**Особенности взаимодействия Unity-приложений с переносимыми устройствами по протоколу Bluetooth Low Energy**

Начать стоит с особенностей и инструментария среды разработки Unity. Большинство функций для работы с операционной системой Android уже встроено в метаданные движка, например взаимодействие с файловой системой, звуком, камерой, wi-fi - с теми инструментами, которые чаще всего используются при разработке приложений под Android. Но Bluetooth не входит в часто используемые инструменты, поэтому, для его интеграции, следует использовать сторонние библиотеки, или писать свои для обращения к API операционной системы. Основной язык ОС Android является Java, официальная документация включает в себя примеры, написанные именно на этом языке, поэтому взаимодействие с API желательно осуществлять с него или с Kotlin.

Bluetooth был задуман, как технология связи ближнего диапазона, призванная заменить провода в таких устройствах, как компьютерные мыши, клавиатуры или персональные компьютеры.

В настоящее время существует два типа устройств с поддержкой Bluetooth:

• Bluetooth Classic - используется в беспроводных наушниках, автомобильных сигнальных системах и там, где обязательно непрерывное соединение для передачи аудиоданных;

• Bluetooth Low Energy – Bluetooth с низким энергопотреблением. Появился с версии Bluetooth 4.0 и используется почти повсеместно в приложениях, датчиках, устройствах, которым важно сохранять заряд на протяжении долгого времени.

Поскольку во большинстве современных устройствах используются большое количество датчиков и сенсоров, BLE стал наиболее часто используемым протоколом связи.

Основная задача стоит такая: написать Android приложение, которое должно уметь взаимодействовать с BLE устройствами.

Первое, что необходимо для реализации – интеграция Java кода в среду разработки Unity. Для интеграции Java, необходимо написать плагин и включить его в сборку основного проекта Unity. Для этого нужно использовать встроенные классы AndroidJavaObject. В переменной этого типа будет храниться экземпляр класса библиотеки. Для верной работы плагина, необходимо использовать шаблон проектирования Singleton. С помощью этого шаблона, состояние экземпляра класса будет сохранять своё состояние на протяжении всего жизненного цикла основного приложения, что важно при работе с Bluetooth. Далее нужно взаимодействовать с контекстом (Context) приложения для обращения к базовыми функциям – которые как раз включают в себя объекты и методы BLE.

При написании Java плагина для основного приложения можно сразу столкнуться с неявными ошибками. Во-первых – **разрешения**. Для взаимодействия с большинством методов из Android API, требуются разрешения от пользователя. При использовании BLE обязательны следующие:

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

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_BACKGROUND\_LOCATION" />

Разрешение ACCESS\_COARSE\_LOCATION (отслеживание местоположения) Google считает «опасным» и для него требуется обязательное согласие пользователя. [https://developer.android.com/guide/topics/connectivity/bluetooth/permissions#availability]

Если приложение запущено на телефоне с Android версии 8.0 и выше, то требуются доп. Разрешения:

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

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

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

На этом моменте начинаются первые подводные камни.

Нужно учитывать, что на андроиде версии ниже, данных разрешений не существует и при запросе будет возвращено значение 2 – что означает: «пользователь отклонил навсегда»

Разрешения запрашиваются асинхронно, и можно запросить сразу несколько, но, например, у разрешения местоположения текст уведомления отличается, и если его запросить вместе со всеми, то оно автоматически отклониться. Это нужно учитывать и разделить разрешения на группы, после чего вести опрос по очереди.

Android BLE API это низкоуровневые операции, в реальных приложениях нужно использовать несколько слоев абстракции. Поэтому обычно нужно самостоятельно сделать:

* очередь команд,
* обслуживание соединений,
* обработку ошибок и багов,
* многопоточный доступ.

Всё вышеперечисленное можно реализовать, как в плагине, так и из основного приложения.

В многопоточный доступ стоит углубиться дальше. Скорее всего, приложение, которое реализовано для взаимодействия с BLE устройствами, должно будет работать в фоновом режиме. Но проблема заключается в том, что Unity-приложение останавливает свою активность при блокировке или сворачивании приложения. Для решения проблемы работы в фоновом режиме есть 2 возможных варианта, которые удобны для решения поставленной задачи.

1. Если скрипт работает не в основном потоке, то, при остановке активности, он продолжит исполняться какое-то время, пока операционная система не решит, что процесс занимает слишком много процессорного времени, после чего остановит его принудительно. Отталкиваясь от этой информации, можно реализовать алгоритм взаимодействия и работы с Java плагином не из основного потока.

Данный вариант решения подойдет для тех приложений, которые должны продолжать работу с BLE на протяжении короткого времени (+- час). Но реализация многопоточности в Unity требует использования определенных костылей.

1. Один из лучших и верных решений, это создание сервиса, который будет работать в фоне, пока пользователь не выключить приложение принудительно. Такой вариант подходит для тех приложений, которые должны обеспечивать соединение и передачу данных по BLE на протяжении долгого времени, например если требуется отслеживать присутствие устройства. Написание собственного сервиса требует углубленных знаний Java и ОС Android. Также нужно учитывать то, что основное приложение будет находиться в паузе, поэтому вся основная логика, должна быть прописана также в плагине.

Также при взаимодействии приложения с BLE устройствами в фоновом режиме, нужно учесть: [https://habr.com/ru/articles/536392/]

* с Android 7 запуск и останов сканирования более 5 раз за 30 секунд временно отключает сканирование.
* c Android 7 вы можете сканировать только в течение 30 минут, после чего Android меняет параметры на SCAN\_MODE\_OPPORTUNISTIC. Очевидное решение, перезапускать сканирование с периодом менее, чем 30 мин.
* c Android 8.1 сканирование без фильтров блокируется при выключенном экране. Если у вас нет никаких ScanFilters, Android приостановит сканирование, когда экран выключен и продолжит, когда экран снова будет включен.
* Для работы приложения или сервиса необходимы разрешения:  
  FOREGROUND\_SERVICE   
  и   
  REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS

Отправка и приём данных по протоколу BLE реализованы в ассинхронных методах. При передаче данных нельзя, не дождавшись окончания, начинать отправлять следующую информацию, иначе операционная система прервёт подключение. Как уже было написано выше, реализовывать очередь придётся самому. Возврат ответа на запись или чтение приходит в соответствующих Gatt callback’ах, но на приём существует 2: OnReadCharacteristic и OnChangeCharacteristic. В какое из прерываний придёт ответ – зависит от устройства, к которому приложение должно подключиться, поэтому для реализации универсального плагина, придётся учитывать этот факт. Для передачи данных на отправку из основного приложения Unity в плагин нужно использовать базовые типы данных (такие как int, string и массивы Byte), так как обращение к библиотекам через AndroidJavaObject довольно ресурсоёмкий.