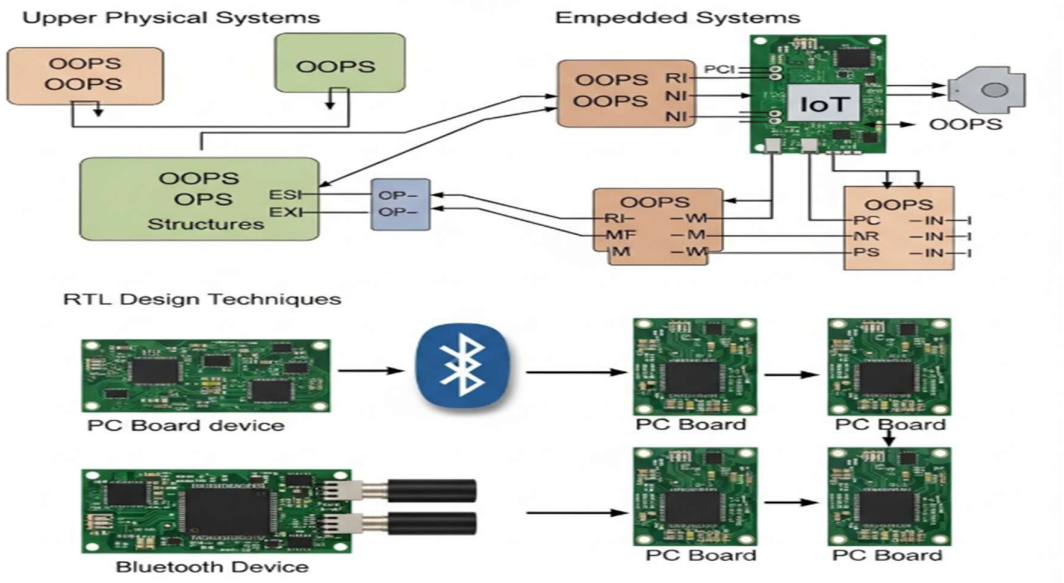
IOT (Internet of Things) and Upper Physical Systems (Basic Embedded concepts and RTL design on Bluetooth device)



Index: List of contents on this topic

IOT(Internet of Things) and Upper Physical Systems

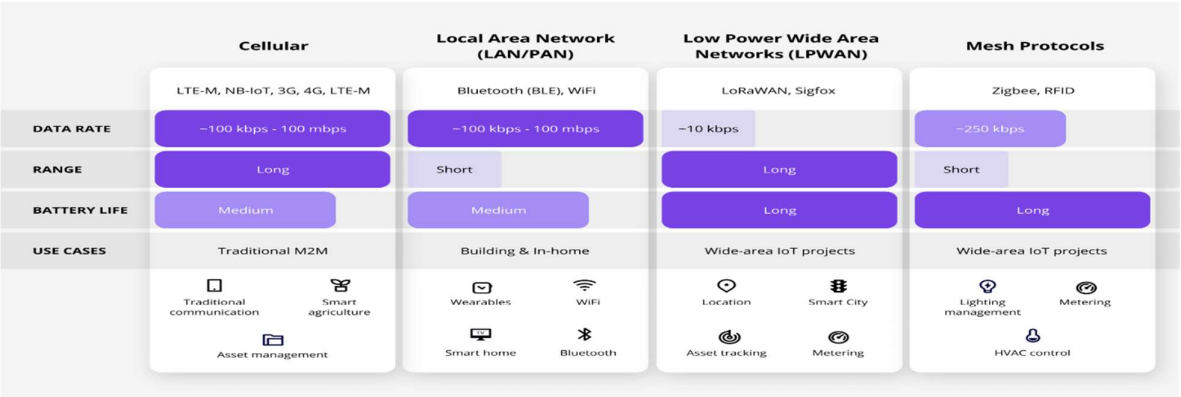
(Embedded OOPS & Data structures and RTL DESIGN)

Introduction: IOT: Internet of things, Upper physical system it tells us a broad range of things such as Integration of sensors, Actuators, Network and connectivity to physical objects and systems. Specially embedded system and data collection is the main sourse for IOT.

Different types of IOT'S : specified for different needs , let me discuss some of these

- **Customer IoT(CIOT):** Used for specific for house needs used for customers like smartwatches, personal health monitors and housing needs etc
- **Industrial Iot(IIOT):** Used for industrial sector either severce or manufacturing like energy, health, logistics , agricultural, vehicals(IOV) etc.
    - **Internet of vehicals(IOV)**: Through sencers connected to vehicals (V2V), vehical to cloud(V2C), vehical to infrastructure (V2I) these connections notably to communicate through IOT's to make life faster and progress.

- **Infrastructure IOT(IOTI):** Focuses on leveraging IoT to manage, develop new cities and aim to increase the quality of work like though who live in the cities, particularly Urban areas. Smart streetlights, secured cemeras, smart traffic management etc.
- **Internet of Medical Things(IOMT):** Specified for health care sector to connected medical and health system etc
- **Internet of Everting(IOE):** A broad classification of IOT to include people to connect through intelligence and process data so fast one place to another place with highly secure protocols.
- **Ambient IOT:** It is self powered and these divices operate through less consuming power and with out exter power sources , communication through less power sources and VLC(Visible light communication) etc. Mostly we use Infrastructure (IOTI)



We have a lot of IOT'S to make life easier while using network design, cloud networks, network data centers, and network virtualisation but foremost thing is that an Embedded system is the basic IOT device act as Brain or Memory for IOT for all modules to interface and communicate with given instructions at a basic design level.

Before working with IOT'S in the above list , how these were made memory level using Embedded system (OOPS AND DATA STRUCTURES) with RTL design, with basic PCB concepts.

Infrastructure IOT(IIOT) Bluetooth: Take some real examples of 'BLUETOOTH' device is the best example in mobile phones, we use predominantly for home applications or sharing messages, data, through mobiles

HC-05

RTL Logical design basic concepts:

BASIC LOGICAL GATES: According to the basic Bluetooth chip standard symbols for AND, OR, and NOT, NAND, NOR, EX-OR, EX-NOR logic gates remember these symbols representing the logical functions

1.AND Gate: In the device the circuit gives a high output (1) only if all its inputs are high, symbol A dot(.)

Figure-1:Logic Symbol of AND Gate

Input		Output
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

Figure-2: Truth Table of AND Gate

2.OR Gate: In the device the circuit that gives a high output (1) only if one or more inputs are high, symbol of OR operator A plus(+).

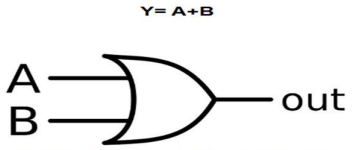


Figure-4:Logic Symbol of OR Gate

Input		Output
A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Figure-5: Truth Table of OR Gate

3.NOT Gate: The Not gate ia an electronic circuit that produces an inverted version of the input at it output.It is also know as inverter oou out is know as NOT, and the symbol is A' or A Bar on the top

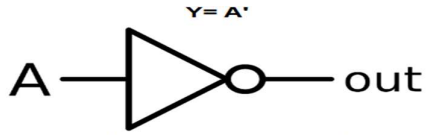


Figure-7:Logic Symbol of NOT Gate

Input	Output
A	Y
0	1
1	0

Figure-8:Truth Table of NOT Gate

4.NAND Gate: The outputs of all NAND GATES are high if any of the inputs are low. the symbol is an AND gate with a small circle on the output, small circle represents inversion, symbol is y = AB Bar

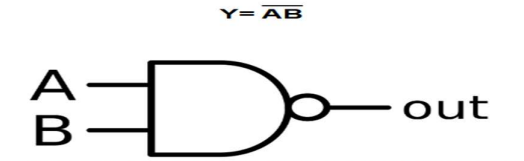


Figure-10:Logic Symbol of NAND Gate

Input	Input	Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Figure-11:Truth Table of NAND Gate

5.NOR Gate: NOT-OR is equal to OR gate fallowed by NOT,the outputs of all NOR gates are low if any of the inputs are high, The symbol is an OR gate with a small circle on the output, the small circle represents inversion, symbol is A+B Bar

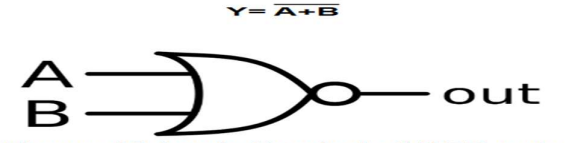


Figure-13:Logic Symbol of NOR gate

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

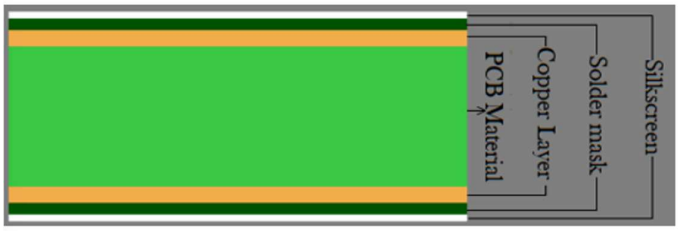
Figure-14:Truth Table of NOR gate

And the rest of the **EX-OR gate** symbol represents as encircled Y= A ⊕B and **EX-NOR gate** symbol Y= A $\bar{\oplus}$B Bar

SOME BASIC PCB concepts available on the Bluetooth, circuit board:

There are several types of PCB:(Printed circuit board) in which components combined together, as we show in the above bluetooth device lOT of circuits combined on this sheet , there are several types of PCB sheets which are applicable for different components they are

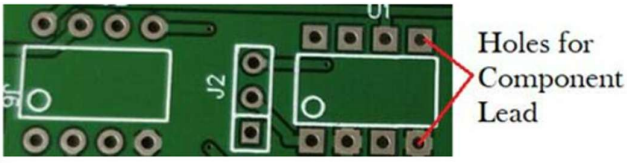
- Single-layer
- Double-layer PCB
- Multi-layer PCB
- Flexible PCB
- Aluminium-backed PCB
- Flex-rigid PCB



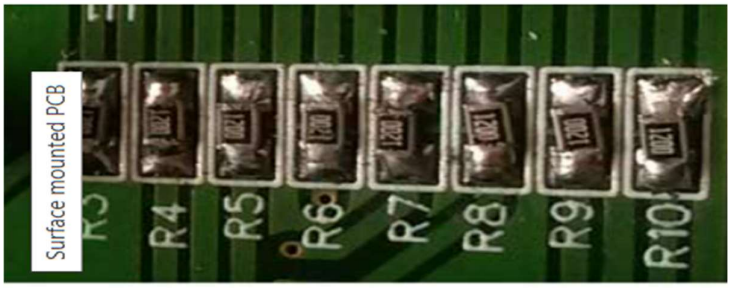
And subtypes of PCB according to design purpose

1. Through-hole PCB
2. Surface mounted PCB

And some of the devices Bluetooth devices placed on the PCB BOARD SURFACE component



Small lead components places in required to placed in order on the surface in one direction for a bluetooth device in one row, small macro or nano pins



Where certain pins are placed on the design part on the PCB board to execute level on different levels on the board, surface board to place, some certain design tools to design on PCB board to execute in a better manner

Embedded Systems OOPS concepts for Bluetooth Devices:

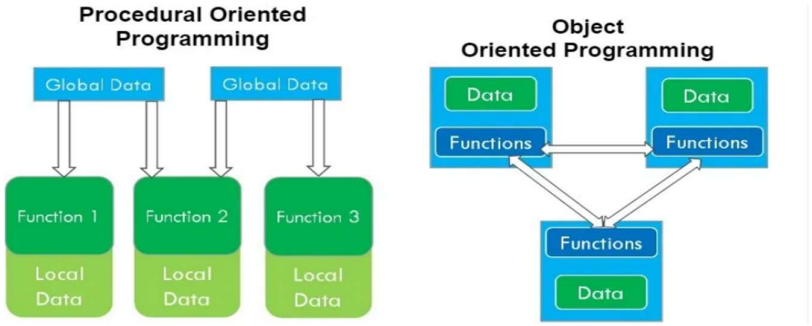
While using Embedded system OOPS concepts to approach Bluetooth devices in cell phones , using some Pseudo-logic with programming example.

Encapsulation: Refers objects internal details and expose only what is necessary for use(public, private, protected)

Abstraction: Involves simplifying complex objects by exposing only essential details and hiding unnecessary details

Inheritance: Inherit properties and methods from another class

Polymorphism: Ability to treat different classes to treate as instances of the same class through common interfaces and overloaded methods.



Using C Language is easy to access to memory, which allows programs to directly interact with hardware components of the Bluetooth chip (Pseudo-logic with programming example).

Bluetooth device might operate in a cell phone simple logic:

// Initialize the Bluetooth device

START BluetoothDevice

PowerOn_Hardware()

Load_Firmware()

Set_Status = "Ready"

END

// Scan for nearby devices

START ScanForDevices

IF BluetoothDevice.Status IS NOT "Ready" THEN

RETURN "Error: Device not ready"

ENDIF

```
Start_Broadcasting_Inquiry_Signal()
  LISTEN for responses for 10 seconds

  FOR EACH response received
    ADD device_name and device_address to DiscoveredDevicesList
  ENDFOR

  RETURN DiscoveredDevicesList
END

// Connect to a specific device
START ConnectToDevice(device_address)
  IF BluetoothDevice.Status IS "Connected" THEN
    RETURN "Error: Already connected"
  ENDIF

  Create_Connection_Request_To(device_address)
  WAIT for acceptance

  IF connection is accepted THEN
    BluetoothDevice.Status = "Connected"
    BluetoothDevice.ConnectedDevice = device_address
    RETURN "Success"
  ELSE
    RETURN "Failed to connect"
  ENDIF
END
```

DSA in Embedded on Bluetooth Systems:

A **stack** operates on a **Last-In, First-Out (LIFO)** principle. Think of it like a stack of plates; you add a new plate to the top and you take a plate from the top.

Pseudo-logic for a Stack Handling Bluetooth Events:

```
// Main loop is processing some data...
   PUSH "Process_Data_Task" onto CallStack
// A new connection request interrupt occurs
    INTERRUPT: NewConnectionRequest
PUSH "Handle_Connection_Task" onto CallStack
// Execute the connection handling logic
 Handle_Connection_Task()
 POP CallStack // "Handle_Connection_Task" is removed
 // After the interrupt, the system resumes the previous task
     RESUME "Process_Data_Task" // The top item on the stack POP CallStack //
"Process_Data_Task" is removed
```

Queues in Embedded Bluetooth Systems:

A queue operates on a First-In, First-Out (FIFO) principle. It's like a line at a ticket counter/ Initialize an empty queue for incoming data. **Pseudo logic**

```
CREATE IncomingDataQueue
// An event where data is received from Bluetooth
EVENT: DataPacketReceived(data)
  ENQUEUE(IncomingDataQueue, data) // Add new data to the back of the queue
// Main processing loop in the system
LOOP FOREVER
  IF IncomingDataQueue is NOT EMPTY THEN
    // Get the oldest data packet from the front of the queue
    CurrentPacket = DEQUEUE(IncomingDataQueue)
 // Process the packet (e.g., play audio, display notification)
    ProcessPacket(CurrentPacket)
  ENDIF
ENDLOOP
```

Reference books and notes: IOT'S

https://learn.microsoft.com/en-us/azure/iot/iot-introduction

https://www.ibm.com/cloud/internet-of-things

https://www.codeguru.com/iot/iot-development-platforms-ibm-watson-iot-overview/

RTL & PCB TOOLS PRACTICE ONLINE:

https://easyeda.com/

https://upverter.com/

https://circuitmaker.com/

RTL PRACTICE:

https://www.edaplayground.com/

https://hardwarebee.com/

https://www.geeksforgeeks.org/digital-electronics-logic-design-tutorials/

Embedded Systems for OOP Concepts with C:

https://embeddedartistry.com/

https://stackoverflow.com/questions/5329618/stateless-vs-stateful

https://www.coursera.org/learn/packt-embedded-systems-object-oriented-pro

DSA structures

https://www.geeksforgeeks.org/data-structures/

https://visualgo.net/en

https://www.hackerrank.com/