EMBEDDED SYSTEMS
DIGITAL ASSIGNMENT 2

COURSE CODE: BCSE305L
SLOT: A1 + TA1

Design the embedded systems for two hard real time systems and 2 soft real time systems and justify your aspects.

1) Design the embedded systems for two hard real time systems and two soft real time systems. Justify your aspects.

Hard Real time Systems

① Aircraft Flight Control System

→ Real-time operating system (RTOS): Choose a highly deterministic RTOS with minimal interrupt latency, such as VxWorks or FreeRTOS. This ensures that critical flight control tasks are executed within strict timing constraints.

→ Hardware Redundancy: Employ redundant sensors and control actuators along with a fault-tolerant architecture to ensure system reliability and fault recovery in case of hardware failures.

→ Scheduling: Utilize static priority-based scheduling algorithms to guarantee that critical tasks such as altitude and heading control are given highest priority, ensuring timely execution and response to external stimuli.

__/__/____

→ Validation and verification : Extensive testing and simulation are crucial to verify the correctness and safety of the system. This includes both functional testing and analysis of worst-case execution times to ensure compliance with real time requirements.

② Medical Life Support System

→ Fault Tolerance : Implement redundancy in both hardware and software components to mitigate risks associated with failures, ensuring continuous operation even in the presence of faults.

→ Safety Certification : Comply with medical device regulations and standards such as ISO 13485 and IEC 60601 to ensure the safety and efficacy of the life support system.

→ Monitoring and Alarming : Incorporate real time monitoring of patient vital signs and system parameters, with immediate alarming and intervention capabilities in case of abnormal conditions.

→ Battery Backup: Include battery backup or uninterruptible power supplies (UPS) to ensure continuous operation during power outages, critical for life support systems where downtime can be life-threatening.

Soft Real-Time Systems

① Online Video Streaming Service

→ Quality of Service (QoS) Management: Employ adaptive bitrate streaming algorithms to dynamically adjust video quality based on network conditions, ensuring a smooth viewing experience for users.

→ Buffering: Implement buffering mechanisms to compensate for network fluctuations and latency, minimizing interruptions and buffering delays during playback.

→ Content Delivery Network (CDN) Integration: Utilize CDN services to distribute content closer to end-users, reducing latency and improving overall streaming performance.

→ User Analytics: Collect and analyze user engagement data in real-time to personalize content recommendations and improve the overall streaming experience.

__/__/____

(2) Automated Home Security System

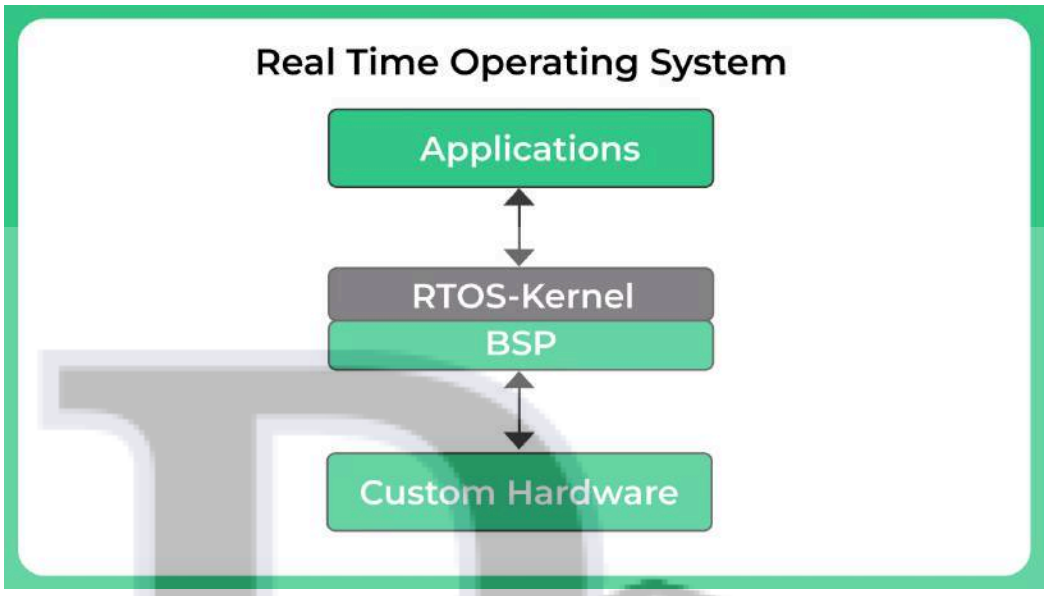
→ Sensor Fusion: Integrate various sensors such as motion detectors, door/window sensors, and cameras to provide comprehensive coverage and accurate detection of security threats.

→ Mobile Alerts: Implement real-time push notifications to homeowners' mobile devices in case of security breaches or anomalies detected by the system.

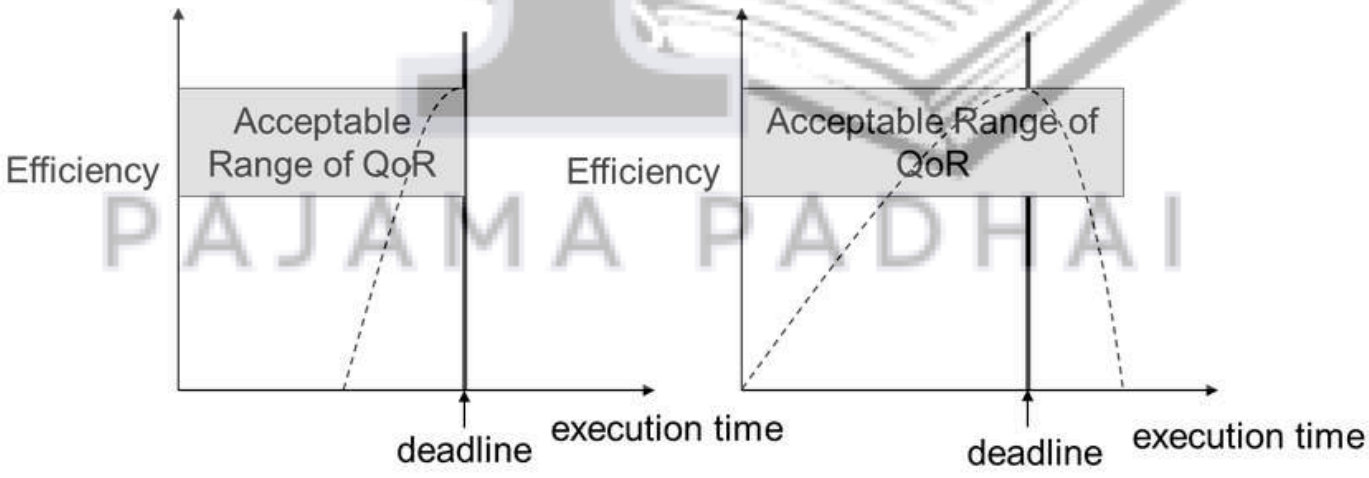
→ Remote Monitoring and Control: Enable users to remotely monitor and control the security system via mobile apps or web interfaces, providing flexibility and convenience.

→ Integration with Emergency Services: Integrate with emergency services such as police or fire departments to enable automated dispatch in case of emergencies, reducing response times and improving overall security effectiveness.

These designs prioritize aspects such as determinism, fault tolerance, safety and user experience based on the specific requirements and constraints of each system, whether they are hard or soft real-time.

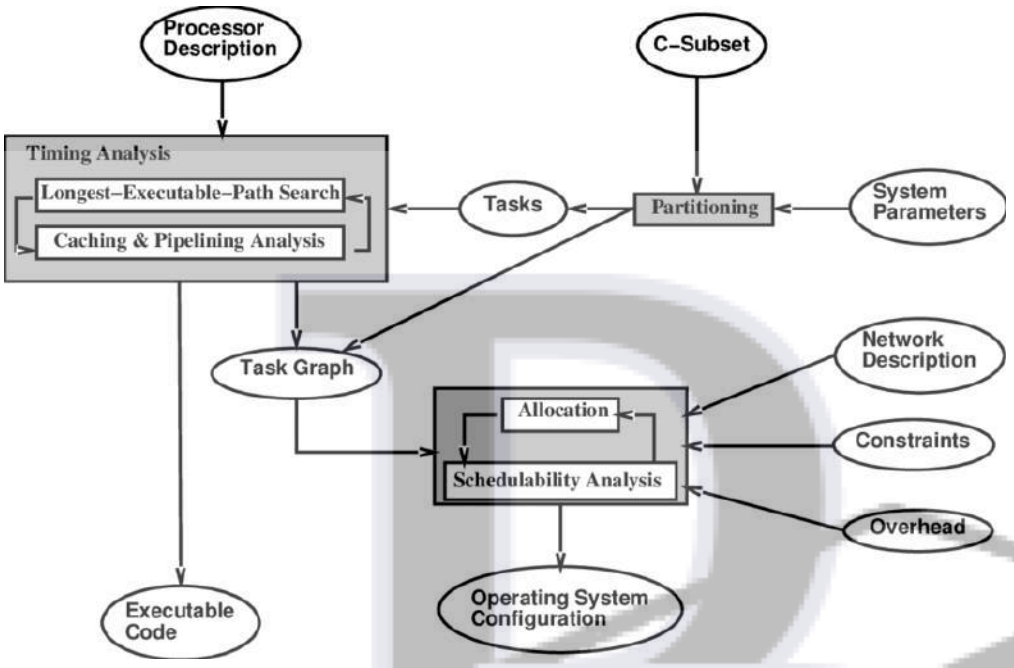


Characteristic	Hard Real Time	Soft Real Time
Deadlines	hard	soft
Pacing	environment	computer
Peak-Load Perform.	predictable	degraded
Error Detection	system	user
Safety	critical	non-critical
Redundancy	active	standby
Time Granularity	millisecond	second
Data Files	small/medium	large
Data Integrity	short term	long term

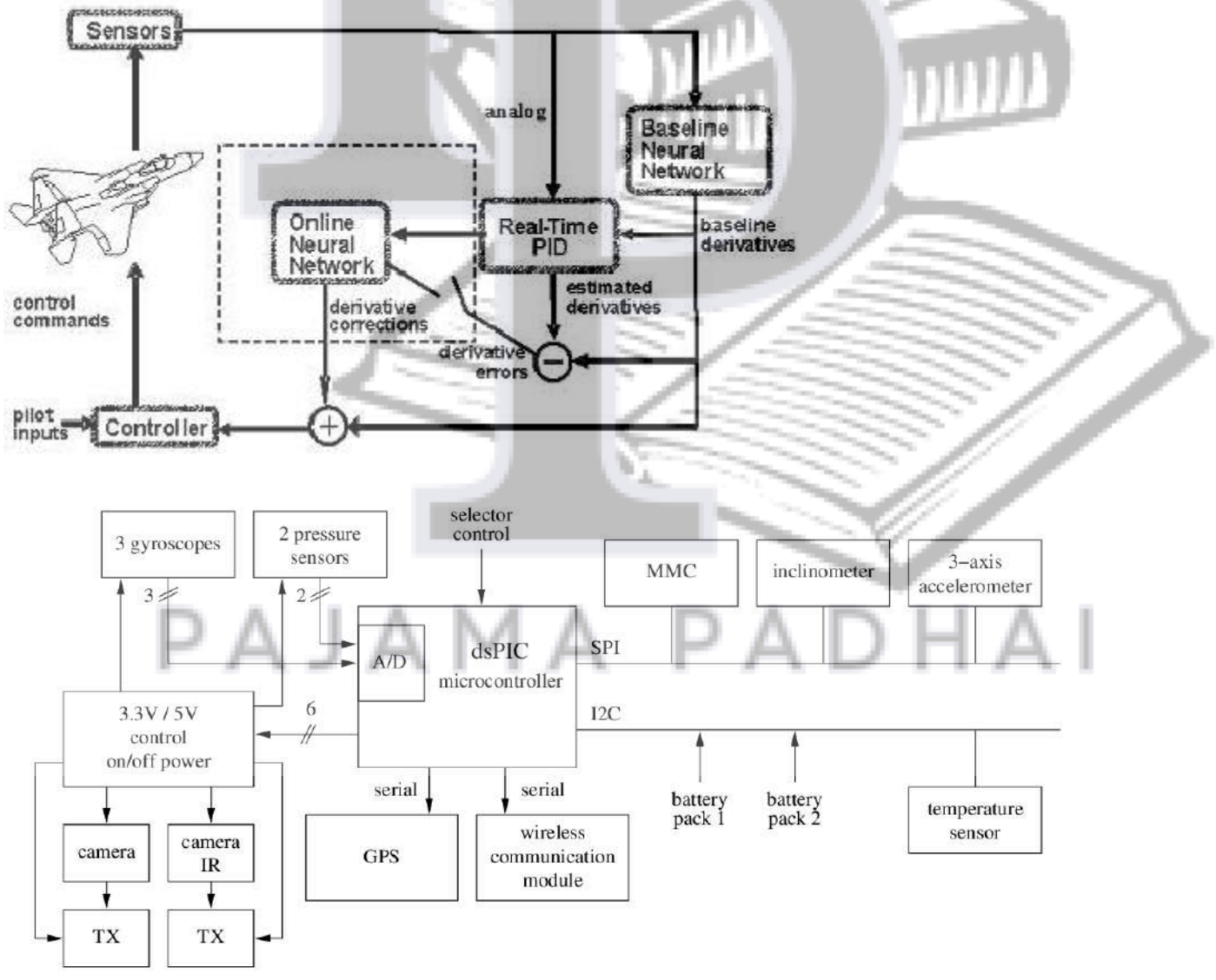


(a) hard real-time system (b) soft real-time system

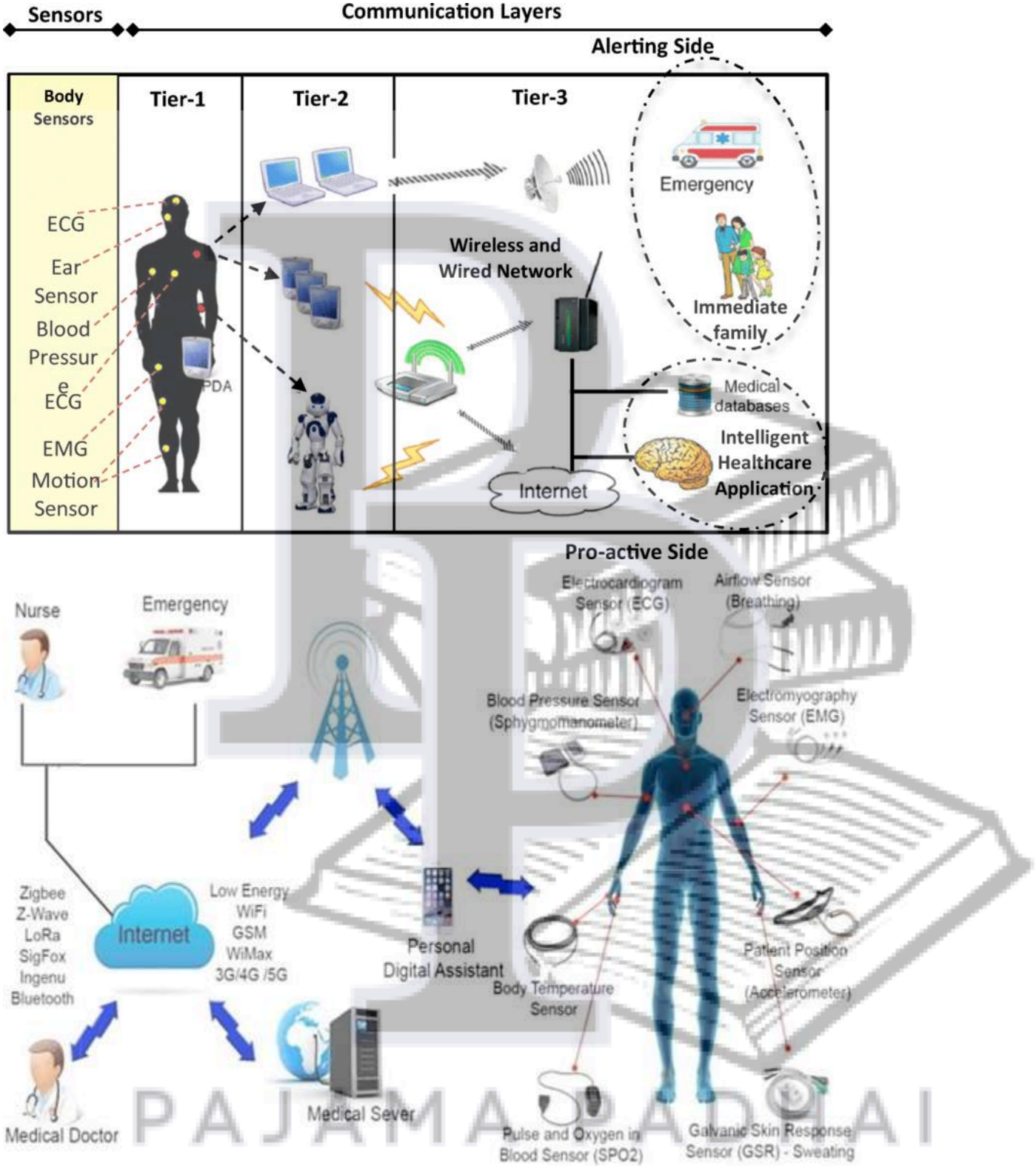
Hard Real-Time Systems:



Aircraft Flight Control System

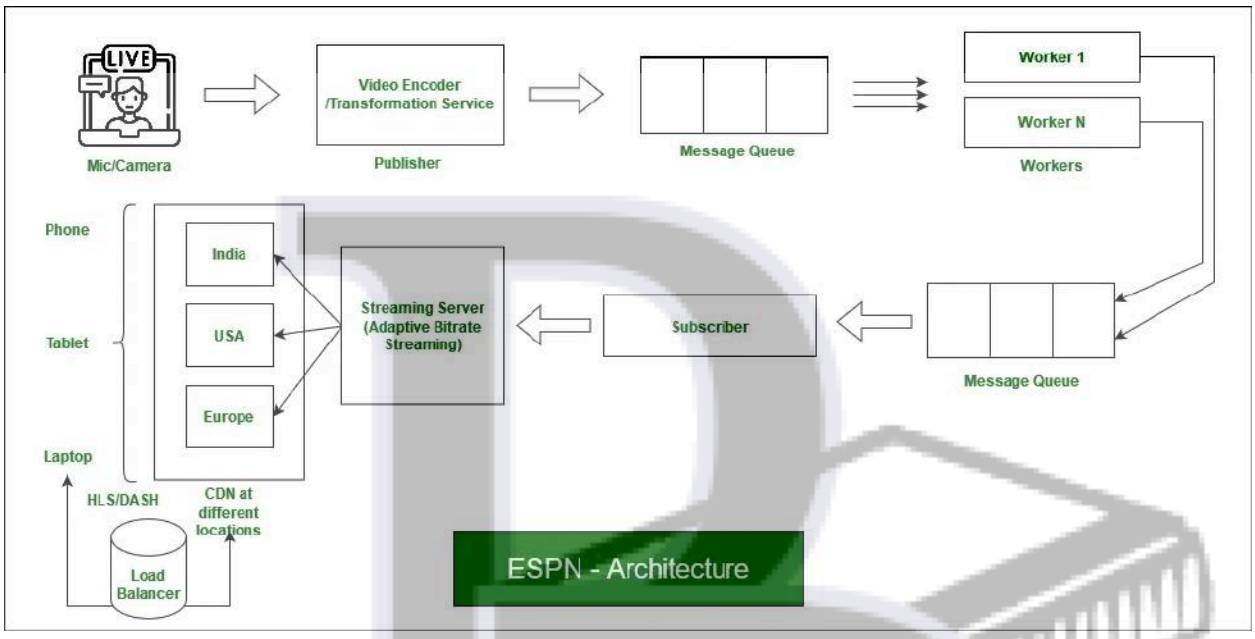


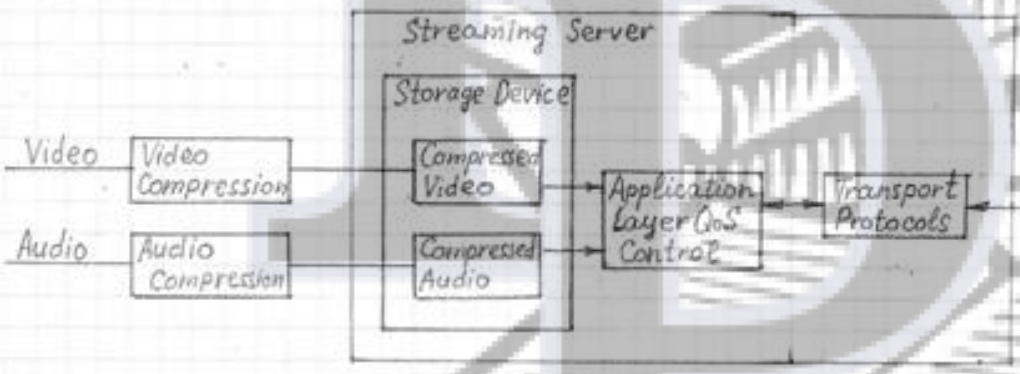
Medical Life Support System

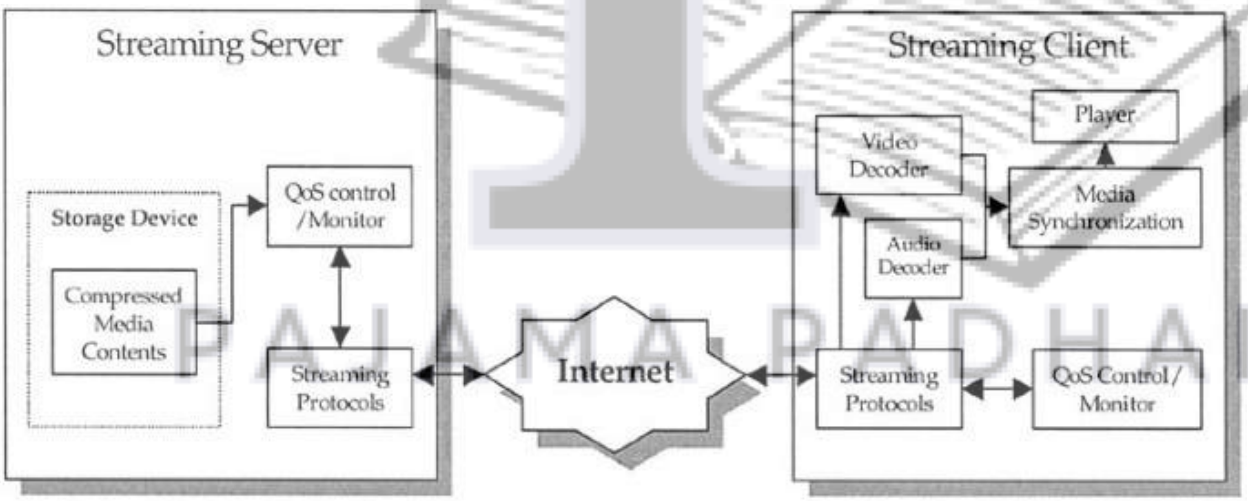


Soft Real-Time Systems:

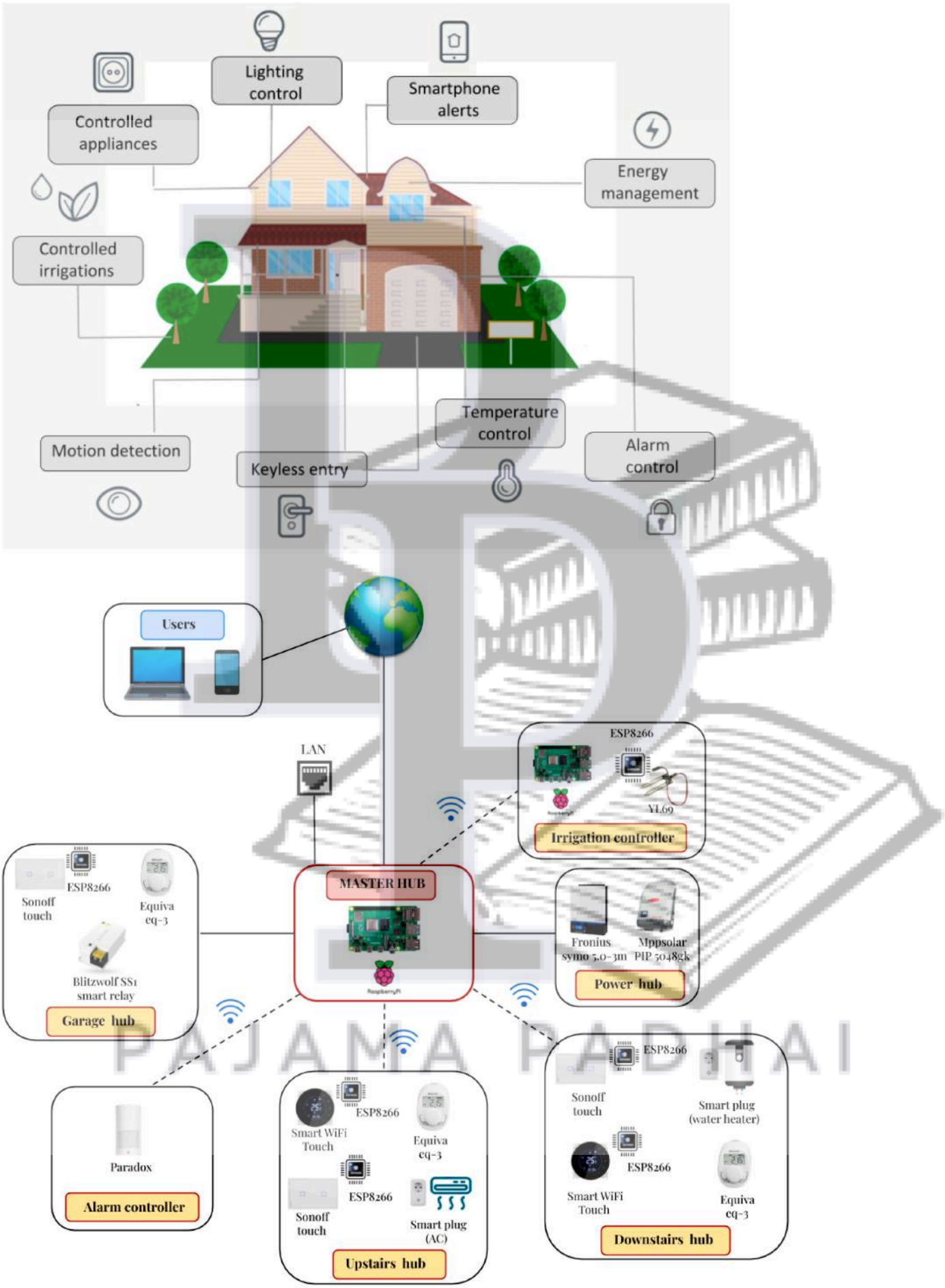
Online Video Streaming Service







Automated Home Security System



Design an embedded system that requires functionality correctness and timing correctness and write down the solution for their real time systems.

2) Design an Embedded System that requires functionality correctness and timing correctness and write down the solution for their real time systems.

Embedded System : Real-Time Traffic Light Controller

Functionality Correctness :—

→ Ensure that the traffic light controller correctly manages the flow of traffic at intersections according to predefined rules and regulations.

→ Accurately detects vehicles and pedestrians to determine the timing and sequence of traffic light changes.

→ Implements fault detection and recovery mechanisms to handle sensor failures or system malfunctions without compromising safety or functionality.

Timing Correctness :—

→ Guarantees that the timing of traffic light changes is precise and adheres to specified intervals to optimize traffic flows and minimize congestion.

→ Responds to real-time events such as vehicle detection and pedestrian crossings with minimal latency to ensure smooth and efficient traffic management.

→ Incorporates deterministic scheduling algorithms to prioritize critical tasks and maintain timing correctness even under varying load conditions.

Solution for Real-Time Systems

1. Hardware Selection

→ Choose a microcontroller or FPGA with sufficient processing power and peripheral interfaces to handle real-time sensor inputs and control signals for traffic lights.

→ Ensure that the selected hardware platform supports deterministic operation and provides mechanisms for precise timing control.

2. Real-Time Operating System (RTOS)

→ Select a lightweight RTOS with deterministic scheduling capabilities such as FreeRTOS or RTEMS.

→ Utilize features like priority-based scheduling to ensure critical tasks related to traffic light control are executed in a timely manner.

3. Sensor Integration

→ Integrate sensors such as vehicle detectors (inductive loops, infrared sensors) and pedestrian detectors (infrared sensors, pressure pads) to accurately detect traffic conditions at intersections.

→ Implement interrupt-driven handling of sensor inputs to respond promptly to changes in traffic flow.

4. Control Algorithm

→ Develop a robust control algorithm that determines the optimal timing and sequence of traffic light changes based on real-time inputs from sensors.

→ Employ a finite state machine (FSM) or similar techniques to model the behaviour of the traffic light controller and ensure correctness of operation.

5. Fault Detection and Recovery

→ Implement built-in self-test (BIST) mechanisms to periodically check the integrity of hardware components and sensor inputs.

→ Incorporate redundancy and failover mechanisms to handle sensor failures or communication errors without disrupting traffic flow.

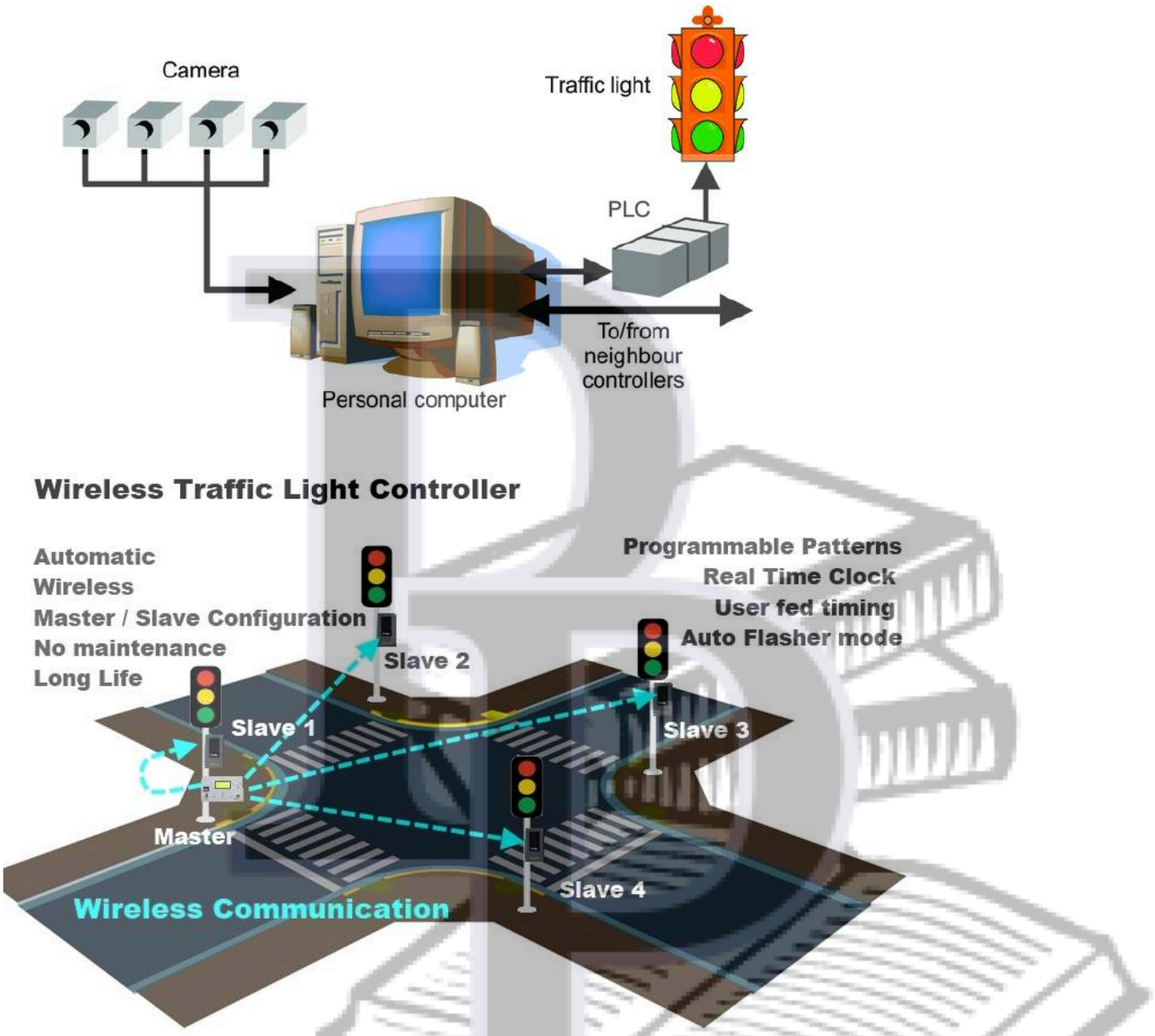
6. Validation and Testing

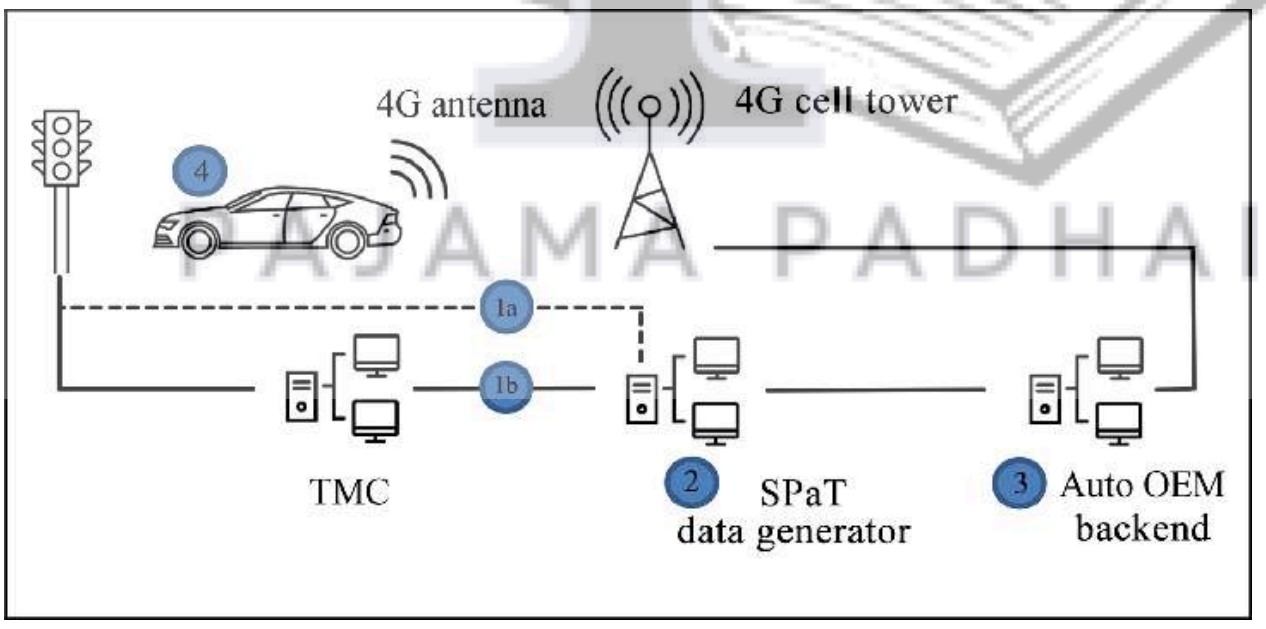
→ Conduct thorough validation and testing of the embedded system to verify functionality correctness and timing correctness under various traffic scenarios and environmental conditions.

→ Use simulation tools or hardware-in-the-loop (HIL) testing to assess the system's performance and validate real-time behaviour.

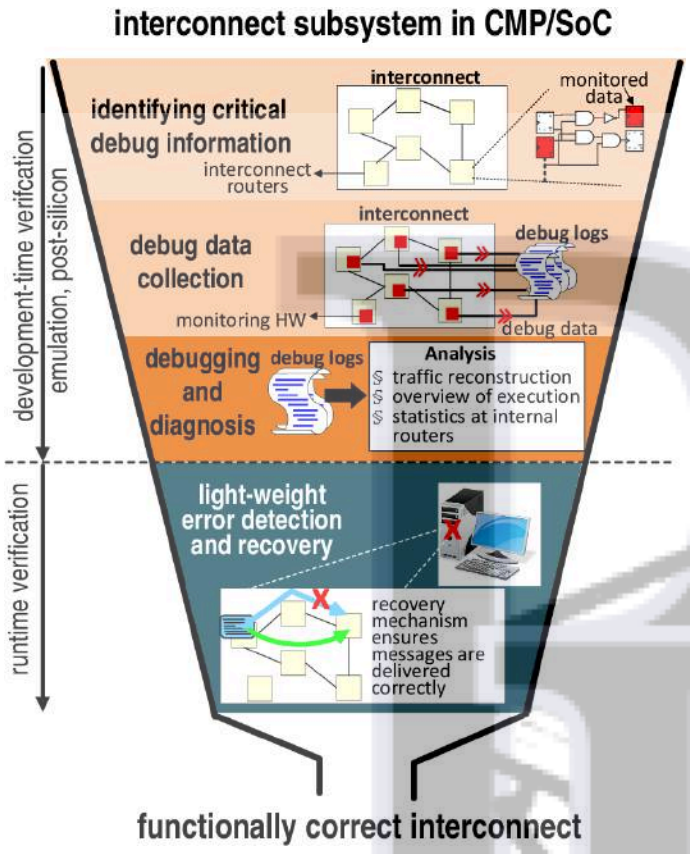
By integrating these components and ensuring functionality correctness and timing correctness, the real-time traffic light controller can effectively manage traffic flows at intersections, enhancing safety and efficiency on the road.

Real-Time Traffic Light Controller

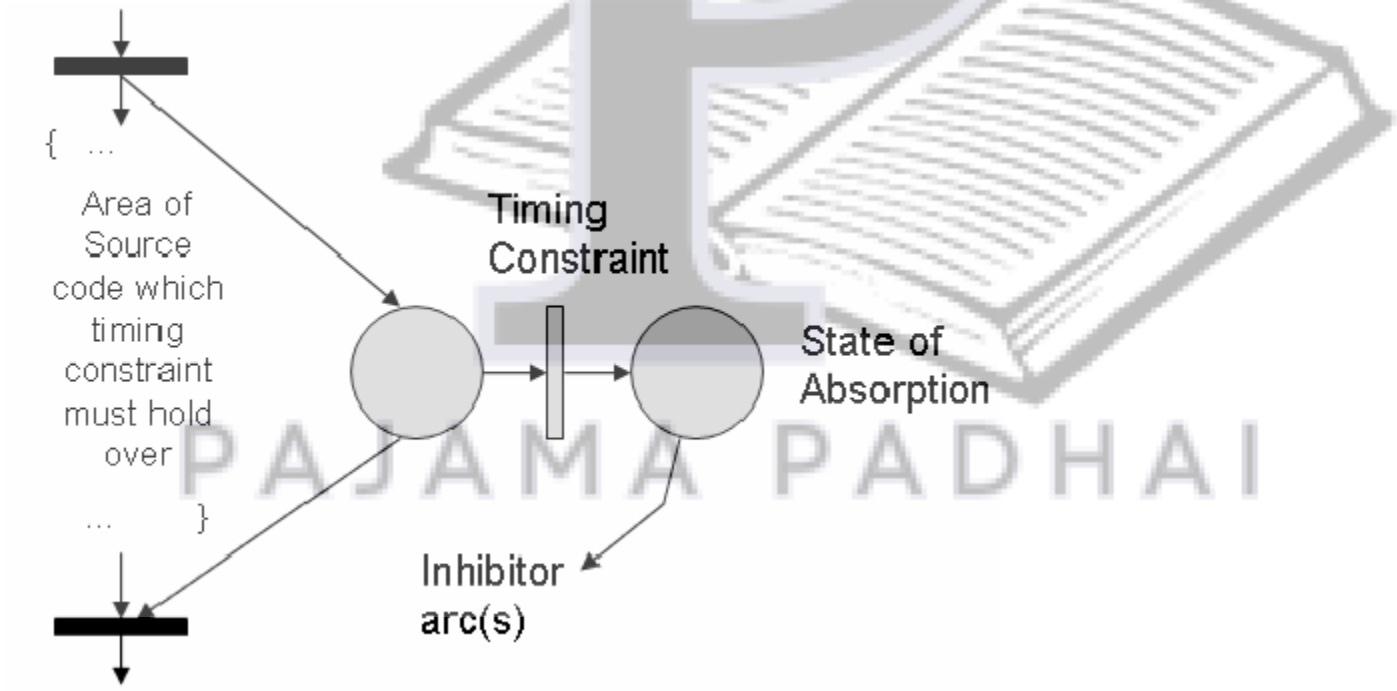




Functionality Correctness:

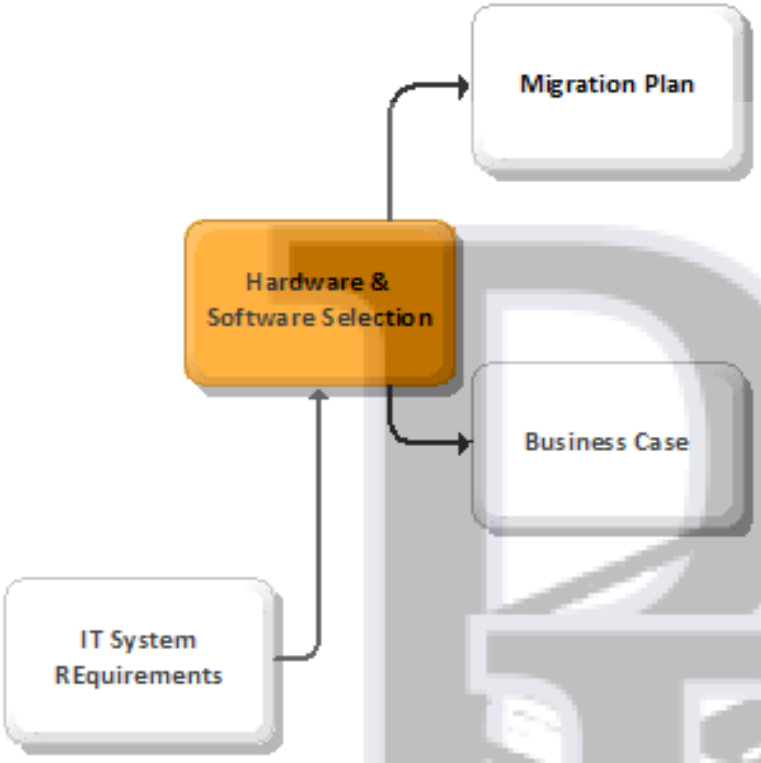


Timing Correctness:

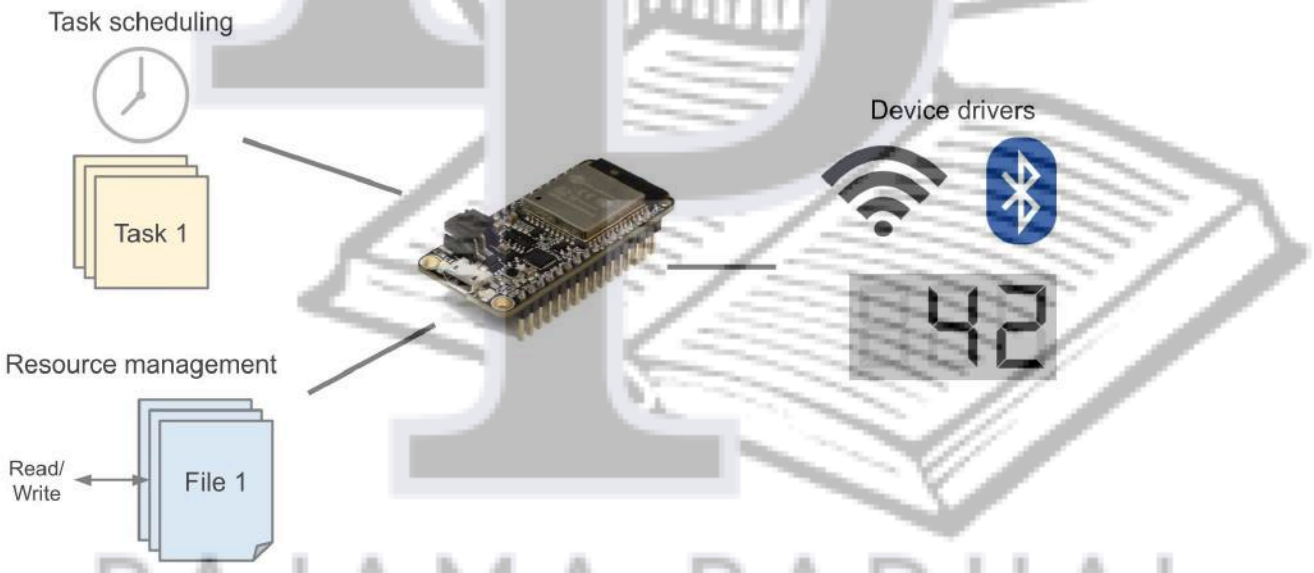


Solution for Real-Time Systems:

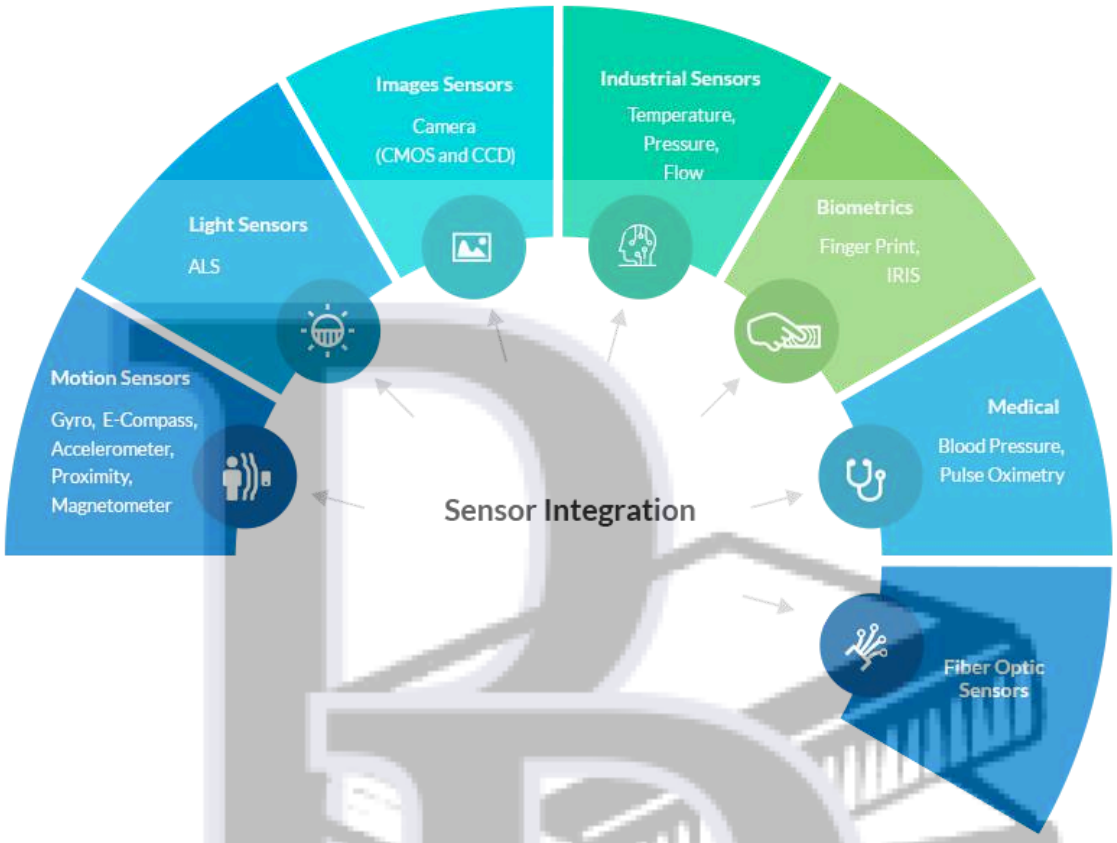
Hardware Selection:



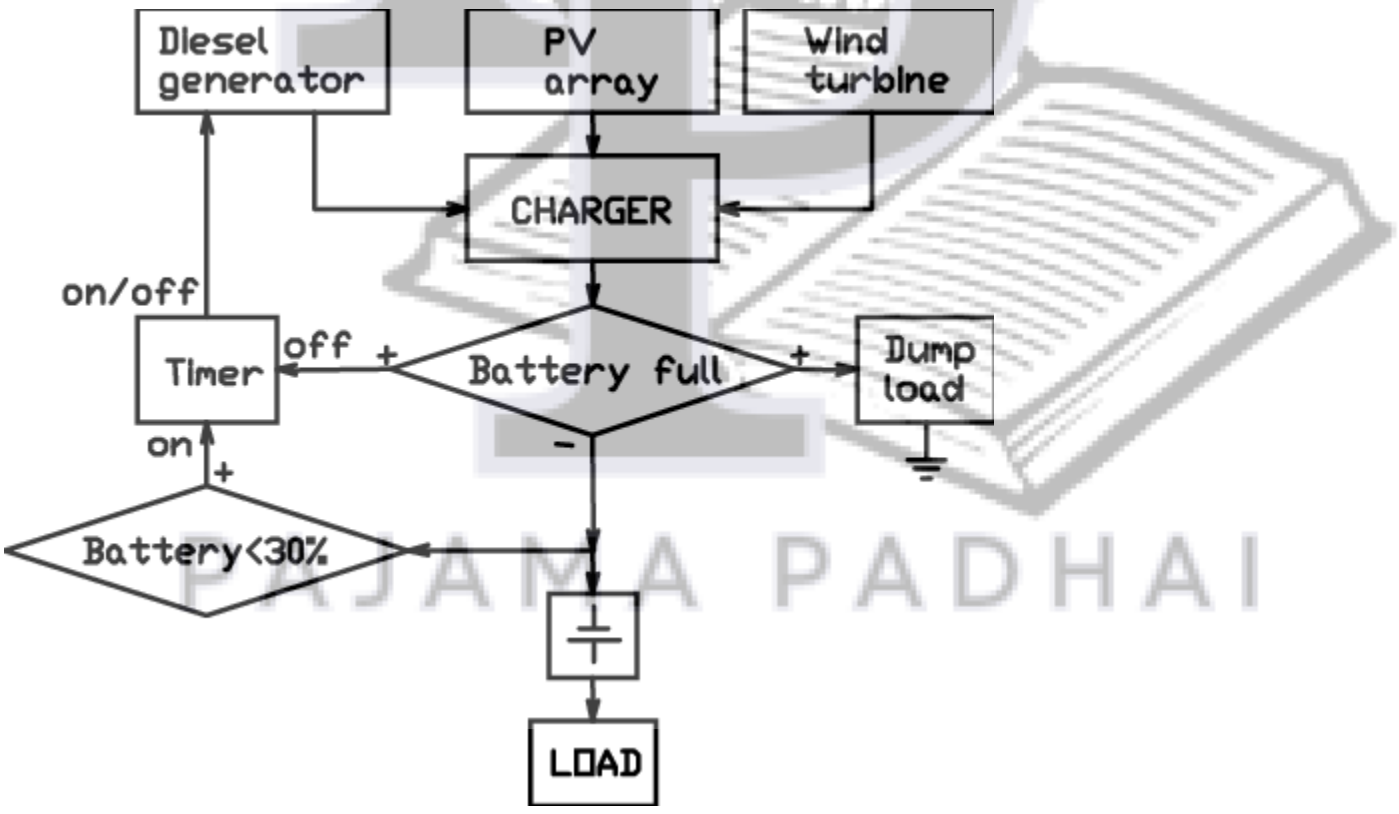
Real-Time Operating System (RTOS):

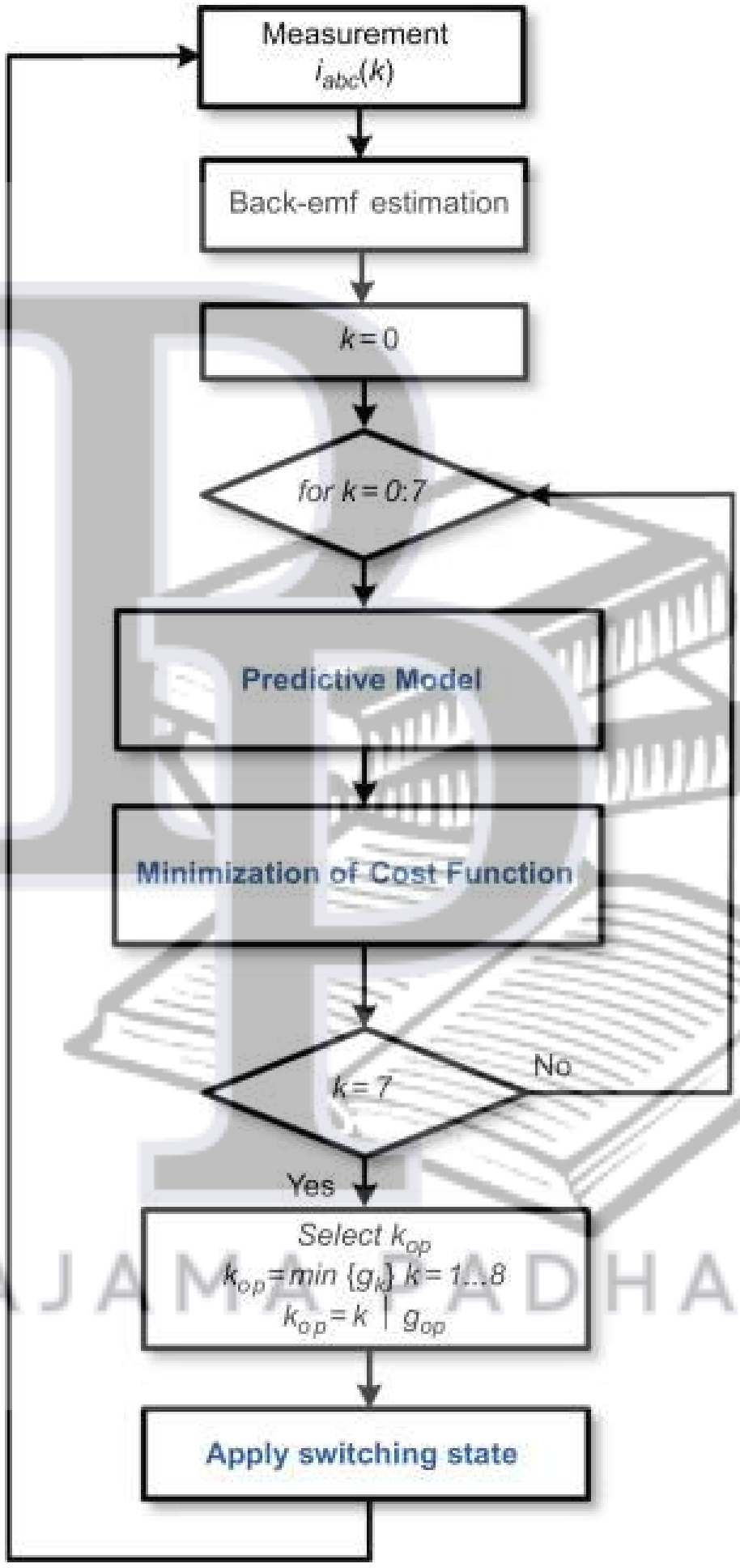


Sensor Integration:

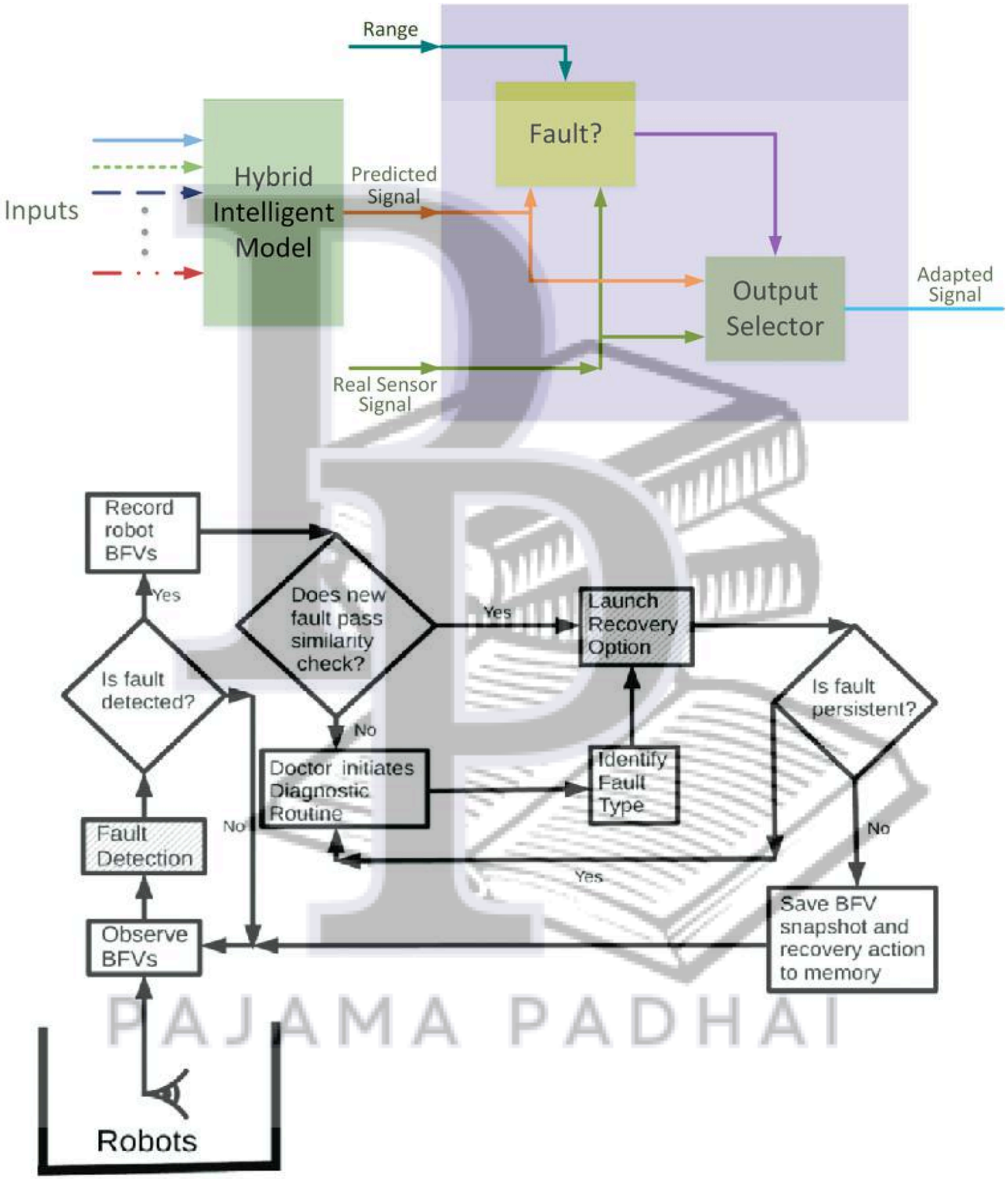


Control Algorithm:





Fault Detection and Recovery:



Validation and Testing: