**ВВЕДЕНИЕт**овый редактор приложение

В настоящее время мобильные технологии интенсивно развиваются. Возможности, доступные для мобильных телефонов и планшетов, постоянно расширяются, что приводит к необходимости разрабатывать новые решения и приложения.

Миллионам людей по всему миру известны преимущества приложений, используемых как средство быстрого обмена электронными сообщениями между друзьями, коллегами, родственниками и просто знакомыми. С проникновением Интернета в мобильные телефоны, электронное общение стало неотъемлемой частью большинства людей. Теперь нет таких мест, где было бы невозможно обменяться информацией, мыслями, шутками или просто наблюдениями с собеседником. При помощи чатов можно абсолютно комфортно общаться в библиотеках, на собраниях или совещаниях, на парах или уроках, на концертах и т.п. А при помощи Bluetooth-чата можно делать это безплатно. Используя Bluetooth-чат, не нужно подключаться к Интернету и платить за трафик, можно экономить, при общении с людьми, находящимися рядом.

1. **BLUETOOTH ТЕХНОЛОГИЯ**

Своё название технология Bluetooth получила в честь короля викингов, которого звали Харольд Голубой Зуб. Такое прозвище ему дали из-за потемневшего переднего зуба. Харольд жил в Дании более тысячи лет назад и вошёл в историю благодаря тому, что объединил данов и принёс им христианство. И, как напоминание о ведущей роли скандинавов в мобильной связи, компании Ericsson, IBM, Nokia и Toshiba назвали свою технологию, которая должна творить историю, именем воина воина-викинга.

Bluetooth позволяет объединять в локальные сети любую технику: от мобильного телефона и компьютера до холодильника. При этом, одними из немаловажных параметров новой технологии должны были стать низкая стоимость устройства связи, соответственно небольшие размеры (ведь речь идет о мобильных устройствах) и, что немаловажно, совместимость, простота встраивания в различные устройства.

1. **ИСПОЛЬЗОВАНИЕ BLUETOOTH В ANDROID**

Платформа Android предоставляет разработчику богатые коммуникационные возможности. Для работы с Bluetooth в состав Android входит мощный API, позволяющий легко производить сканирование окружающего пространства на предмет наличия готовых к соединению устройств, передачу данных между устройствами и многое другое.

Работа с Bluetooth состоит из четырех этапов:

* установка настроек bluetooth адаптера;
* поиск доступных для соединения устройств;
* установка соединения;
* передача данных.

1. **ОБЗОР ANDROID BLUETOOTH API**

Bluetooth API располагается в пакете android.bluetooth. В его состав входит несколько классов:

**BluetoothAdapter**- отвечает за работу с установленным в телефоне Bluetooth модулем. В состав этого класса входят методы, позволяющие производить поиск доступных устройств, запрашивать список подключенных устройств, создавать экземпляр класса BluetoothDevice на основании известного MAC адреса и создавать BluetoothServerSocket для ожидания запроса на соединение от других устройств.

**BluetoothDevice-** класс, ассоциирующийся с удаленным Bluetooth устройством. Экземпляр этого класса используется для соединения через BluetoothSocket или для запроса информации об удаленном устройстве (имя, адресс, класс, состояние).

**BluetoothSocket-** интерфейс для Bluetooth socket, аналогичный TCP сокетам. Это точка соединения, позволяющая обмениваться данными с удаленным устройством через InputStream и OutputStream.

**BluetoothServerSocket-** представляет открытый сокет сервера, готовый к обработке входящего запроса. Для того чтобы соединить два Android устройства, одно из них должно открыть сокет с помощью этого класса. Когда удаленное устройство пошлет запрос на соединение, BluetoothServerSocket вернет объект BluetoothSocket.

**BluetoothClass-** описывает основные параметры Bluetooth модуля. Объект этого класса доступен только в режиме чтения и может быть полезен при определении типа устройства.

1. **РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ**

**4.1. УСТАНОВКА НАСТРОЕК BLUETOOTH АДАПТЕРА ИЗ ANDROID**

При использовании в своей программе возможности Bluetooth модуля, необходимо, прежде всего, подключить соответствующий пакет API.

import android.bluetooth.\*;

Дальше необходимо дать приложению разрешение на использования Bluetooth модуля. Для этого в манифест программы нужно добавить строку.

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

Для использования критических с точки зрения безопасности возможностей, например, изменить имя устройства, то в манифест программы нужно добавить строку.

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

Прежде чем соединяться с кем-нибудь и передавать данные нужно убедиться, что устройство имеет bluetooth модуль. Первым делом при работе с bluetooth API нужно создать экземпляр класса BluetoothAdapter.

BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter ();

Если устройство не поддерживает bluetooth, будет возвращено значение "null". На практике нужно всегда проверять это условие, чтобы избежать ошибок.

if(bluetooth!=null)

{

// С Bluetooth все в порядке.

}

Даже если ваш аппарат оснащен Bluetooth модулем, он может быть недоступен, поскольку пользователь просто отключил его. Для проверки доступности Bluetooth служит метод isEnabled(). В случае, если модуль отключен, можно предложить пользователю включить его.

if (bluetooth.isEnabled()) {

// Bluetooth включен. Работаем. }

else{

// Bluetooth выключен. Предложим пользователю включить его.

Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION\_REQUEST\_ENABLE);

startActivityForResult(enableBtIntent, REQUEST\_ENABLE\_BT);}

**4.2. ОРГАНИЗАЦИЯ ПОИСКА ДОСТУПНЫХ BLUETOOTH УСТРОЙСТВ**

С помощью класса BluetoothAdapter, Вы можете найти удаленное bluetooth устройство, запустив сканирование или запросив список спаренных устройств.

При сканировании осуществляется поиск доступных bluetooth модулей вокруг вас. Если в поле досягаемости окажется устройство с разрешенным bluetooth, оно отправит в ответ на запрос некоторую информацию о себе: имя, класс, свой уникальный MAC адрес. На основе этой информации можно организовать соединение и передачу данных.

Сразу после установки соединения с удаленным устройством, пользователю будет автоматически показан запрос на соединение. В случае положительного ответа полученная информация (имя, класс и MAC адрес) сохраняется и может затем использоваться через bluetooth API. Так при следующем сеансе связи с данным удаленным устройством вам уже не придется проводить сканирование, поскольку необходимый MAC адрес уже будет занесен в базу вашего телефона и его можно просто выбрать из списка спаренных устройств.

Необходимо различать понятие спаренных и соединенных устройств. Спаренные устройства просто знают о существовании друг друга, имеют ссылку-ключ, которую могут использовать для аутентификации, и способны создать шифрованное соединение друг с другом. Соединенные устройства разделяют один радиоканал и могут передавать данные друг другу. Текущая реализация bluetooth API требует, чтобы устройства были спарены перед соединением. (Спаривание выполняется автоматически, когда вы начинаете шифрованное соединение через Bluetooth API)

**4.3. ЗАПРОС НА СОЕДИНЕНИЕ СО СПАРЕННЫМ УСТРОЙСТВОМ**

Прежде чем приступать к поиску устройств вокруг имеет смысл показать пользователю список уже известных системе устройств. Вполне возможно, что требуемое устройство окажется в этом списке. Метод getBondedDevices() возвращает множество (Set) устройств BluetoothDevice, с которыми уже происходило соединение.

**4.4. ПОИСК УСТРОЙСТВ**

Для того, чтобы начать сканирование радиодиапазона на предмет наличия доступных устройств необходимо вызвать метод startDiscovery(). Сканирование происходит в отдельном асинхронном потоке. Метод возвращает true, если запуск сканирования прошел успешно. Обычно процесс сканирования занимает порядка 10-15 секунд. Чтобы получить информацию о найденных устройствах приложение должно зарегистрировать BroadcastReceiver для интента ACTION\_FOUND. Этот интент вызывается для каждого найденного устройства. Интент содержит дополнительные поля EXTRA\_DEVICE и EXTRA\_CLASS, которые содержат объекты BluetoothDevice и BluetoothClass соответственно.

Поиск Bluetooth устройств требует много ресурсов. Как только устройство нашло подходящее устройство, процесс сканирования остановиться. Это делается при помощи метода cancelDiscovery().

**4.5. СОЕДИНЕНИЕ УСТРОЙСТВ**

Чтобы соединить два устройства, необходимо написать серверную и клиентскую часть кода. Одно из устройств должно открыть серверный сокет, а второе - инициализировать соединение, используя MAC адрес сервера. Сервер и клиент считаются соединенными, когда они оба имеют активный BluetoothSocket на одном и том же RFCOMM канале. После этого они могут получать и отправлять потоки данных. Сервер и клиент по-разному получают требуемый BluetoothSocket. Сервер получает его, когда входящее соединение принято. Клиент - когда открывает RFCOMM для сервера.

**4.6. СЕРВЕР**

При соединении устройств одно из них должно вести себя как сервер, то есть удерживать открытый BluetoothServerSocket. Цель сервера - ждать запроса на входящее соединение, и когда оно подтверждено, создать BluetoothSocket. После этого BluetoothServerSocket можно закрыть. Рассмотрим поэтапно процедуру соединения с точки зрения сервера:

Получить BluetoothServerSocket вызвав метод listenUsingRfcommWithServiceRecord(String, UUID). Первый параметр метода представляет собой идентификационное имя вашего сервиса. Система автоматически добавит его в базу Service Discovery Protocol (SDP). Обычно в качестве этого параметра просто указывают название приложения. Второй параметр также идентифицирует сервис. Этот параметр используется клиентом при подтверждении соединения.

Начинаем прослушивать запрос на соединения через метод accept(). Это блокирующий метод, который возвращает результат либо, когда соединение подтверждено, либо, когда произошло исключение. Соединение считается подтвержденным, когда удаленное устройство пошлет запрос на соединение с UUID, указанным при регистрации серверного сокета. В случае успеха, accept() возвращает настроенный на соединение BluetoothSocket.

Если Вы хотите принять дополнительное соединение, вызовите метод close(). Это приведет к освобождению сокета и всех его ресурсов, но не закроет соединенный BluetoothSocket. В отличие от TCP/IP, RFCOMM позволяет работать только с одним клиентом на канале, поэтому в большинстве случаев имеет смысл вызывать метод close() срезу же после установки принятия сокета.

Поскольку метод accept() является блокирующим, его не стоит вызывать из потока главной деятельности, поскольку это приведет к подвисанию интерфейса. Обычна вся работа с BluetoothServerSocket и BluetoothSocket выполняется в отдельном потоке. Чтобы прекратить выполнение метода accept(), вызовается метод close() для BluetoothServerSocket (или BluetoothSocket) из любого другого потока приложения.

**4.7. КЛИЕНТ**

Для инициализации соединения с удаленным устройством (устройством, которое держит открытым серверный сокет) необходимо получить объект BluetoothDevice, содержащий информацию о нем. Этот объект используется для получения BluetoothSocket и инициализации соединения.

Процедура соединения: Получим BluetoothSocket вызвав метод BluetoothDevice.createRfcommSocketToServiceRecord(UUID). Значение параметра UUID должно совпадать с значением, указанным при вызове listenUsingRfcommWithServiceRecord сервера.

Инициализируем соединение, вызвав метод connect(). После вызова этого метода система будет выполнять SDP поиск на удаленном устройстве, чтобы сопоставить UUID. В случае успеха при условии подтверждения запроса со стороны сервера будет открыт RFCOMM канал. Это блокирующий вызов. Если по каким-то причинам соединение сорвется или выйдет timeout (около 12 секунд), будет сгенерировано исключение.

Как и в случае с accept, метод connect() следует выполнять в отдельном потоке, в противном случае может произойти подвисание интерфейса.

Замечание. Прежде чем вызывать connect() убедитесь, что в данный момент не происходит сканирование с целью поиска доступных устройств. В случае одновременного выполнения этих операций соединение будет устанавливаться намного медленнее, и вы рискуете не уложиться в timeout.

Для остановки сканирования эфира вызывается метод cancelDiscovery(). Перед вызовом этого метода можно проверить идет ли сканирование с помощью isDiscovering().

После завершения работы с BluetoothSocket всегда вызывайте метод close(). Это поможет сэкономить ресурсы телефона.

**4.8. ПЕРЕДАЧА ДАННЫХ**

После успешного соединения, каждое из соединенных устройств имеет объект BluetoothSocket с помощью которого легко реализовать передачу/прием данных:

С помощью методов getInputStream() и getOutputStream() полуить объекты InputStream и OutputStream, управляющие передачей через сокет.

Читать и писать данные в поток с помощью методов read(byte[]) и write(byte[]).

Вы должны использовать отдельный поток для чтения и записи данных. Это важно, поскольку методы read(byte[]) и write(byte[]) являются блокирующими и их вызов в основном потоке может парализовать вашу программу. Главный цикл в этом отдельном потоке должен считывать данные из InputStream. Для записи в OutputStream имеет смысл создать отдельный public метод.

В конструкторе создаются объекты для работы с потоками данных, после чего поток оживает входящие данные. После того как прочитан очередной блок данных из входящего потока он посылается в главную деятельность посредствам вызова метода Handler родительского класса. Для отправки данных из главной деятельности просто вызывается метод write(). Внутри этого публичного метода происходит вызов write(byte[]). Метод close() также можно вызвать из главной деятельности. Он разрывает соединение.

1. **ВЫБОР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Самая популярная IDE на данный момент — Android Studio, поскольку она разрабатывается Google, интегрирована с Gradle, использует новую структуру проекта по умолчанию, находится в состоянии стабильной сборки и заточена под Android-разработку.

1. **ЗАКЛЮЧЕНИЕ**

В работе было показано, что интерфейс любой программы должен быть простым и интуитивно понятным, а также приятен для восприятия человеком, а точнее пользователем. Красивый и удобный дизайн интерфейса в первую очередь предрасполагает к себе среди конкурентов, которые не уделили этому должного внимания, т.е. можно сказать, что грамотно разработанный и реализованный интерфейс позволяет завоевать большую часть аудитории в той или иной тематике приложений.

**ЛИТЕРАТУРА**

1. Конспект лекций.

2. http://startandroid.ru/

3. Харди Б., Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов. 2-е изд. — СПб.: Питер, 2016.