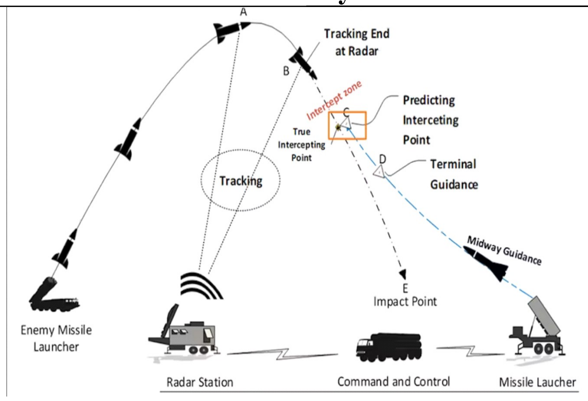
School of Computer Science and Engineering

Winter Semester 2023-24

Continuous Assessment Test – I1

Programme Name &Branch: BTech (CSE) **SLOT:A2+TA2**
Course Name & Code: **Embedded Systems – BCSE305L**
Faculty Name (s): All
Exam Duration: 90 Min. **Maximum Marks: 50**

<table>
<tr><td rowspan="2">Q. No.</td><td colspan="2">Note: The evaluation must be based on the individual answers. The keys are provided just for the reference</td></tr>
<tr><td>Questions</td><td>Keys</td></tr>
<tr><td>1.</td><td>Analyse meticulously the following scenario based on given criteria:-
➢ “A Smart anti-missile launching system”
Note: Position sensor, Motion sensor, Thermal sensor and relevant actuators may be part of the system and misguided enemy missile should be taken care appropriately
Given Criteria:
• Types of Events
• Functional correctness
• Timeliness</td><td>

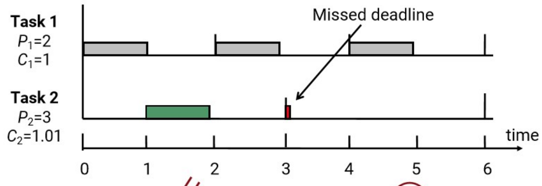
Type of events: (Position, Motion & Thermal sensors will be used)
• Detection
• Launching
• Tracking
• Navigation
• Interception
• Dynamic-path estimation (misguided enemy missile)
Functional Correctness:
• Estimation of parameters such as
✓ Speed
✓ Trajectory
✓ Angular velocity
✓ Altitude
✓ Point of interception
Timeliness:
• Events detection and estimation of parameters should be performed in real time with time stamping
• Relevant offset will be adjusted with received parameters to compensate any latency
• Estimated results should be communicated from the command and control center to the ant-missile launching Site
• Suitable anti-missile (heat seeking missile to counter the threat of misguided enemy missile) will be launched and guided precisely to intercept and destroy the incoming enemy missile
The best example of an extremely complicated real-time embedded system is a smart anti-missile launching system. It may comprises of multiple sub-system and require interconnection among these subsystem. In order to have machine-to-machine or device-to-device interaction, latency must be reduced to a level that is acceptable. Hence, from idea to implementation, the aforementioned requirements are crucial.</td></tr>
</table>

2. **Using** two sample data sets of your choice, **prove** that how do the rate monotonic scheduling technique:-

- Will fail to schedule a given set of tasks and
- Will be successful in scheduling a given set of tasks.

Find out the causes for the scheduling success and failure. **Provide** a remedy for the scheduling failure and enough justifications.

RMS-will fail to schedule a given set of tasks



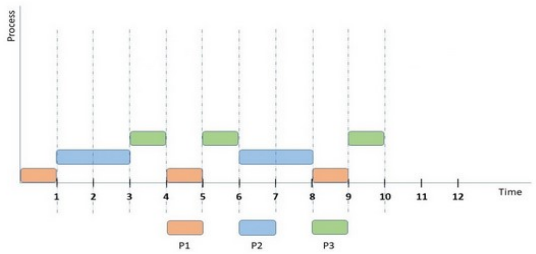
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 0.833$$

Unschedulable

- Question: Is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed?

RMS- will be successful in scheduling a given set of tasks

Process	Execution time(E)	Period (P)
P1	1	4
P2	2	6
P3	3	12



3. **Evaluate** how do the following parameters affect the schedulability of real time tasks:-

- Arrival Time
- Current Time / Scheduling point
- Execution Time
- Rate or Period
- Deadline

Suggest an optimal scheduling scheme using any three relevant parameters as mentioned above and **apply** it for the following dataset:-

Task	Arrival Time	Execution Time	Period	Deadline
T1	0	5	20	9
T2	0	4	15	6
T3	0	4	20	12

Note: Consider current time as per the scheduling points. **Illustrate** the task time-line graph for **at least three cycles**.

Arrival time – determines the task priority and the overall schedulability of any given set of real-time tasks.
Current Time & Execution Time-acts as a driving factor in implementing hybrid real time scheduler.
It also affects the balancing of task loads.
Rate or period – faster the rate makes higher the task priority. Any system having more tasks with faster rate may lead to deadline misses, task starvation and decreased processor utilization
Deadline-based on time, context, content, parametric minimization and maximization. It drives the classification of real time embedded systems into hard, fair and soft. It serves as a deciding factor when choosing an appropriate real-time scheduler.
Case-1: (execution time, Current time & deadline)
T3 will miss the deadline.
Case-2: (execution time, current time & period)
Every task will be successfully scheduled.
Note: Illustration must be done for 3 cycles.

4. **Construct** FSM model for the given scenario: -

- ***"Pick and Place Robot"***

Specification:

1. **Ambience: -** Factory floor with racks on the walls and two robots per floor.
2. Rack compartments are of different size.
3. Different type of objects will be supplied via conveyor belt.
4. Mobile robot with 360^0 scanning capability.

Requirements:

1. Optimal rack **space utilization**
2. **Collision** avoidance to be incorporated
3. Relevant **states, events and actions** to be considered

Entity

- Factory floor
- Types of objects
- Size of rack and compartments
- Mobile robots

Objects

- Number of objects
- Types
- Size-Space

Racks and compartments

- Empty
- Full
- Partial
- Large
- Medium

Illustrate the CDFG for the above scenario.

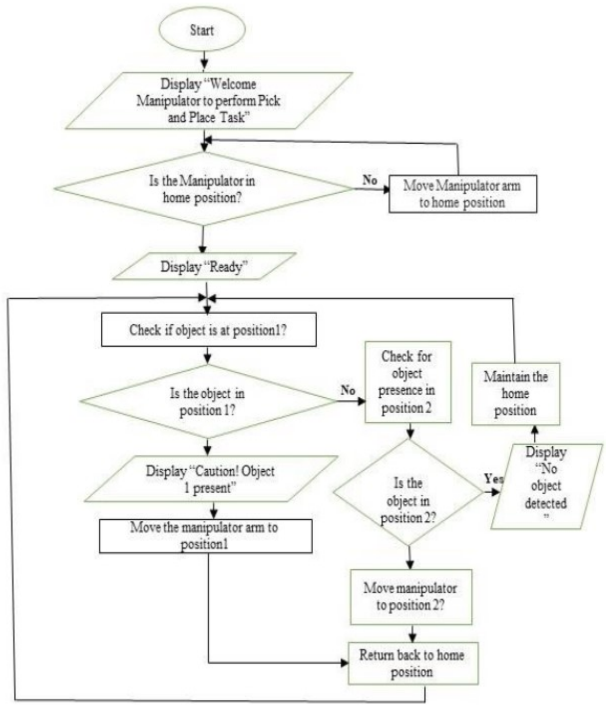
- Small

Robot

- Idle / home
- move-forward
- move-backward
- turn-left
- turn-right
- arm-rotate(360^0)
- scan / search
- pick / select
- sort
- place
- halt

From State	Event/Action	To State
Idle/Home	Pick-object/Search-rack/Select-compartment/ Place-object/ Rack-empty-partial/ Rack-full/Compartment-empty-partial/Compartment-full/obstacle-true/obstacle-false/Evade or any other	Move-forward/ Move-backward/ Turn-left/ Turn-right/Evade/Scan
Move-forward/ Move-backward/ Turn-left/ Turn-right/Evade/Scan	Rack-empty-partial/ Rack-full/Compartment-empty-partial/Compartment-full/obstacle-true/obstacle-false/Evade or any other	Halt

Partial-CDFG



5.	**Investigate** the challenges and issues faced by embedded system programmer. **Using the findings** from your investigation optimize the code snippet as given below and **provide** appropriate justifications. *int x;* *for (int i=0; i<x+300; i++)* *{* *for (int i=0; i<300; i++)* *Temperature[i] = i ^ 5;* *Moisture[i] = i ^ 3;* *x=1;* *If (x == 0)* *for(k=0; k<20;k++)* *printf("I am always a Star");* *}*	**Challenges and Issues:** • Unique requirements • Meeting deadlines • Logic optimization • Functional customization • Time to market • Hardware-Software integration • Selection of programming tools • Minimization of space & time complexities • Code portability issues • Data criticality • Task dependency • Task & Thread-level parallelism **Optimization Techniques** • Code motion • Strength reduction • Dead code elimination • Loop unrolling • Array access using pointer • Loop fusion