**Faculty of Engineering,**

**Electronics and Communications Engineering,**

**Communications Systems,**

**Lab1, Section 5.**

**17 April 2021**

**Ghaidaa Samir Mohamed 180**

**Omar Mohamed Mounir 163**

**Omar Nabil Fathy 165**

**Omar Nasr Mohamed Younis 166**

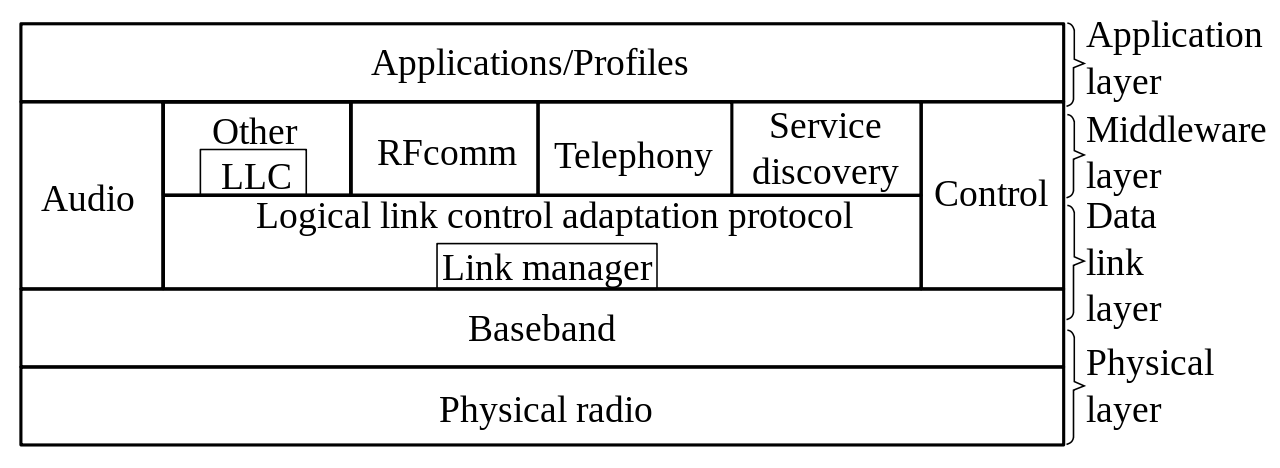
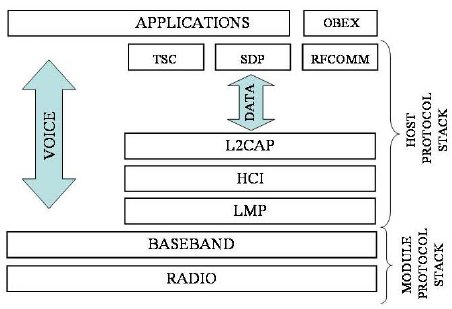
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Communication Systems

Lab1

# The Bluetooth Standard:

## The Bluetooth Stack



Bluetooth is defined as a layer protocol architecture consisting of core protocols, cable replacement protocols, telephony control protocols, and adopted protocols. Mandatory protocols for all Bluetooth stacks are LMP, L2CAP, and SDP. In addition, devices that communicate with Bluetooth almost universally can use these protocols: HCI and RFCOMM.

* **Link Manager (LMP)**

The Link Manager (LM) is the system that manages the establishment of the connection between devices. It is responsible for the establishment, authentication, and configuration of the link. The Link Manager locates other managers and communicates with them via the management protocol of the LMP link. To perform its function as a service provider, the LM uses the services included in the Link Controller (LC). The Link Manager Protocol basically consists of several PDUs (Protocol Data Units) that are sent from one device to another. The following is a list of supported services:

1. Transmission and reception of data.
2. Name request.
3. The request of the link addresses.
4. Establishment of the connection.
5. Authentication.
6. Negotiation of link mode and connection establishment.

* **Host Controller Interface (HCI)**

The Host Controller Interface provides a command interface for the controller and for the link manager, which allows access to the hardware status and control registers. This interface provides an access layer for all Bluetooth devices.

The HCI layer of the machine exchanges commands and data with the HCI firmware present in the Bluetooth device. One of the most important HCI tasks that must be performed is the automatic discovery of other Bluetooth devices that are within the coverage radius.

* **Logical Link Control and Adaptation Protocol (L2CAP)**

The Logical Link Control and Adaptation Protocol (L2CAP) is used to multiplex multiple logical connections between two devices using different higher-level protocols. Provides segmentation and reassembly of on-air packets.

1. In Basic mode, L2CAP provides packets with a payload configurable up to 64 kB, with 672 bytes as the default MTU, and 48 bytes as the minimum mandatory supported MTU.
2. In Retransmission and Flow Control modes, L2CAP can be configured either for isochronous data or reliable data per channel by performing retransmissions and CRC checks.

* **Service Discovery Protocol (SDP)**

The Service Discovery Protocol (SDP) allows a device to discover services offered by other devices and their associated parameters.

For example, when you use a mobile phone with a Bluetooth headset, the phone uses SDP to determine which Bluetooth profiles the headset can use (Headset Profile, Hands-Free Profile (HFP), Advanced Audio Distribution Profile (A2DP), etc.).

* **Radio Frequency Communications (RFCOMM)**

Radio Frequency Communications (RFCOMM) is a cable replacement protocol used for generating a virtual serial data stream. RFCOMM provides for binary data transport and emulates EIA-232 (formerly RS-232) control signals over the Bluetooth baseband layer, i.e., it is a serial port emulation.

Many Bluetooth applications use RFCOMM because of its widespread support and publicly available API on most operating systems. Additionally, applications that used a serial port to communicate can be quickly ported to use RFCOMM.

* **Bluetooth Network Encapsulation Protocol (BNEP)**

The Bluetooth Network Encapsulation Protocol (BNEP) is used for transferring another protocol stack's data via an L2CAP channel. Its main purpose is the transmission of IP packets in the Personal Area Networking Profile. BNEP performs a similar function to SNAP in Wireless LAN.

* **Object Exchange Protocol (OBEX)**

Session-layer protocol for the exchange of objects, providing a model for object and operation representation

* **Wireless Application Environment/Wireless Application Protocol (WAE/WAP)**

WAE specifies an application framework for wireless devices and WAP is an open standard to provide mobile users access to telephony and information services.

Baseband error correction

Depending on packet type, individual packets may be protected by error correction, either 1/3 rate forward error correction (FEC) or 2/3 rate. In addition, packets with CRC will be retransmitted until acknowledged by automatic repeat request (ARQ).

## Bluetooth Evolution

* **Bluetooth 5 (2016)**

A more robust version with extended battery life, BT 5 increased the outdoor transmission range from 50 to 200 meters. Location services are enhanced because it can convey more information prior to establishing a connection. The first smartphones to support BT 5 were the Galaxy S8 and iPhone 8 and X.

* **Bluetooth 4.2 (2014)**

Designed for the Internet of Things (IoT), BT 4.2 increased the payload size in the Bluetooth packet by 10x, dramatically lowering the overhead to yield 2.5 times more data. The low-power wireless personal area network (WPAN) version of IPv6 (6LoWPAN) is supported, which enables billions of devices to have a unique IP address. It also supports beacon privacy, which prevents retail shops from sensing a user's presence (see iBeacon). See 6LoWPAN, Internet of Things.

* **Bluetooth 4.1 (2013)**

More efficient data exchange and better co-existence with LTE frequencies. BT 4.1 maintains connections with less manual intervention, and devices can be both client and hub at the same time, enabling Bluetooth devices to communicate with each other. Prior to BT 4.1, devices transmitted to a hub either built into the computer or in a stand-alone dongle.

* **Bluetooth 4 (2010)**

Introduced low-power Bluetooth Low Energy, branded as "Bluetooth Smart." See Bluetooth LE.

* **Bluetooth 3 + HS (2009)**

Branded as Bluetooth 3.0 + HS (High Speed), it started the connection via Bluetooth but transmitted data over Wi-Fi.

* **Bluetooth 2.1 (2007)**

Secure Simple Pairing (SSP) was added to make pairing faster and more secure. Encryption was made mandatory, security was improved, and less power was used.

* **Bluetooth 2 (2004)**

Branded as Bluetooth 2.0 + EDR (Enhanced Data Rate), three-bit encoding (versus one) increased the data rate from 1 to 3 Mbps (in practice 2.1 Mbps). Interference handling was improved, and less power was used.

* **Bluetooth 1.2 (2003)**

BT 1.2 (Basic Data Rate) was the first widely used Bluetooth technology. Adaptive frequency hopping (AFH) helped avoid interference with Wi-Fi and other technologies in the same frequency. Pairing speed was improved.

* **Bluetooth 1.1 (2001)**

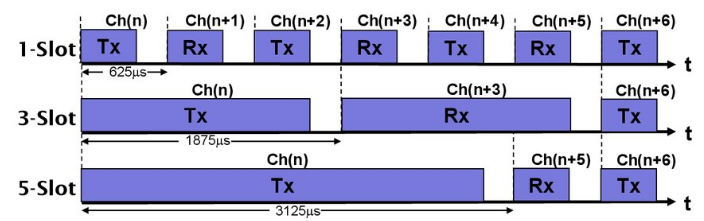
improvements to reliability and interoperability; mostly backward compatible but not 100%.

* **Bluetooth 1.0 and 1.0B (1999)**

The first Bluetooth specs. There were deployment issues that kept BT from gaining ground quickly.

# Specification Questions:

1. Bluetooth uses a range of 2.4 GHz. Bluetooth shares a 2.4 GHz frequency band with other wireless technologies as wifi, Bluetooth uses frequency hopping multiple access to minimize interference.
2. Bluetooth 1.2 uses GFSK (Gaussian frequency shift keying) modulation, bandwidth (B), time (T); BT = 0.5, Modulation index ranges between 0.28 & 0.35 .
3. Bluetooth 1.2 uses GFSK modulation and has a maximum data rate of 780 Kbit/sec, Bluetooth 2.0 uses DQPSK/8DPSK modulation and has a max data rate of 2178 Kbit/sec.
4. There are 79 channels available for Bluetooth, each with 1 MHZ bandwidth.
5. The time slot duration is 625 microseconds.



1. Bluetooth power classes:
   1. Class 1:
      1. Up to 100mw.
      2. USB Bluetooth sticks for pcs.
      3. Work over a distance of 100m.
   2. Class 2:
      1. Up to 2.5mw.
      2. Bluetooth adapter
   3. Class 3:
      1. Up to 1mw.
      2. Battery drivers (mobile phones).
      3. Work over a distance of 10m.
2. The master device is the device that initiates the connection, and it makes channel assessment to measure interference on each channel to select channels to be used in the hopping sequence.

Example: PC with a mouse, keyboard, speakers, etc. One master and up to 7 slaves.

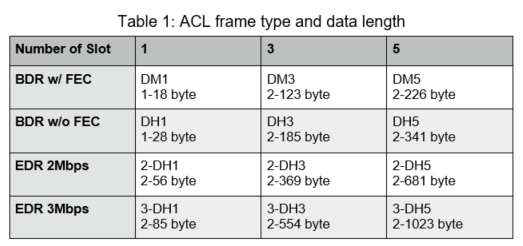
* 1. Bluetooth shares a 2.4 GHz frequency band with other wireless technologies,
  2. Bluetooth uses frequency hopping multiple access to minimize interference.
  3. There are 79 channels available for Bluetooth.
  4. The carrier frequency of transmission changes after each packet is transmitted.
  5. Each piconet has its own frequency hopping sequence, so they don’t interfere with each other.
  6. If packet length is 1Ts then hopping frequency = 1600 Hop/sec.
  7. If packet length is 5Ts then hopping frequency = 320 Hop/sec.

1. **ACL has two frame types,**
   1. DM (which provides Forward Error Correction: FEC).
   2. DH (which doesn’t provide FEC).

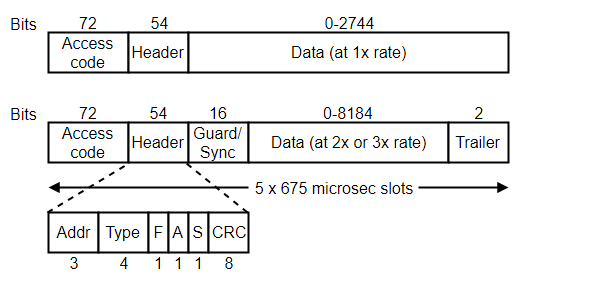
Forward error correction (FEC) is a method of obtaining error control in data transmission in which the source (transmitter) sends redundant data and the destination (receiver) recognizes only the portion of the data that contains no apparent errors.

In addition to these, the number is appended to these frame types like “DM1” or “DH3” to indicate how many 625 micro-seconds “slots” will be occupied.

In EDR (Enhanced Data Rate) introduced with Bluetooth 2.0, DH frame got double and triple the data rate mode denoted like “2-DH1” or “3-DH5”.



**Bluetooth Frame:** Access code(72bit) + Packet header(54bit) + Payload+ CRC(16bit).



1. The first field is an access code that usually identifies the master.
2. Next is a 54-bit header that contains several fields.
   1. The Address field identifies which of the active devices the frame is intended for.
   2. The Type field identifies the frame type (ACL, SCO, poll, or null), the type of error correction used in the data field, and how many slots long the frame is.
   3. The Flow bit (F) is set by a slave when its buffer is full and it can’t receive any more data.
   4. The Acknowledgement bit (A) adds an ACK to the frame,
   5. The Sequence bit (S) is used to number frames, to enable retransmission. Since the protocol is stop-and-wait, 1 bit is enough to number frames adequately.

The header is repeated three times. This is to add redundancy. Redundancy is important in Bluetooth, which runs in a noisy environment over low-powered devices.

1. There are various formats that are used for the data field. The basic-rate SCO frames are a simple example. The data field is always 240 bits.

There are three variants defined that set either 80, 160, or 240 bits of actual payload. The rest are used for error correction. In the 80-bit version, the bits are repeated three times for redundancy.

1. **Bluetooth Low Energy (BLE)** is a low-power wireless communication technology that can be used over a short distance to enable smart devices to communicate.

Some of the devices you interact with every day such as your smartphone, smartwatch, fitness tracker, wireless headphones, and computer are using BLE to create a seamless experience between your devices.

**Features**

1. **The lowest power consumption**

Everything from physical design to use models is designed to keep power consumption at a minimum. To reduce power consumption, a BLE device is kept in sleep mode most of the time. When an event occurs, the device wakes and a short message is transferred to a gateway, PC, or smartphone. Maximum/peak power consumption is less than 15 mA and the average power consumption is about 1 μA. The active power consumption is reduced to a tenth of the energy consumption of classic Bluetooth. In low duty cycle applications, a button cell battery could provide 5-10 years of reliable operation.

1. **Cost-efficient and compatible**

To offer compatibility with classic Bluetooth technology and cost efficiency for small battery-operated devices, there are two chipset types:

1. Dual-mode technology with both BLE and classic Bluetooth functionality
2. Stand-alone BLE technology optimized for small battery-operated devices with low cost and low power consumption as their focus
3. **Robustness, security, and reliability**

BLE technology uses the same adaptive frequency hopping (AFH) technology as classic Bluetooth technology. This enables BLE to achieve robust transmission in the ‘noisy’ RF environments found in the home, industrial, and medical applications. To minimize the cost and energy consumption of using AFH, BLE technology has reduced the number of channels to 40 2-MHz wide channels instead of the 79 1-MHz wide channels used with classic Bluetooth technology.

1. **Wireless coexistence**

Bluetooth technology, Wireless LAN, IEEE 802.15.4/ZigBee, and several proprietary radios use the license-free 2.4GHz Industrial Scientific Medical (ISM) band. With so many technologies sharing the same radio space, interference can decrease wireless performance (i.e., increasing latency and decreasing throughput) due to the need for error correction and retransmission. In demanding applications, interference can be reduced through frequency planning and special antenna design. As both classic Bluetooth technology and BLE technology utilize AFH, which minimizes interference with other radio technologies, Bluetooth transmission is robust and reliable.

1. **Connection range**

BLE technology has a slightly different modulation than classic Bluetooth technology. This modulation differentiation offers a range of up to 300 meters with a 10 dBm radio chipset (BLE maximum).

1. **Ease of use and integration**

A BLE piconet is typically based on a master connected to a number of slaves. A device is either a master or a slave, but never both. The master controls how often the slaves are allowed to communicate, and the slave only communicates by request from the master. A new feature BLE adds compared to classic Bluetooth technology is “advertising” functionality. With this feature, a device acting as a slave can announce that it has something to transmit to the master. An advertisement message can also include an event or a measurement value.

# Experiment Questions:

**Data rate = 3 Mbps**

**Coverage distance: up to 100m**

**Power class = class 1**

**Features**

* Fully-certified Class 1 Bluetooth 2.1 + EDR module
* Onboard embedded Bluetooth stack (no-host processor required)
* UART (SPP or HCI) and USB (HCI only) data connection hardware interfaces
* Supports Bluetooth data link to iPhone/iPad/iPod Touch
* Supports HID profile for making accessories such as keyboards, mouse, pointing devices
* Programmable low power modes
* Secure communications, 128-bit encryption
* Error correction for guaranteed packet delivery
* UART local and over-the-air RF configuration
* Auto-discovery/pairing requires no software configuration (instant cable replacement)
* Castellated SMT pads for easy and reliable PCB mounting
* The standard part number (RN41-I/RM) supports SPP and DUN profiles
* Available in multiple configurations: Apple-compatible firmware (RN41APL-I/RM), HCI mode (RN41HCI-I/RM), HID mode (RN41HID-I/RM), USB mode (RN41U-I/RM), and socket module (RN41SM-I/RM)
* Available without antenna (RN41N-I/RM)
* Bluetooth SIG qualified
  1. **Change Bluetooth module name**

**UART3\_Write\_Text("SN,BlueTooth-2"); // Name of device**

* 1. **Change Bluetooth module from slave to master**

**UART3\_Write\_Text("SM,1");**

**// Set mode (0 = slave, 1 = master, 2 = trigger, 3 = auto, 4 = DTR, 5 = ANY)**

# Simulations:

Script available here: [Comm\_systems\_labs](https://drive.google.com/drive/folders/19wZ9mpv4wvDhURv6lkEq6XSGoHsaBMaC)

**M = 8;**

**N = 10^6;**

**SNR = linspace(0,15,30);**

**seq = randi([0,M - 1],[1,N]);**

**z = dpskmod(seq, M);**

**for t = 1:length(SNR)**

**for c = 1:length(seq)**

**r(t, c) = z(c) + 0.5\*(1/sqrt(SNR(t)))\*(randn([1,1]) + 1i\*randn([1,1]));**

**end**

**end**

**for n = 1:length(SNR)**

**x(n, :) = dpskdemod(r(n, :),M);**

**end**

**[number, ratio] = biterr(x, seq);**

**subplot(1,2,1);**

**plot(SNR, ratio, 'LineWidth', 2)**

**title('BER')**

**subplot(1,2,2);**

**plot(SNR, number, 'LineWidth', 2)**

**title('Bit Error Number')**

