A close-up of a logo

Description automatically generated

# Name: Apurba Koirala

# Reg no: 22BCE3799

# Subject Code: BCSE305L

# Course Title: Embedded Systems

# Guided by: Dr. M. Narayanamoorthi

# Digital Assignment II

1. What is real time operating system? Compare and contrast the following with neat illustrative process table: RMS and EDF

Answer: A **Real-Time Operating System (RTOS)** is an operating system designed to serve real-time applications that process data within a strict deadline. It ensures predictable execution times for tasks and is used in embedded systems, robotics, aerospace, and industrial automation.

RTOS is classified into:

1. **Hard RTOS** – Missing a deadline leads to system failure (e.g., pacemakers, airbag systems).
2. **Soft RTOS** – Missing a deadline degrades performance but does not lead to failure (e.g., video streaming).

| **Criteria** | **Rate Monotonic Scheduling (RMS)** | **Earliest Deadline First (EDF)** |
| --- | --- | --- |
| **Scheduling Type** | Fixed Priority Scheduling | Dynamic Priority Scheduling |
| **Priority Assignment** | Shorter period → Higher priority | Closer deadline → Higher priority |
| **Preemptive?** | Yes | Yes |
| **Complexity** | Simple (Fixed priorities) | More complex (Dynamic priorities) |
| **Utilization Bound** | ≤ 69% for n tasks (Liu & Layland Bound) | Can reach 100% CPU utilization |
| **Optimality** | Not optimal; may lead to missed deadlines even if CPU is underloaded | Optimal for uniprocessor systems |
| **Overhead** | Low (no dynamic priority changes) | Higher (frequent recalculations) |
| **Use Case** | Industrial controllers, real-time OS where periodic tasks dominate | Multimedia, robotics, AI-based systems |

### **Illustrative Process Table**

Consider three tasks:

* **T1**: Period = 3ms, Execution Time = 1ms
* **T2**: Period = 6ms, Execution Time = 2ms
* **T3**: Period = 8ms, Execution Time = 2ms

#### **RMS Scheduling Table**

| **Time(ms)** | **Task Execution** |
| --- | --- |
| 0 - 1 | T1 runs |
| 1 - 3 | T2 runs |
| 3 - 4 | T1 runs |
| 4 - 6 | T2 runs |
| 6 - 7 | T1 runs |
| 7 - 8 | T3 runs |
| 8 - 9 | T1 runs |
| 9 - 10 | T3 runs |

#### **EDF Scheduling Table**

| **Time(ms)** | **Task Execution** |
| --- | --- |
| 0 - 1 | T1 runs |
| 1 - 3 | T2 runs |
| 3 - 4 | T1 runs |
| 4 - 6 | T2 runs |
| 6 - 7 | T1 runs |
| 7 - 8 | T3 runs |
| 8 - 9 | T1 runs |
| 9 - 10 | T3 runs |

EDF dynamically adjusts based on deadlines, whereas RMS sticks to fixed priorities.

1. Compare and contrast the following embedded networking protocols: I2C, CAN and Ethernet

Answer: Embedded networking protocols facilitate communication between microcontrollers, sensors, and other devices. The **Inter-Integrated Circuit (I2C), Controller Area Network (CAN), and Ethernet** are widely used in different embedded systems.

| **Protocol** | **Full Form** | **Type** | **Common Use Case** |
| --- | --- | --- | --- |
| **I2C** | Inter-Integrated Circuit | Serial, Multi-Master | Sensor interfacing, EEPROMs, LCDs |
| **CAN** | Controller Area Network | Serial, Multi-Master | Automotive, Industrial Automation |
| **Ethernet** | - | Packet-switched, Full Duplex | High-speed networking, IoT, Industrial Systems |

| **Feature** | **I2C** | **CAN** | **Ethernet** |
| --- | --- | --- | --- |
| **Topology** | Multi-Master, Multi-Slave | Multi-Master | Point-to-Point, Star, Bus |
| **Speed** | 100 kHz (Standard), 400 kHz (Fast), 3.4 MHz (High-speed) | 1 Mbps (Classic CAN), 5 Mbps (CAN FD) | 10 Mbps, 100 Mbps, 1 Gbps+ |
| **Number of Devices** | 127 (7-bit addressing) | 2048 | Virtually unlimited |
| **Communication Type** | Synchronous | Asynchronous | Asynchronous |
| **Message Prioritization** | No | Yes (Lower ID = Higher Priority) | No |
| **Error Handling** | Basic | Advanced (CRC, Retransmission) | Advanced (TCP/IP Error Handling) |
| **Power Consumption** | Low | Medium | High |
| **Cable Length** | Short (up to 1 meter at high speed) | Medium (40m @ 1 Mbps) | Long (100m per segment) |
| **Cost** | Low | Medium | High |

* **I2C:** Used in small embedded systems for sensor and peripheral communication.
* **CAN:** Used in automotive, industrial control, and medical devices for robust and real-time communication.
* **Ethernet:** Used in IoT, industrial networking, and internet-based applications requiring high-speed data transfer.

| **Protocol** | **Pros** | **Cons** |
| --- | --- | --- |
| **I2C** | Simple, Low power, Cost-effective | Limited speed and distance |
| **CAN** | Reliable, Fault-tolerant, Prioritization | Limited speed compared to Ethernet |
| **Ethernet** | High speed, Scalable, Supports Internet | Expensive, Higher power consumption |

1. Compare and contrast the following: Bluetooth, ZigBee and Wi-Fi

Answer:

Bluetooth, ZigBee, and Wi-Fi are widely used **wireless communication protocols** that serve different purposes in networking and embedded systems.

| **Protocol** | **Technology Type** | **Primary Use Case** |
| --- | --- | --- |
| **Bluetooth** | Short-range wireless | Personal device communication (headphones, smartwatches) |
| **ZigBee** | Low-power mesh network | Smart home automation, IoT sensors |
| **Wi-Fi** | High-speed wireless networking | Internet access, streaming, IoT |

| **Feature** | **Bluetooth** | **ZigBee** | **Wi-Fi** |
| --- | --- | --- | --- |
| **IEEE Standard** | 802.15.1 | 802.15.4 | 802.11 (a/b/g/n/ac/ax) |
| **Frequency** | 2.4 GHz | 2.4 GHz | 2.4 GHz / 5 GHz |
| **Range** | ~10 meters (Class 2) | ~10–100 meters (Mesh) | ~50–100 meters indoors, 300m outdoors |
| **Data Rate** | 1–3 Mbps (Bluetooth Classic), 2 Mbps (BLE) | 250 kbps | 11 Mbps (802.11b) – 9.6 Gbps (802.11ax) |
| **Power Consumption** | Medium | Low | High |
| **Network Type** | Point-to-Point (P2P) | Mesh Network | Infrastructure (Router/AP) |
| **Number of Devices** | 7 (Classic), Unlimited (BLE) | 65,000+ (Mesh) | 200+ (Typical Network) |
| **Security** | 128-bit AES, Secure Simple Pairing (SSP) | 128-bit AES | WPA2/WPA3 Encryption |
| **Latency** | Low (~5 ms) | Low (~30 ms) | Higher (~150 ms) |
| **Best For** | Personal Area Networks (PAN) | IoT, Smart Homes, Industrial Automation | High-speed internet & large-scale networking |

* **Bluetooth:** Wireless audio (headphones, speakers), wearables, file transfers.
* **ZigBee:** Smart home (lighting, locks, sensors), industrial IoT.
* **Wi-Fi:** Internet access, smart TVs, large-scale IoT, cloud computing.

| **Protocol** | **Pros** | **Cons** |
| --- | --- | --- |
| **Bluetooth** | Low power (BLE), simple pairing | Limited range and speed |
| **ZigBee** | Mesh networking, power-efficient | Low data rate |
| **Wi-Fi** | High-speed internet, widely available | High power consumption |

1. Identify the design challenges and the various functions to design drones with neat illustrative diagram and pseudocode?

Answer: Designing drones involves solving a variety of **engineering and technological challenges** to ensure safety, performance, and efficiency. Some of the **key challenges** include:

1. **Stability and Control:**
   * Drones need stable flight even in dynamic environments (wind, weather).
   * This involves designing robust flight controllers and stabilizing algorithms.
2. **Power Efficiency:**
   * Drones typically rely on batteries, which limits flight time.
   * Optimizing power consumption is critical for extending operational time.
3. **Communication and Networking:**
   * Reliable communication between the drone and its operator or autonomous systems is essential, especially in remote areas or under varying signal conditions.
4. **Sensor Integration:**
   * Drones need various sensors (GPS, IMUs, cameras, LiDAR, etc.) for navigation, collision avoidance, and environmental awareness.
   * Ensuring sensor fusion and correct data interpretation is crucial.
5. **Payload Capacity:**
   * Drones must balance their weight and payload capacity.
   * Overloading or uneven distribution of the payload can lead to instability or reduced flight time.
6. **Autonomy:**
   * Autonomous flight is a significant challenge, requiring real-time decision-making, path planning, obstacle avoidance, and adaptive behavior.
   * Integrating AI and machine learning algorithms for navigation, environment sensing, and decision-making.
7. **Regulations and Safety:**
   * Regulatory constraints limit drone operations in certain areas.
   * Designing fail-safes, geofencing, and emergency landing protocols ensures safe operation.
8. **Weather and Environmental Conditions:**
   * Drones must be designed to operate in a range of weather conditions (wind, rain, temperature).

**Functions to Design a Drone**

**1. Flight Control System (FCS)**

* The flight control system is responsible for stabilizing the drone in all directions.
* Functions:
  + **Gyroscope/Accelerometer Calibration**: Ensures accurate orientation.
  + **PID Controller**: Maintains stability by adjusting the motor speeds.
  + **Altitude Hold**: Maintains constant altitude through sensor feedback.

**2. Navigation and Path Planning**

* This system enables the drone to navigate autonomously from one point to another.
* Functions:
  + **GPS Integration**: Provides location and trajectory tracking.
  + **Obstacle Avoidance**: Uses sensors (LiDAR, camera) to detect and avoid obstacles.
  + **Path Optimization**: Determines the most efficient route using algorithms like A\* or Dijkstra’s.

**3. Power Management**

* Ensures that the drone is using power efficiently.
* Functions:
  + **Battery Monitoring**: Tracks battery voltage and usage to prevent power failure.
  + **Power Saving Modes**: Adjusts power consumption based on flight status.

**4. Communication System**

* Maintains communication with ground control, other drones, or remote stations.
* Functions:
  + **Telemetry**: Sends data about drone status (altitude, speed) to the operator.
  + **Real-time Video Streaming**: For surveillance or FPV (first-person view).
  + **Long-range Communication**: For remote or autonomous missions.

**5. Sensor Fusion and Data Processing**

* Combines inputs from various sensors for precise navigation and decision-making.
* Functions:
  + **Sensor Calibration**: Adjusts sensor readings to align them.
  + **Data Filtering**: Removes noise and improves data quality.
  + **Real-time Processing**: Ensures the drone responds immediately to changes in the environment.

**6. Safety Protocols**

* Ensures safe flight even in the case of a failure.
* Functions:
  + **Fail-safe Protocols**: Triggers emergency landing if critical systems fail.
  + **Geofencing**: Prevents the drone from flying into restricted zones.
  + **Return-to-Home**: Automatically brings the drone back to the launch point in case of signal loss.

**Illustrative Diagram of a Drone System**

Here’s a simple block diagram to visualize the functions involved in designing a drone:

sql

CopyEdit

+----------------------------+

| Flight Control |

| System (FCS) |

+----------------------------+

|

+-----------+-----------+

| |

+---------+ +------------+

| Navigation| | Power Management|

| & Path | | System (Battery)|

| Planning | +------------+

+---------+ |

+-----------+-----------+

| |

+--------------------------+ +-------------------+

| Communication System | | Sensor Fusion & |

| (Telemetry, Video, RF) | | Data Processing |

+--------------------------+ +-------------------+

|

+------------------+

| Safety |

| Protocols |

+------------------+

**Pseudocode for Basic Drone Flight Control**

Here's a pseudocode example for a **basic drone flight controller**:

pseudocode

CopyEdit

initialize DroneSystem

initialize sensors (gyroscope, accelerometer, barometer)

initialize motors

initialize GPS

initialize communication (telemetry, video)

initialize power management

while DroneSystem is active:

read sensor data (gyroscope, accelerometer, altitude)

calculate pitch, roll, yaw from gyroscope and accelerometer

adjust motor speeds based on PID control loop

if GPS data is available:

calculate desired trajectory

navigate towards target using path planning

if obstacle detected by sensors:

initiate obstacle avoidance

if power is low:

switch to power-saving mode

if emergency condition (low battery or signal loss):

activate fail-safe (Return to Home)

send telemetry data to operator

stream video (if applicable)

adjust flight based on operator commands

end