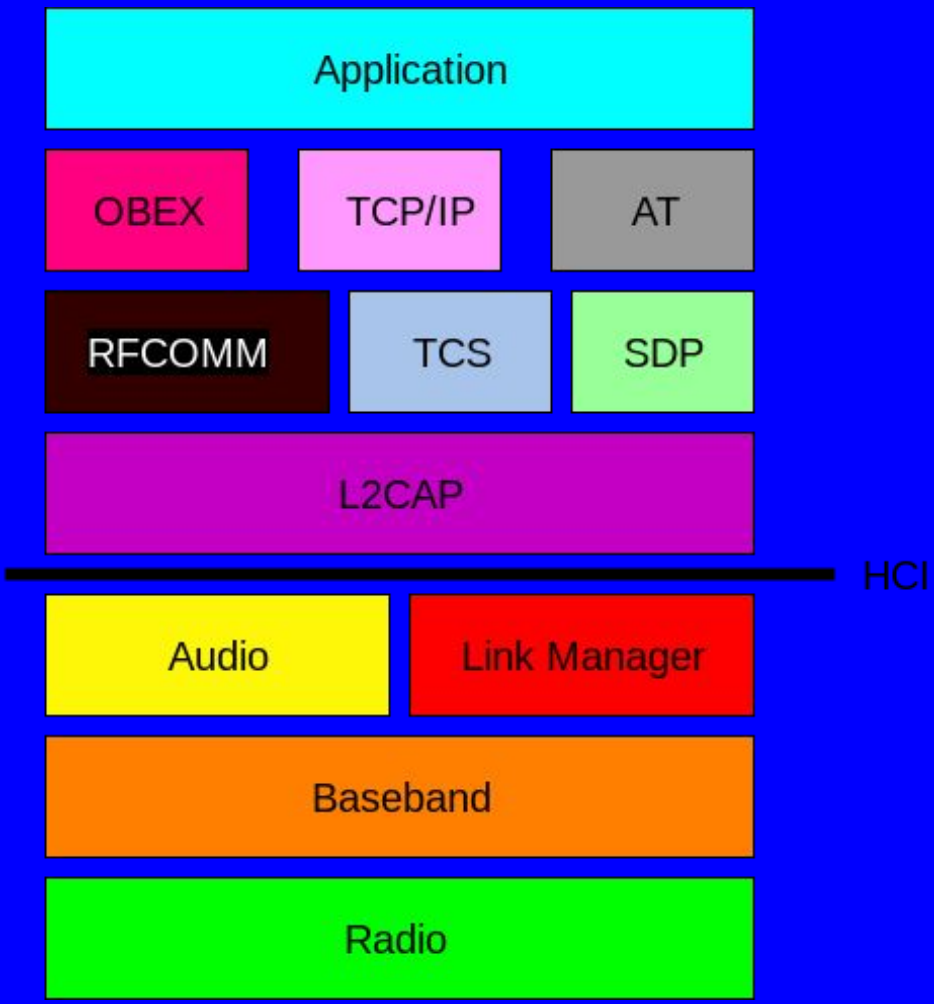
Bluetooth Security

An illustrated starter guide

Produced by James Ogden for SteelCon 2018


Application
OBEX
TCP/IP
AT
RFCOMM
TCS
SDP
L2CAP
HCI
Audio
Link Manager
Baseband
Radio

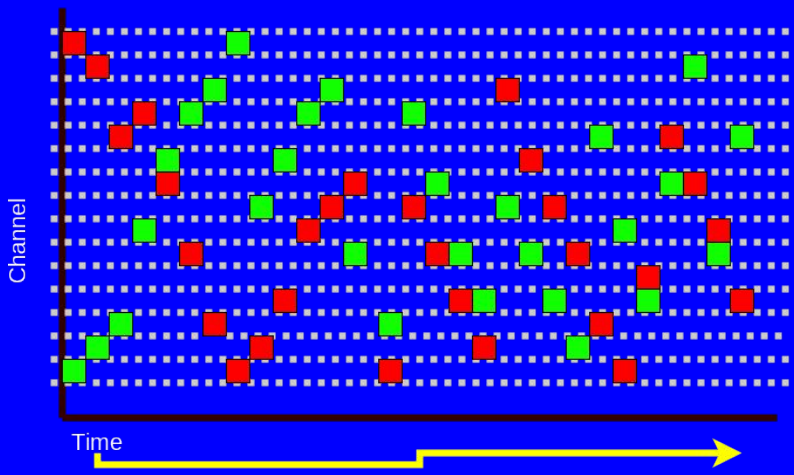
Bluetooth Classic (hereafter simply referred to as “Bluetooth”) is a wire replacement technology designed to take the place of cables such as those used for various kinds of audio, serial and network connections.

Work on technology that would later become Bluetooth began at Ericsson in 1989 however it wasn’t until 1994 the technology really started to take shape (this is the year that Bluetooth is generally considered to have been invented.) It took three more years for the name Bluetooth to be proposed in 1997. Despite early versions of the technology existing in the 1990’s it wasn’t until the early 2000’s that Bluetooth became a more popular and widespread technology - thanks in part to a number of devices such as wireless headsets and handsfree car kits becoming available.

Bluetooth has evolved through several different versions, increasing speed, security and convenience in a number of ways while trying to remain backwards compatible due to the number of devices already deployed that cannot be upgraded to newer versions of the standard.

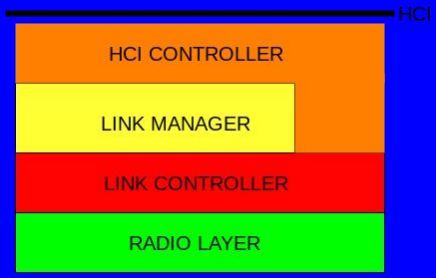
It’s quite straightforward to build software and devices that use Bluetooth - several low level components of the Bluetooth stack are usually implemented in hardware with HCI (host controller interface) being used to move data to/from the host system. In embedded systems you may find that some higher level constructs are also handled by the hardware, perhaps in the form of a module that can be integrated into a design without having to burden the host system with any Bluetooth specific code whatsoever.

There is much more that can be said about Bluetooth than I can fit in these few pages! The Bluetooth standards documents published by the Bluetooth SIG (Bluetooth Special Interest Group - the industry organisation for Bluetooth) run to thousands of pages and are extremely detailed. I hope, however, to give a brief overview and to answer some of the frequently asked questions that have been put to me in the past about Bluetooth.



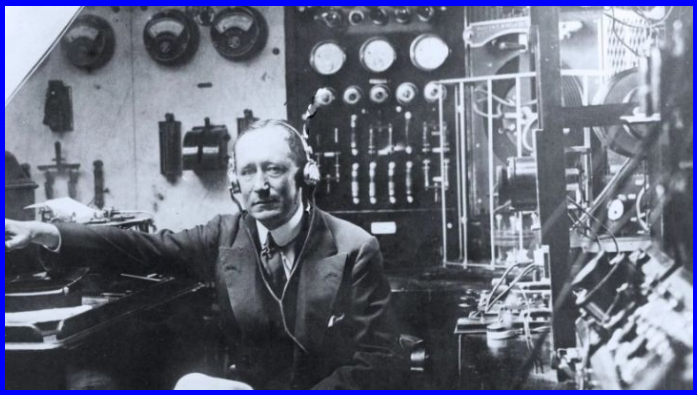
Bluetooth operates in the 2.4 GHz band where it occupies 79 channels from 2402 to 2480 MHz. Bluetooth uses frequency hopping which gives it a good degree of immunity to the interference found where it has to coexist with other users of the band such as CCTV cameras and various forms of remote control or telemetry.

Above is an illustration of frequency hopping as used in Bluetooth. 1600 hops per second is the base rate but this may reduce to 800 hops per second in AFH (adaptive frequency hopping) mode or increase to 3200 hops during the connecting phase. Devices take it in turns to transmit or receive (time division duplexing.)



The link manager is responsible for establishing the pattern of frequency hops and may agree with the other device to exclude certain channels if there is interference of some sort.

The lowest layers of the stack form a sort of ‘hindbrain’ in that their operation is autonomous and hidden from the host system.



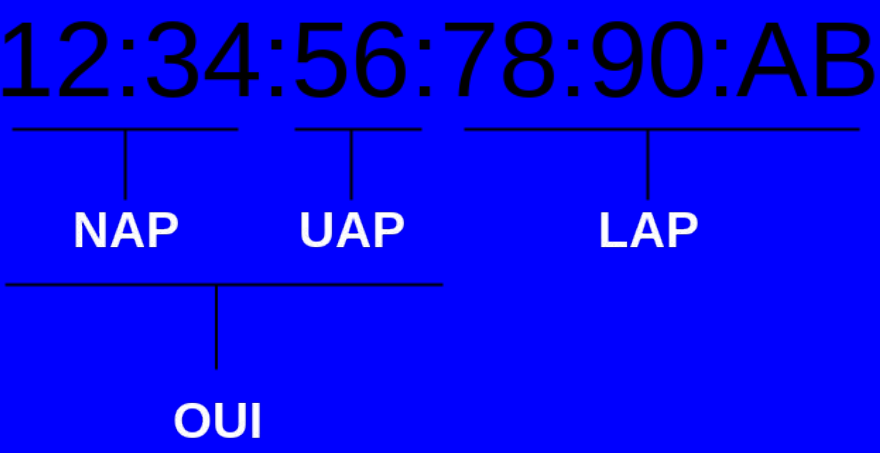
Class	Max Power	Range Meters	Use cases
Class 1	100 mW	~100	Industrial / Long range
Class 2	2.5 mW	~10	Mobile Phones / PC adapters
Class 3	1 mW	~1	Keyboards / Mice

Bluetooth devices may be manufactured to operate at one of these maximum power levels to achieve the typical ranges stated,

By far the most common type are Class 2 which provides a balance of good range without excessive power consumption (which is good when running from battery power.) Mobile phones and many laptiops contain Class 2 adapters.

Class 1 adapters can be used to achieve the greatest range. The Sena Parani UD-100 is an example of an adapter with both high power and good sensitivity and can easily achieve ten times the stated max range with a good antenna (even when the other side is a Class 2 device!)

The design of Bluetooth calls for devices to use the minimum transmit power possible within the range available to them and to cooperate with the other device via the link management protocol to agree what that is.


12:34:56:78:90:AB
NAP
UAP
LAP
OUI

This is the Bluetooth address, also known as the BD_ADDR. Similar in appearance to a MAC address and also 48 bits.

In the same way as a MAC address the first part is an OUI (organisationally unique identifier) which identifies the device manufacturer and is issued by the IEEE.

The BD_ADDR has three distinct parts:

NAP - The non-significant address part is the first two bytes of the OUI, as previously mentioned this is allocated by the IEEE and (along with the UAP) identifies the device manufacturer.

UAP - The upper address part. This is the remaining part, along with the NAP that forms the OUI. The UAP can be found using special hardware such as the Ubertooth One, however some extra computation is required so this isn't recovered in every case.

LAP - The lower address part is allocated by a device manufacturer ro a piece of hardware. Using the Ubertooth One or hardware with similar capabilities it can be easily discovered as it is transmitted as part of the access code with each frame. By waiting on a single Bluetooth channel we should see a nearby device several times each second and can recover it's LAP - this technique is used by the Ubertooth One.

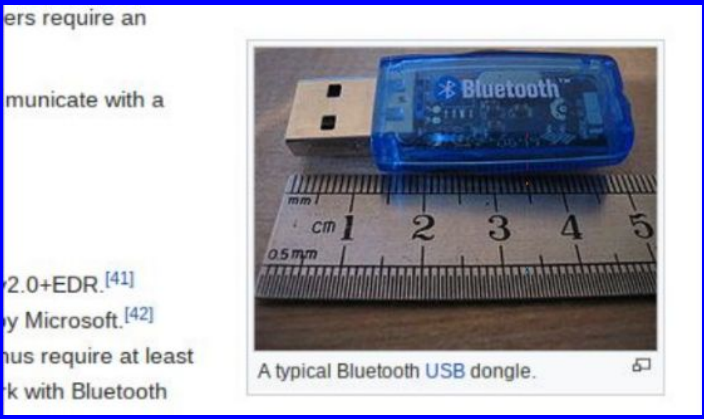
The LAP and UAP together are sometimes referred to as the SAP (significant address part) .

If an attacker can impersonate another device by manipulating their Bluetooth address this doesn't immediately compromise security as they will have no knowledge of the link keys that have previously been negotiated.

The BD_ADDR, along with the master devices clock (a 28 bit integer value incrementing 3200 times a second) is used to determine the frequency hopping pattern and also to seed various cryptographic algorithms.

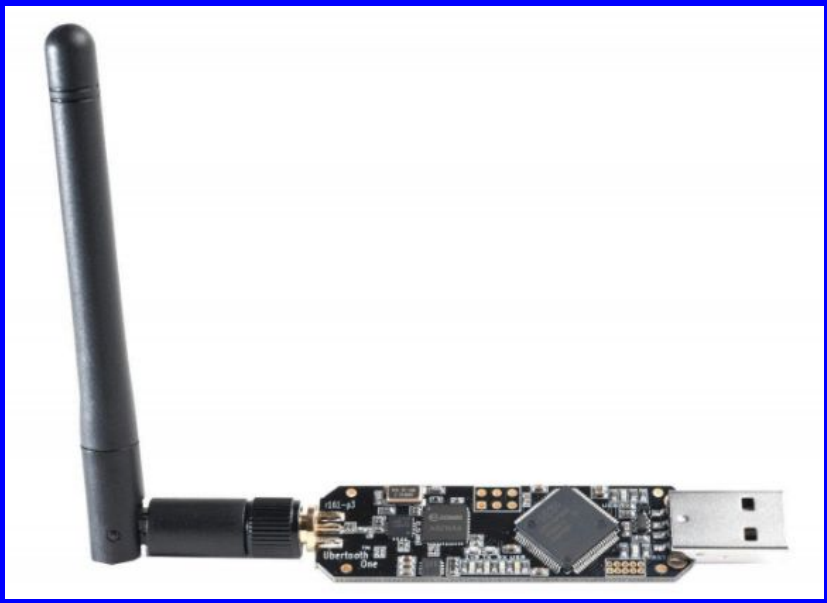
When exploring Bluetooth suitable hardware must be used. You may have a laptop with a Bluetooth adapter built in - this is OK for starting out with but you may wish to consider something that has a bit more power, extra configuration options or the ability to connect an external antenna.

You may wish to consider specialised hardware such as the Ubertooth One, not itself a full Bluetooth adapter but rather a device that enable some useful capabilities when experimenting. Some hardware options are:



A “typical Bluetooth USB dongle” is a good piece of hardware to start exploring Bluetooth with Some (such as the one pictured) have chips that allow the address of the device to be reconfigured - very useful.

This UD100 is a high quality, long range Bluetooth adapter with an RP-SMA connector allowing a variety of external antenna types to be connected depending on use-case. It has both high (Class 1) transmit power and good receive sensitivity.



The Ubertooth One from Great Scott Gadgets is a 2.4 GHz development platform. Very usefully we can use the ‘ubertooth scan’ command to discover nearby devices passively. The Ubertooth One can always recover the LAP, sometimes recover the UAP (via extra computation) but doesn’t recover the NAP. The lack of NAP isn’t a problem as many devices will accept a connection anyway if you set those bytes to a value such as “00:00”.

```
ubertooth scan
```

```
systime=1525849847 ch=56 LAP=9eb6ca err=2 clk100ns=1193755916 clk1=191000 s=-79 n=-55 snr=
systime=1525849847 ch=56 LAP=a7e066 err=0 clk100ns=1193823863 clk1=191012 s=-34 n=-55 snr=
systime=1525849847 ch=26 LAP=9e66ca err=0 clk100ns=1195265308 clk1=191242 s=-81 n=-55 snr=
systime=1525849847 ch=26 LAP=9e66ca err=0 clk100ns=1195465370 clk1=191274 s=-79 n=-55 snr=
systime=1525849848 ch=62 LAP=543d02 err=2 clk100ns=1199695661 clk1=191951 s=-85 n=-55 snr=

Scan results:
??:??:14:A7:E0:66       Moto G (5)
AFH map: 0xffffffffffffffff7f
```

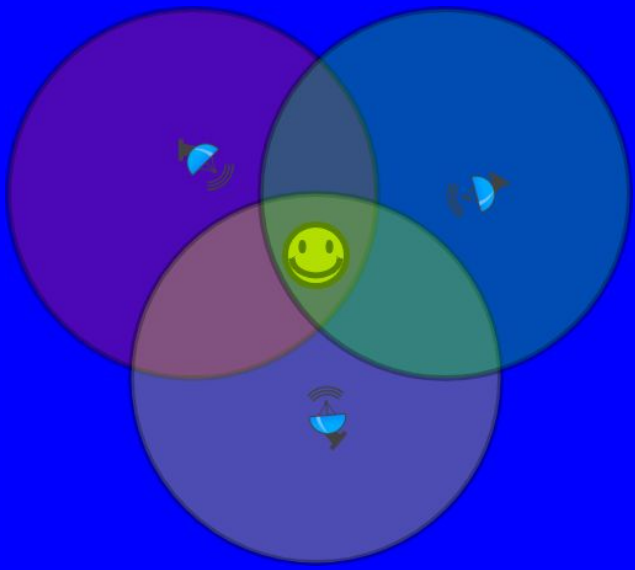
The maximum range of Bluetooth is influenced by a number of factors. In general communications at this frequency are best done ‘line of sight’ in an open area.

With devices where an external antenna may be used there are some considerations when choosing which type is best::

Antennas such as a simple dipole or ‘rubber duck’ are omnidirectional. These are good ‘all rounders’ that allow signals to be sent and received in all directions.

Antennas such as a cantenna or a yagi give a very high gain but are directional so greater care must be taken to aim them towards your target.

With careful aim, a high gain antenna and a good quality Bluetooth adapter ranges of a mile or more are not unheard of in open conditions!

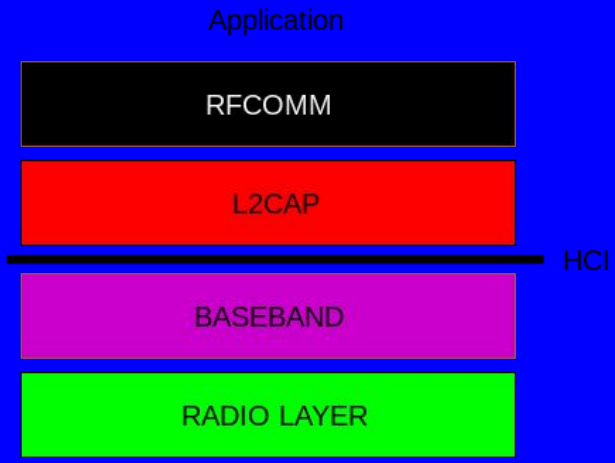


In the same way as 802.11 Wi-Fi Bluetooth may lend itself to tracking the route taken by devices as they move through a space. This may be used to track people in either an anonymous or a targeted fashion.

If multiple stations are able to compare received signal strength they can infer the location of the device. This could be used over large or small areas depending on the number of sensors deployed.

There are commercial solutions to track the movement of people through shopping centres or similar this way.

If this is a concern then the most effective countermeasure is to disable Bluetooth when not in use.



Bluetooth is used as a wire replacement protocol. Applications may be simple or complex. A simple serial (RS-232) style application may look like this. This type of application was the original use case conceived for Bluetooth where it was thought that it would be a low cost (target price of $5/unit) replacement for the wires used in keyboards and mice .

L2CAP (Logical Link Control and Adaptation Protocol) underpins everything built on Bluetooth. This is analogous to layer 2 / link layer in the OSI model.

L2CAP is used for multiplexing data between different higher layer protocols, segmenting and reassembling packets and can itself provide a functionality similar to ICMP echo packets via the use of the l2ping tool.

RFCOMM sits one level higher and provides us with RS-232 serial port emulation. Applications can be built on top of this and send/receive data as though connected by wire.

RFCOMM provides a simple, reliable data stream that can support up to 60 connected devices at a time.

RFCOMM is commonly used as a carrier for AT commands in telephony applications.

Because RFCOMM is such a simple protocol it is straightforward to write applications that use it and is also a simple matter to take applications that communicate via a serial port and convert them to use Bluetooth. .

BlueZ is the official Linux Bluetooth stack. Core components sit inside the Kernel with userspace applications built using libraries for languages such as C or Python.

BlueZ ships with a number of helpful utilities for investigating and debugging everything related to Bluetooth

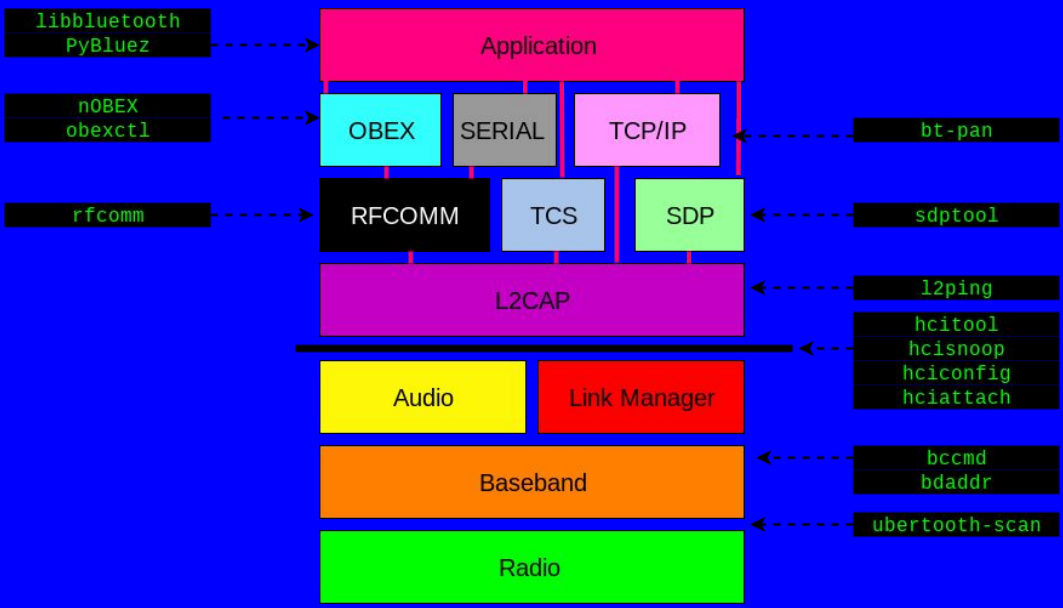
The BlueZ project can be found at http://http://www.bluez.org.

```
socket_desc = socket(AF_BLUETOOTH , SOCK_STREAM , BTPROTO_RFCOMM);
if (socket_desc == -1)
{
   printf("Failed to create socket\n");
   _exit(1);
}

loc_addr.rc_family = AF_BLUETOOTH;
str2ba(src, &loc_addr.rc_bdaddr);
loc_addr.rc_channel = (uint8_t) channel;

if (bind(socket_desc,(struct sockaddr *)&loc_addr , sizeof(loc_addr)) < 0)
{
    printf("Bind failed. Error");
    return 1;
}
listen(socket_desc , 1);
```

The code here shows the skeleton of a simple RFCOMM application in C. This is similar to any other socket code with calls to socket(), bind() and listen() but with the AF_BLUETOOTH socket family and the BTPROTO_RFCOMM protocol.

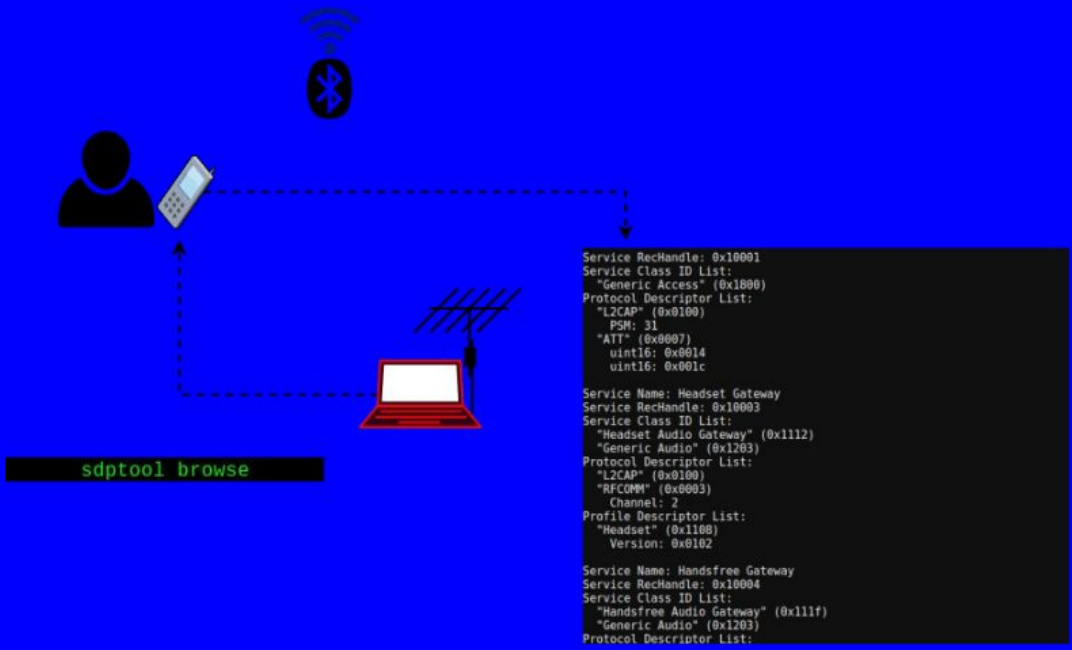
Once a connection is established data can be exchanged in the normal way.

In this way code designed for other socket types can be ported to provide those services over Bluetooth without much (if any) knowledge of how Bluetooth works ‘under the hood.’

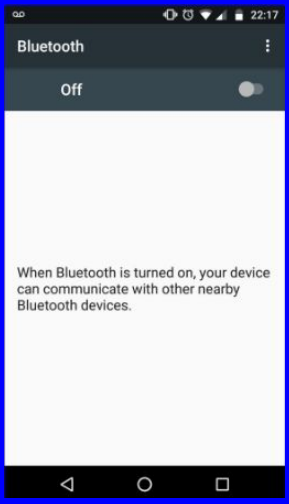


Here are some useful tools when working with Bluetooth on Linux.

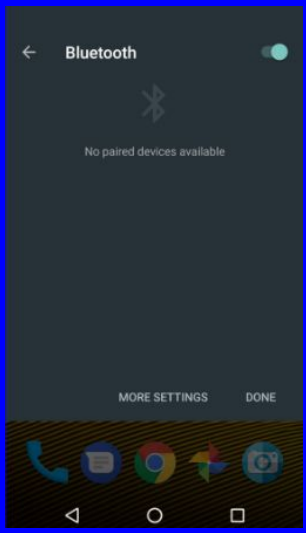
Hcitool name can be used to discover the name associated with a given bd_addr by sending an inquiry and receiving a response.



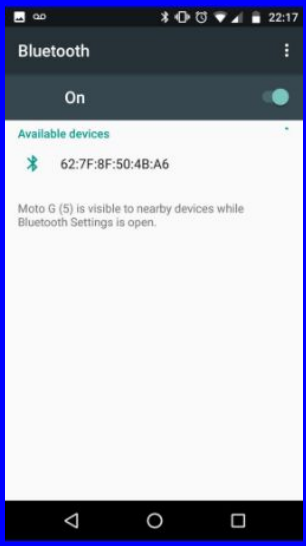
Sdptool browse can be used to discover the services available on a given bd_addr by sending an inquiry and receiving a response



In this state Bluetooth is OFF / DISABLED.



In this state the device may be connected to if it's address is known.



In this state the device is discoverable and it will respond to scans.

Devices may be in these states. known as “on” , “off” and “discoverable”

A device which is in the “off” state cannot send or receive data over Bluetooth. This is arguably the best way to ensure good security! It is, however, not very convenient if you want to actually use Bluetooth for something!

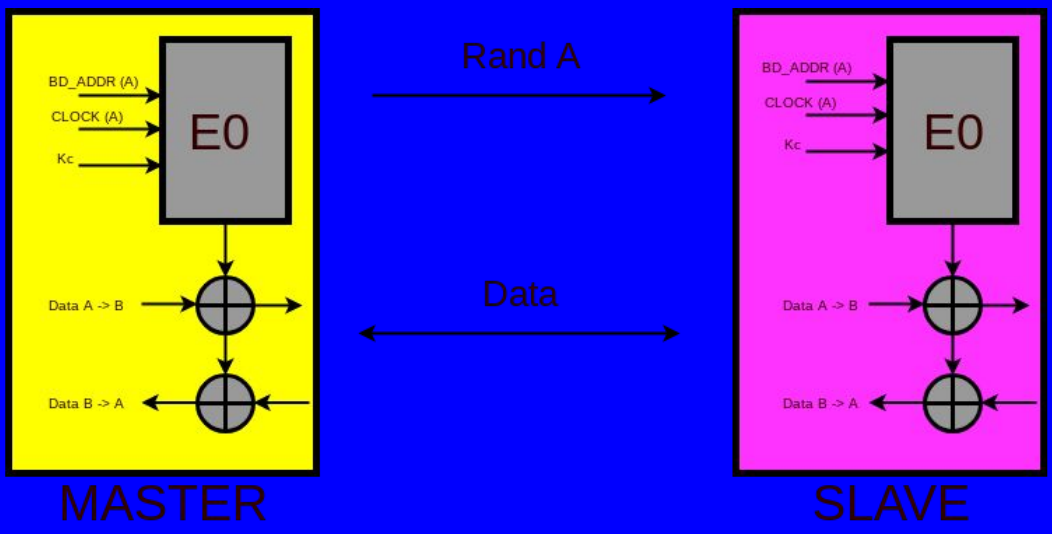
A device which is in the “on” state will respond to requests if it’s address is known (or can be discovered or guessed.) This is known as responding to a paging scan. In this state we can run command such as l2ping and sdptool browse against the device as well as initiating file transfer requests. Many devices are in this state all the time to allow Bluetooth to be used when required.

A device which is in the “discoverable” state will respond to any device nearby who broadcasts a request to find such devices. This is known as responding to an inquiry scan. Some devices are always in this state while others will only be like this when the relevant settings menu is open or for a period of time such as 30 or 60 seconds after activation of this mode.

If you scan for Bluetooth devices using a command like hcitool scan on Linux or by opening up the Bluetooth settings on your phone then the devices you see listed are those that are in the discoverable state. Although many devices only stay in this state for a brief period of time there are lots that will *always* respond (as you may see when running those commands in a busy area.)

Both the paging and inquiry scans take place over 32 channels with the master moving across 16 channels to transmit ID packets on and alternately listening for a response from the slave.

Slave devices may not listen at all times, even when capable of responding to a paging request, therefore the process of getting a response may take up to 4096 slots or 2.56 seconds. This is one reason Bluetooth device discovery may sometimes be perceived to be a slow process as we must ideally wait at least this time to ensure the device will or won’t respond.



This is the E0 encryption algorithm as used in Bluetooth to protect the data payload. It is a stream cipher with a variable key length (although 128 bit is the most common)

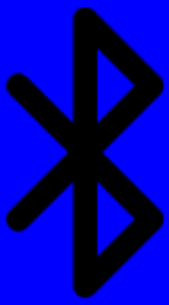
E0 encryption is divided into three phases:

1) Payload key generation
2) Keystream generation
3) Encoding

E0 has been the subject of cryptanalysis with several attacks and weaknesses being discovered that allow the key to be recovered in as few as 2^{38} computations under certain circumstances.

A well commented C implementation of E0 may be found here: https://github.com/adelmas/e0

01:23:45:67:89: AB



01:23:45:67:89: AC

Many manufacturers of devices containing both wifi and Bluetooth will use a scheme similar to this to allocate device addresses.

If we listen for 802.11 frames such as probe requests or data packets we can learn of addresses that might be tried.

This is a powerful technique that allows many devices to be discovered and requires no special hardware.

The process of ‘pairing’ sets up link keys that may be used to secure data in transit.

Two devices that have previously paired may also use these keys to confirm (via challenge response) that they are indeed talking to the same endpoint as before.

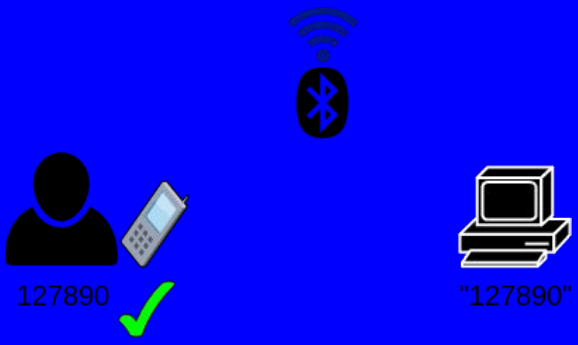
Although Bluetooth devices may initially pair by exchanging a four digit PIN (possibly even a well known one like 1234 or 0000) the link key that is generated is a 128 bit value.

When performing the challenge response the standard also mandates an exponentially increasing delay to prevent guessing of the link key.

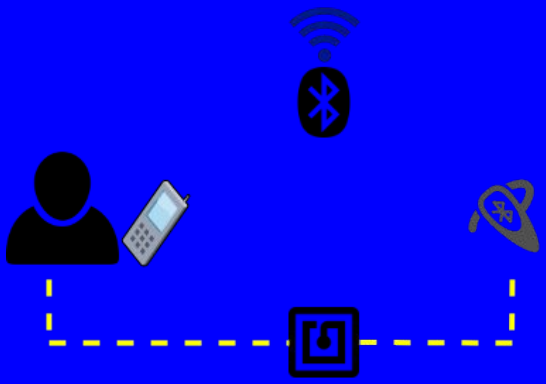
Pairing **is** required for some operations that are sensitive such as extracting call history, reading messages, streaming audio etc.

Pairing **isn’t** required to perform some operations such as offering the user a file transfer or enumerating which services are available.

In this scheme a numeric PIN is entered. Often this is 1234 or 0000 for ease of use. .



In this scheme a number is displayed which can be confirmed on the device.



In this scheme the pairing happens ‘out of band’ perhaps by tapping an NFC tag or similar.

Certain applications may necessitate the use of multiple profiles. Here we see the profiles used by some Bluetooth headphones. The audio stream is via A2DP (advanced audio distribution profile) with remote control of the music player using AVRCP (audio video remote control protocol.)

It's not unusual to find more complex applications such as hands free car kits that use several profiles in parallel to allow them to access SMS messages, phonebook entries etc in addition to the profile they use for hands free calling.

PBAP

OBEX

RFCOMM

L2CAP

HCI

BASEBAND

RADIO LAYER

MAP

OBEX

RFCOMM

L2CAP

HCI

BASEBAND

RADIO LAYER

@linuxthor