https://communities.intel.com/docs/DOC-23147>

### Android Studio

Android Studio – новая Java IDE для разработки на Android. Этот пакет написан на основе IntelliJ IDEA (<https://www.jetbrains.com/idea/>). На данный момент пакет имеет статус бета-версии, но работает он вполне стабильно, да и функциональность обеспечена полностью. Если вы уверенно работаете над Andorid-разработками в среде Eclipse для Java (или IntellliJ IDEA), то и при работе в Android Studio не должно возникнуть особых сложностей или проблем - вполне достаточно просмотреть ключевые моменты в обучающих материалах.

Благодаря включению в Android Studio пакета для разработчиков Android SDK, процедура установки значительно упрощается. Она сводится к загрузке, распаковке содержимого zip-файла и запуску инсталлятора studio.exe из папки Bin.

Об Android Studio подробнее написано здесь: <https://developer.android.com/sdk/installing/studio.html>

### Android SDK

Вероятно, потребуется и загрузка дополнительных пакетов SDK. Удобнее всего сделать это при помощи кнопки "SDK Manager" на панели инструментов интерфейса Android Studio. Конфигурирование и настройка Android SDK - тема для отдельной статьи, поэтому останавливаться на этом мы не будем. Подробная информация о дополнительных SDR подобрана на этой странице:

<https://developer.android.com/sdk/installing/adding-packages.html>

Если SDK-пакеты для Android уже установлены в системе, платформа Android Studio позволяет указать правильный путь к их расположению в файловой системе, см. Рисунок 4.

Для вызова этого меню в Android Studio щёлкаем по *Configure -> Project Defaults -> Project Structure.* В появившемся диалоге можно прописать путь к папке с SDK.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | илоьии |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |



***Рисунок 4.*** *Указываем путь до SDK в Android Studio*

### Тестируем Intel Edison

До начала ознакомления с возможностями платы (ниже), настоятельно рекомендую убедиться, что платформа способна корректно запустить скетч-образец **Blink**. Скетч включён в состав загружаемого дополнения Arduino IDE; его расположение: *examples -> 01.Basics -> Blink* (папка).

Разработчики Edison подготовили приличный гайдбук для начинающих (англ.):

<https://communities.intel.com/community/makers/edison/getting-started>

### Неувядающая проверочная классика Hello World для Android

После установки Android Studio убедитесь, что программа позволяет создавать новые проекты и запускать их на устройстве с Android.

1. Подсоединяем Android-девайс к компу.
2. Открываем Android Studio.
3. Выбираем "Новый проект…".
4. Указываем имя и путь, 3 раза жмём "Далее" (API 15/Blank Activity).
5. Жмём “Финиш” и ждём, пока проект создастся (процесс может растянуться секунд на 20+).
6. Жмём зелёную иконку Play в тулбаре.
7. Выбираем наш девайс и жмём "OK".

Если всё было сделано правильно, то на экране девайса появится “Hello world!”:



***Рисунок 5****: Приложение Hello World для Android Studio*

## Как работает BLE?

Принцип работы BLE описан уже в его названии: Low Energy. Протокол подразумевает передачу данных короткими пакетами по необходимости, затем - выключение передатчика. Низкое энергопотребление частично достигается применением именно этого принципа. Вместо классического тандема в обычном Bluetooth, устройства BLE связываются друг с другом лишь при необходимости отправки или получения информации.

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). Дескрипторы помогают определить тип данных. К примеру, разберём сервис под названием "Heart Rate Monitor" (монитор пульсомера). Среди его характеристик присутствуют такие, как "измерение пульса".

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

## Ключевые термины и концепции протокола BLE

Предлагаю вашему вниманию краткий обзор ключевых терминов протокола BLE и его концепций. До начала работы над проектом BLE нужно понимать каждый из них.

### Профиль общих атрибутов (GATT)

Профиль общих атрибутов (General Attribute Protocol, GATT) - это обязательный профиль с общими спецификациями отправки и приёма коротких порций данных, известных в Bluetooth Low Link под названием “атрибуты”. Все нынешние профили приложений LE основаны на GATT. Институт стандартизации и разработки протокола - Bluetooth Special Interest Group (SIG) ([https://www.bluetooth.org](https://www.bluetooth.org/)) уже задал для устройств BLE несколько профилей. Эти профили представляют собой спецификации, описывающие способ применения и взаимодействия с устройствами.

### Протокол атрибутов (ATT)

Протокол атрибутов (он же - Attribute Protocol, или ATT) - основывается на GATT. ATT - оптимизированный протокол, созданный исключительно для устройств BLE. Принцип ATT - отсылать столь малое количество байтов, насколько это возможно. У каждого атрибута есть уникальный универсальный идентификатор (англ.: Universally Unique Identifier - UUID). UUID представляет собой стандартизированный 128-битный строковый ID, используемый для идентификации уникальной информации. Формат атрибутов, передаваемых как ATT, бывает двух типов: характеристики и сервисы (определения даны ниже).

### Характеристика (Characteristic)

Характеристика содержит однозначный параметр, а также дескрипторы. Количество дескрипторов может быть равно нулю, т.е. это не обязательная часть характеристики. Дескрипторы описывают значение характеристики.

### Дескрипторы (Descriptors)

Дескрипторы представляют собой определённые атрибуты, которые описывают значение характеристики. Дескрипторы могут быть в виде понятных описаний на вполне человеческом языке, определять единицы измерений, а также задавать ряд допустимых значений.

### Сервис (Service)

Сервис - совокупность характеристик. Список существующих профилей на основе GATT можно просмотреть здесь: <https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx>

## Отправка данных из Android на Intel Edison

### Что нам понадобится

Оставшаяся часть данной статьи подразумевает, что у читателя есть система для разработки, сконфигурированная как на стороне Intel Edison, так и для Android. Рекомендую взглянуть на список следующих этапов предварительной подготовки, а при необходимости вернуться к предыдущим частям статьи.

* Установленная IDE для Intel Arduino
* Установленные драйверы для Intel Edison
* Установленный пакет Android Studio
* Установленный Android SDK
* Возможность развёртывания и запуска “демки” Blink на Intel Edison
* Возможность развёртывания и запуска пустого проекта “Hello world” на Android.

### BLE в Android

Готовый проект можно утянуть с GitHub:

<https://github.com/adrianstevens/Edison_to_Android_BLE/tree/master/Android/BLEConnect>

Однако я рекомендую создать свой собственный проект, вводя код строчку за строчкой, основываясь на ссылке выше.

#### **Создание нового проекта**

Открываем Android Studio (или другую IDE на ваш выбор), создаём новое пустое Android-приложение и называем его BLEConnect. Убедитесь, что минимальный уровень SDK установлен не меньше, чем на API 18: иначе использовать API для BLE не получится.



***Рисунок 6.*** *Создание нового приложения для Android*

Затем добавляем в открытый файл AndroidManifest.xml над тегом **<application>** требуемые разрешения:

<uses-permission android:name="android.permission.BLUETOOTH"/>

<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"/>

### Настройка UI

Не будем усложнять себе жизнь: используем стандартную разметку по умолчанию. Тем не менее, при этом потребуется установить ID для нашего TextView. Для этого нужно открыть файл activity\_main.xml в меню layout, выбрать TextView и установить id в значение **mainText** Теперь мы можем ссылаться на него в нашем коде.



***Рисунок 7.*** *установка id для TextView*

### MainActivity

C

### Переменные Class Level и статические значения

Нам потребуется сохранить несколько переменных, так как мы соединяемся с модулем BLE, ищем сервисы и отсылаем заданное сообщение. Также мы добавим несколько известных статических значений для модуля Grove BLE v1 (чип TI CC2540). При использовании вами другого модуля может потребоваться изменение этих значений в соответствии с чипом. Я особенно рекомендую задать значения характеристик приёма и передачи, к примеру:

CHARACTERISTIC\_TX = "0000ffe1-0000-1000-8000-00805f9b34fb"

CHARACTERISTIC\_RX = "0000ffe1-0000-1000-8000-00805f9b34fb"

### Метод Status helper

Для упрощения нашей задачи будем показывать прогресс в TextView, который мы уже отметили для этой цели. Код включает в себя простой метод-”хелпер” под названием **statusUpdate** - им мы воспользуемся для вывода сообщений о статусе и на экране, и на консоли. Также метод размещает статус в треде UI, что позволит нам безопасно вызвать его из любого треда.

### Подключение к BLE-устройству

Для получения данных ссылочного типа (референса) для диспетчера Bluetooth сначала проверяем, доступен ли BLE на нашем устройстве. Получение референса для диспетчера Bluetooth **(BluetoothManager)** происходит вызовом метода **getSystemService**, затем - метода **getAdapter()**. Альтернативный способ - использование метода статичного хелпера **getDefaultAdapter** прямо из класса диспетчера Bluetooth Manager.

### Поиск BLE-устройств вокруг

Для поиска устройств с заданным промежутком времени мы воспользуемся таймером. Обращение к объекту в Bluetooth-менеджере происходит при помощи **startLeScan**, а при обнаружении устройств передаётся обратный вызов.

API постоянно производит сканирование, и, в результате, мы, скорее всего, получим многочисленные уведомления в **LeScanCallback** для каждого устройства. Поэтому перед добавлением устройства следует убедиться, что данное устройство уникально. Также мы проверим имена устройства для нашего модуля и добавим их к списку. Однако на самом деле для данного примера не требуется сохранять устройства в список.

### @Overridepublic void onLeScan(final BluetoothDevice device, final int rssi, byte[] scanRecord) { …}Обнаружение сервиса связи

Большинство BLE-устройств выделяют один или большее количество сервисов для связи / взаимодействия. У чипа TI CC2540 в модуле Grove BLE есть ключевой сервис с ID “0000ffe0-0000-1000-8000-00805f9b34fb”. Нужно отыскать и сохранить референс этого сервиса.

Сначала подключимся к устройству. Чтобы получить уведомление о подключении (или об обнаружении данного сервиса), понадобится объект **BluetoothGattCallback,** который мы подменяем объектами **onConnectionStateChanged** и **onServicesDiscovered**.

После получения уведомления в методе **onConnectionStateChanged**, которое мы получаем при соединении, опрашиваем **mBluetoothGatt.discoverServices()** для поиска сервисов. После обнаружения нужного сервиса можно продолжить работу и отправить сообщение.

@Override
public void onConnectionStateChange (BluetoothGatt gatt, int status, int newState)
{
 …
}

@Override
public void onServicesDiscovered (BluetoothGatt gatt, int status)
{
 …
}

Существуют и другие методы, которые можно замещать. Документация на эту тему расположена здесь:

<https://developer.android.com/reference/android/bluetooth/BluetoothGattCallback.html>

#### **Отправка сообщения**

В коде-образце присутствует метод **sendMessage**. Мы определяем требуемые характеристики посредством его UDID и обращаемся к **setValue** в характеристиках. Наконец, в нашей сноске BluetoothGatt мы обращаемся к **writeCharacteristic**, отсылая туда характеристики в виде значений для отправки данных.

Для **setValue** существует несколько перегрузок. Есть и более простой способ перегрузки для отправки строковых данных; но, так как большинство BLE-устройств отправляют команды в виде байтов, данный образец будет полезнее.

Настала очередь сборки Intel Edison.

### Сборка Intel Edison, платы Grove и модуля BLE

Начнём со сборки основного “железа”. Если до этого момента элементы конструкции были в разобранном виде, то подключаем Intel Edison в интерфейсную плату Arduino.

Затем вставляем “шилд” Grove, выровнив пины на нижней части “шилда” в соответствии с разъёмами на плате Arduino. Наконец, подключаем модуль Grove BLE v1 к последовательному порту UART.



***Рисунок 8.*** *Включённая плата Intel Edison с подключёнными модулями Grove Shield и BLE*

### Наш первый скетч

Цель работы - установка простого последовательного соединения между устройством Android и Intel Edison. Но, в то же время, есть желание и посмотреть, что именно отправляется и принимается в ходе этого процесса. Для этого воспользуемся Serial Monitor, встроенным в интегрированную среду разработки (IDE) Arduino.

Завершённая версия рассматриваемого скетча находится по этой ссылке:

<https://github.com/adrianstevens/Edison_to_Android_BLE/tree/master/Sketches/SimpleSerial>

Открываем IDE Intel Arduino и создаём новый скетч. Сохраняем его под названием “SimpleSerial”. В отличие от некоторых совместимых с Arduino плат, у Intel Edison два доступных последовательных порта. Весмьа полезное решение, которое позволяет ПК подключаться к Edison, в то время как сам Edison отсылает и принимает данные посредством Grove BLE. Главный последовательный UART подключается к ПК через порт microUSB, а для подключения к модулю BLE мы воспользуемся интерфейсом UART на модуле Grove.

Скетч запускается автоматически при его развёртке на Intel Edison. Сначала он запускает функцию **setup()**, затем постоянно опрашивает функцию **loop()**. Это позволяет считывать и отвечать на входящие с последовательного интерфейса данные.

#### **Инициализация последовательных соединений**

По умолчанию скорость передачи данных Grove BLE составляет 9600 бод, с этого и начнём. Потребуется сконфигурировать оба последовательных порта для работы на этой скорости. Также понадобится пара AT-команд модулю Grove BLE для сброса и начала работы в данном режиме. Всё описанное можно просмотреть в функции **setup()** нашего скетча.

Отмечу, что сначала следует сконфигурировать “Serial” (это microUSB-порт UART), а затем - “Serial1” (UART, соединённый с Grove BLE).

#### **Цикл**

Единственная задача данного скетча - чтение данных с обоих последовательных портов и отправка этих данных с одного на другой. Для этого обратимся к функции **read()** на последовательных портах, которая возвращает один символ, затем используем **print()** на другом последовательном порту.

Цикл у Edison работает достаточно быстро, поэтому 9600 бод не станут помехой.

### Развёртывание скетча

Нажимаем на кнопку Verify в IDE Arduino (на галочку), и фиксим возможные ошибки. После проверки нужно убедиться, что Intel Edison соединён с ПК и передаёт скетч (правая стрелка). После завершения передачи, скетч входит в цикл - и мы готовы соединиться с ним из приложения Android. Открываем Serial Monitor на Arduino IDE (лупа сверху справа) - и можно получать и отправлять данные.

Откройте приложение Android BLEConnect после запуска скетча на Intel Edison. На Serial Monitor должно появиться сообщение “Hello Grove BLE”.

Если что-то не сработало, то, вероятнее всего, проблема кроется на стороне приложения Android. Проверьте индикатор состояния, который должен подсказать, где произошёл сбой.

В репозитории GitHub выложен скетч, который также отправляет сообщение на экран Grove LCD. Убедитесь, что “шилд” Grove установлен на 5В и подсоедините LCD-монитор к любому интерфейсу I2C.



***Рисунок 9.*** *Запуск BLEConnect на смартфоне*



***Рисунок 10.*** *Adruino IDE Serial Monitor получает BLEConnect-сообщения*

### Что дальше

Создание более сложного проекта означает закладку архитектуры и в код для Android, и в скетч. Я рекомендовал бы перенести большую часть кода Android BLE в сервис, чтобы отделить его от UI и облегчить его применение для разнообразных задач и проектов. При создании более продвинутых скетчей пригодится использование библиотеке Arduino Time Library, которая позволяет воспроизводить запуск многократных циклов при одновременном получении данных (<http://playground.arduino.cc/Code/Time>). Собственные примеры кода я буду выкладывать в данный репозиторий GitHub (<https://github.com/adrianstevens/Edison_to_Android_BLE>), а обсуждение будущих концепций - тема для отдельной статьи.