BLUETOOTH PAIRING

Pairing and bonding AINA's devices with phone by using half automated connection, QR-code or NFC

Version 1.20

PUBLIC

Complicated Connection Process Frustrates the End-Users

Currently users must take several steps before being able to use their PTT Voice Responder with a PTT application. These steps include:

1. Turning their smartphone's Bluetooth
2. Scanning for available devices
3. Finding their own PTT Voice Responder from the list
4. Allowing for the pairing process to begin
5. Opening the PTT application
6. And adjusting the application's settings for the AINA device's buttons to work

The obvious issue with this is that the process is **TOO LONG** and **COMPLICATED** for the users. Leaving them with the question, how is this any better than my good old simple walkie-talkie, which when turned on works instantly?

AINA's Solution

Making Bluetooth PTT devices as simple to use as radios

There are three ways with which we can tackle the complexity of the current connection process:

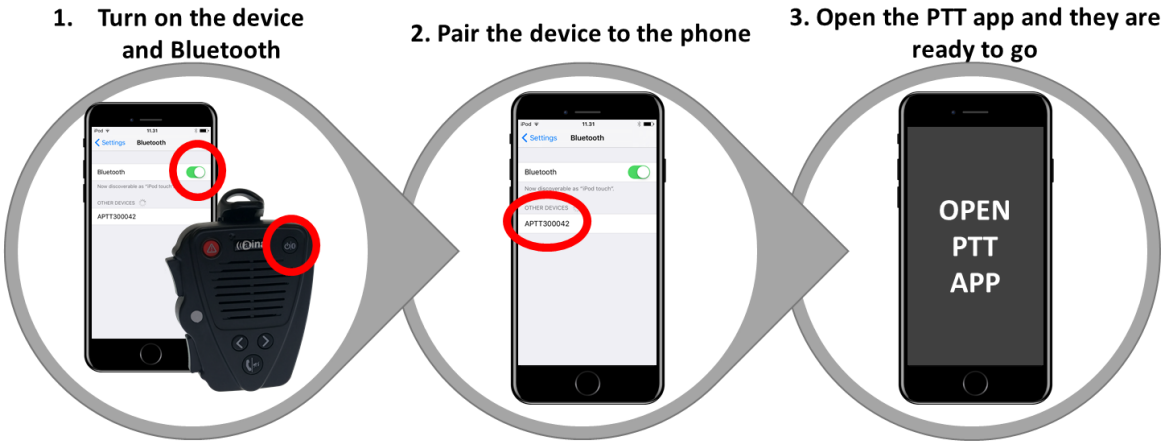
1. **Half Automated Connection** – Remove the need for the users to manually connect their AINA devices to the application after Bluetooth pairing.

2. **NFC Touch Pairing** – Use Near Field Communication (NFC) technology to allow the users to both pair and connect their device to their smartphone and application with a simple touch of the devices.

3. **QR-Code Pairing** – Have the PTT application scan the QR-code found in the back of all AINA devices to pair and connect the device to the user's smartphone and application in use.

Quicker Connection
Half automated connection (Android and iOS)

Most users will not have the time, skills or patience to go through a long pairing process. A step which frequently leaves them confused is the need to still connect the AINA Voice Responder to the PTT app even after successful Bluetooth pairing. Therefore, the solution is to completely remove this step so that users only have to:



More on how to make the PTT app connect automatically to the device in section 6.

Touch and Go!
NFC Touch Pairing (Android Only)

Starting end of March, all of AINA's devices will include Near Field Communication (NFC) tags which will contain all the necessary information to complete the pairing and connection process to NFC-enabled smartphones and PTT applications. The connection process will be as simple as:

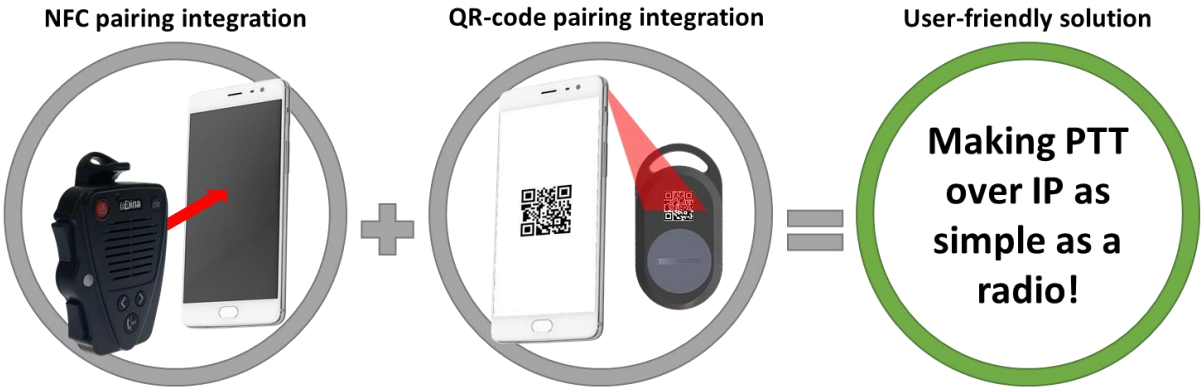
As an app developer, you will need to add the necessary functionality to the application so it can read our NFC tag. This can be made by using Android's NFC API (more on this in section 4).

Scan to Pair
QR-Code Pairing (Android and iOS)

By using their smartphone's camera, users will also be able to pair their AINA Voice Responder by scanning the QR-codes located at the back of their devices with the PTT application. The connection process will be as simple as:

As an app developer, you will need to implement a QR-scanner into your application. Within the Android operating system this can be made using Google's mobile vision API, or for iOS using iOS' AVFoundation framework (more on this in section 5).

The End Goal

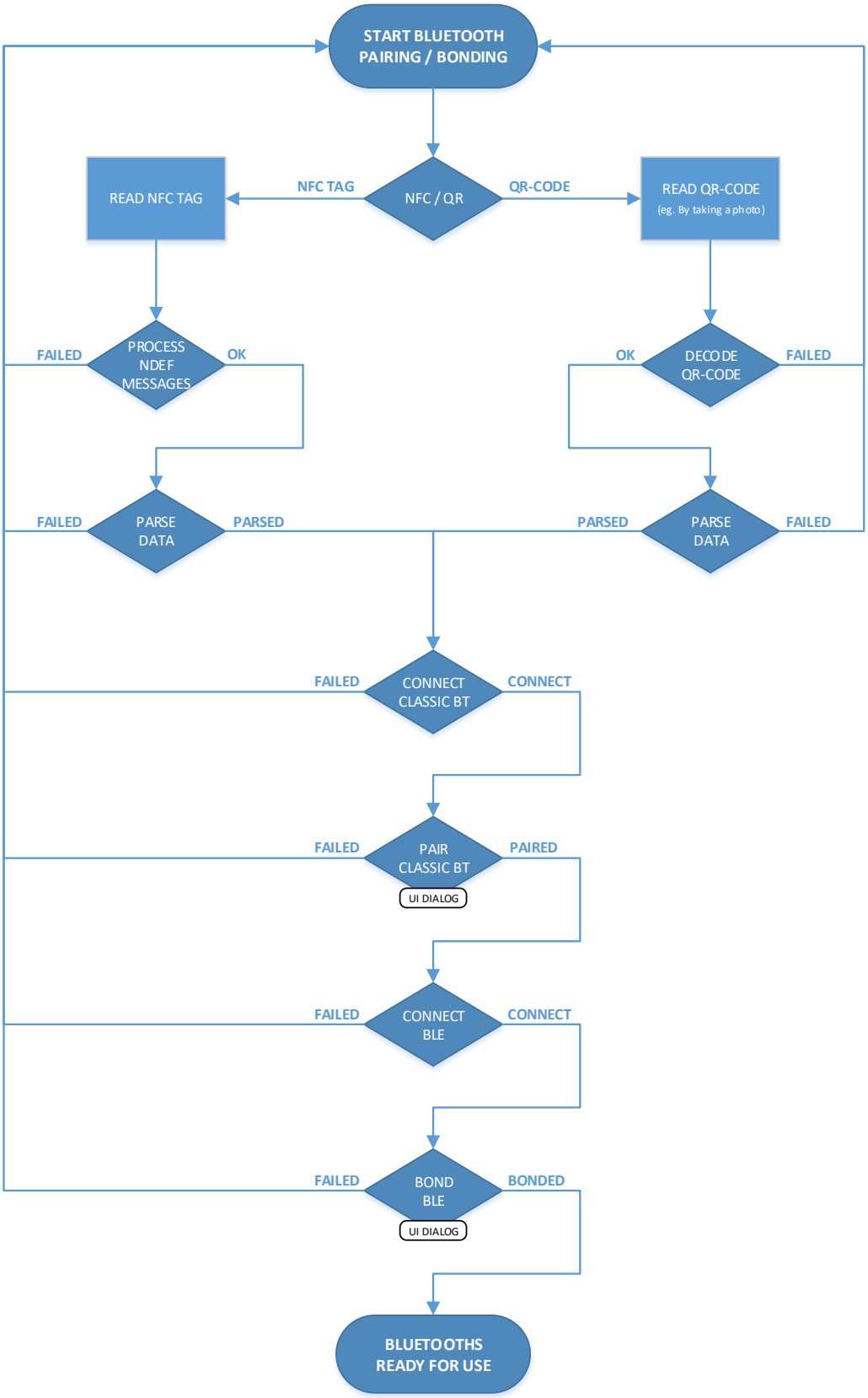


All the needed technical information for application developers to incorporate the necessary functionalities so that half automated connection, NFC touch pairing and/or QR-code pairing works with the application can be found below.

1. Abbreviations

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BR	Basic Rate
BLE	Bluetooth Low Energy
EDR	Enhanced Data Rate
GATT	Generic Attribute Profile
iOS	iPhone Operating System
MAC	Media Access Control
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
OOB	Out-Of-Band
OS	Operating System
PTT	Push-To-Talk
QR	Quick Response
SW	Software
UUID	Universally Unique Identifier

2. Simplified flowchart of pairing and bonding



3. Android and iOS features and limitations

3.1. Android

On Android based phones, it is possible to get access to both the phone's Bluetooth Classic and BLE modules functionalities via Android's Bluetooth API.

If the phone has NFC support, pairing / bonding with Bluetooth devices can be achieved by using an NFC tag to deliver the needed information to conduct Bluetooth Classic pairing and BLE bonding. The necessary functionality to read the NFC tag can be made by using Android's NFC API.

Also, if the phone has a camera, pairing and bonding with Bluetooth devices can be achieved by using a QR-code to deliver the pairing and bonding information to the phone. The implementation of a QR-scanner to the PTT app can be made for example by using Google's mobile vision API.

3.2. iOS

On iOS-based phones, it is only possible to access the phone's BLE module and functionality. This module can be accessed through iOS' Core Bluetooth Framework.

Unfortunately, iOS does not allow access to the phone's NFC reader, so pairing with an NFC tag is not yet an option

However, with iOS devices' camera, bonding with BLE devices can be achieved by using a QR-code to deliver the needed information to conduct bonding. The implementation of a QR-code scanner to the PTT app can be made for example by using iOS' AVFoundation framework.

4. NFC tag handling with Android

Some Android based phones have support for NFC and this can be used to pair and bond Bluetooth devices with them.

AINA's Bluetooth devices (including the AINA PTT Voice Responder), have an NFC tag inside of them. This NFC tag contains pairing information in OOB data format as described in the Bluetooth specification. For NFC transmission, this OOB data becomes the payload of an NDEF record termed as Bluetooth BR/EDR carrier configuration record and BLE carrier configuration record. If the device contains only one module, either Bluetooth Classic or BLE, only this information is included in the NFC tag.

Simplified sequence for reading the NFC tag's payload with NFC API:

1. Crete an NFC adapter object
2. Create an intent handler to get *ACTION_NDEF_DISCOVERED* actions
3. Check that the intent type is OOB *(application/vnd.bluetooth.ep.oob)*
4. Read the received NDEF message
5. Read all the records of the NDEF message
6. Parse the MAC addresses from the payload of the records
7. Ready to make pairing with Bluetooth Classic and bonding with BLE (see section 6)

For more information, see the links list on section 8.

5. QR-code handling

AINA's Bluetooth devices (including the AINA PTT Voice Responder), have a QR-code marked on their casing. This QR-code contains the MAC addresses of the Bluetooth Classic and/or BLE modules (depending on the device)

Example of an actual QR-code used for pairing and its content in ASCII:

C059C4136D58/38B8EB300315

As from the ASCII decoded text it can be seen that there are two slash- separated fields in the QR-code. The first one, C059C4136D58, is an MAC address of the BLE module. The second one, 38B8EB300315, is an MAC address of the Bluetooth Classic module.

5.1. QR-code scanning with Android

Simplified sequence for QR-scanning with the mobile vision API:

1. Create a *BarcodeDetector* object
2. Capture the QR-code image
3. Launch the *mediaScan* intent
4. Decode the image stream
5. Detect the QR-code with the *BarcodeDetector*
6. Parse the MAC addresses from the values (content) of the QR-code
7. Ready to make pairing with Bluetooth Classic and bonding with BLE (see section 6)

For more information, see the links list on section 8.

5.2. QR-code scanning with iOS

Simplified sequence for QR-scanning with the AVFoundation framework:

1. Create an instance of the *AVCaptureDevice* of media type *AVMediaTypeVideo*
2. Create and initialize the AVCaptureSession object
3. Create an *AVCaptureMetadataOutput* object and tell to object we are interested in *AVMetadataObjectTypeQRCode*
4. Create a delegate method *captureOutput: didOutputMetadataObjects:fromConnection,* the only one that the *AVCaptureMetadataOutputObjectsDelegate* provides
5. Parse the MAC addresses from the first object of the *metadataObjects* array
6. Ready to make bonding with BLE (see section 6)

For more information, see the links list on section 8.

6. Connecting, Pairing and bonding

Depending on the application and AINA device (+ firmware version) used, the connection process may differ.

For example, in case of the AINA PTT Voice Responder, which has both a Bluetooth Classic module as well as a BLE module, the pairing and bonding order will depend on the used firmware version. Also, if the AINA device has only a BLE module in it (ex. the AINA PTT Smart Button), the Bluetooth Classic pairing part can be ignored.

Even if NFC or QR-code assisted pairing cannot be done with some of AINA's older devices and firmwares* or PTT applications which are not ready to implement these features, the user experience can be significantly improved by integrating half automated pairing. This is achieved by first pairing the Classic Bluetooth as you would normally do, and then using the Bluetooth Classic's pairing information to connect the AINA device to the BLE module.

*These older devices may also have older firmware versions, which do not support BLE bonding (BLE firmware versions starting with 16xxx). In these devices BLE is used in connected mode without bonding.

For more information, see the links list on section 8.

6.1. Pairing and bonding with Android

Pairing Bluetooth Classic, as well as bonding BLE, can be achieved in several different ways. The simplest way is by using the MAC address' directly to do the pairing and bonding, without doing the Bluetooth device discovery first.

UUID to be used with the AINA PTT Voice Responder with Bluetooth Classic is 00001101-0000-1000-8000-00805F9B34FB.

Simplified sequence for pairing Bluetooth Classic with Bluetooth API using only MAC address information (no device discovery used):

1. Initialize the Bluetooth adapter
2. Request to switch the Bluetooth module on (if switched off)
3. Check if the device is already paired
4. If not, try to connect to the device by using the device's MAC address and UUID
5. Create profile listeners (e.g. for A2DP and HEADSET profiles)
6. Connect to the Bluetooth socket (launches Android's pairing dialog)
7. Bluetooth Classic is ready to be used if pairing was accepted

Simplified sequence for bonding BLE with Bluetooth API using only MAC address information (no device discovery used):

1. Initialize the Bluetooth adapter
2. Request to switch the Bluetooth module on (if switched off)
3. Check if the device is already bonded
4. If not, try to connect to the GATT service with autoconnect set to true (launches Android's bonding dialog)
5. Start service discovery
6. After services are found, register to characteristics notify service (see section 7)

6.2. Connecting and bonding with iOS

Simplified sequence for bonding BLE with AVFoundation framework:

1. Start-up the central manager
2. Discover devices
3. Check if the device is already connected
4. Connect to the desired device (with MAC we have read before)
5. Discover services
6. Compare found services UUIDs to wanted service UUID (like the one mentioned in next section)
7. Bonding happens on iOS automatically when reading the encrypted characteristics (see section 7)

7. Access to BLE's Service Characteristics

Depending of the AINA device, used service UUIDs may differ and the characteristic UUIDs may also vary.

For the AINA PTT Voice Responder, the following applies for the service:

- Service UUID: 127FACE1-CB21-11E5-93D0-0002A5D5C51B

And the following applies for the characteristics:

- Keys pressed UUID: 127BEEF1-CB21-11E5-93D0-0002A5D5C51B
- SW versions UUID: 127C0FF1-CB21-11E5-93D0-0002A5D5C51B

7.1. Reading characteristics with Android

After connecting to the GATT service, you can read the characteristics provided by the GATT.

Simplified sequence for reading characteristics with the Bluetooth API:

1. Connect to GATT (see section 6)
2. Get desired service with the service UUID
3. Get desired characteristic with the characteristic UUID
4. Parse data from the characteristics value array

Or register notification service for desired characteristic. In this case You get notification callback every time a characteristics value changes. The sequence for registering to notification service:

1. Set characteristic notification for desired characteristic
2. Get descriptor for characteristic
3. Enable notification for descriptor
4. Write descriptor
5. Create overridden onCharacteristicChange function, which will be called when characteristic has a new value

For more information, see the links list on section 8.

7.2. Reading characteristics with iOS

After connecting to the GATT service, you can read the characteristics provided by the GATT.

Simplified sequence for reading characteristics with the AVFoundation framework:

1. Connect to the service (see previous section)
2. Discover characteristics of the service
3. Compare the found characteristic UUIDs to the wanted characteristic UUID (for example the ones listed above)
4. Read or subscribe to the characteristic
5. Parse data from the characteristics value property

For more information, see the links list on section 8.

8. Example codes and useful links

AINA Wireless' example codes can be found from GitHub:
https://github.com/AINAWireless

Android & NFC:
https://developer.Android.com/guide/topics/connectivity/nfc/nfc.html

Android & QR-Code:
https://developers.google.com/vision/Android/barcodes-overview

https://code.tutsplus.com/tutorials/reading-qr-codes-using-the-mobile-vision-api--cms-24680

Android & Bluetooth:
https://developer.Android.com/guide/topics/connectivity/bluetooth.html

iOS & QR-Code:
https://www.appcoda.com/qr-code-ios-programming-tutorial/

iOS & BLE:
https://developer.apple.com/bluetooth/