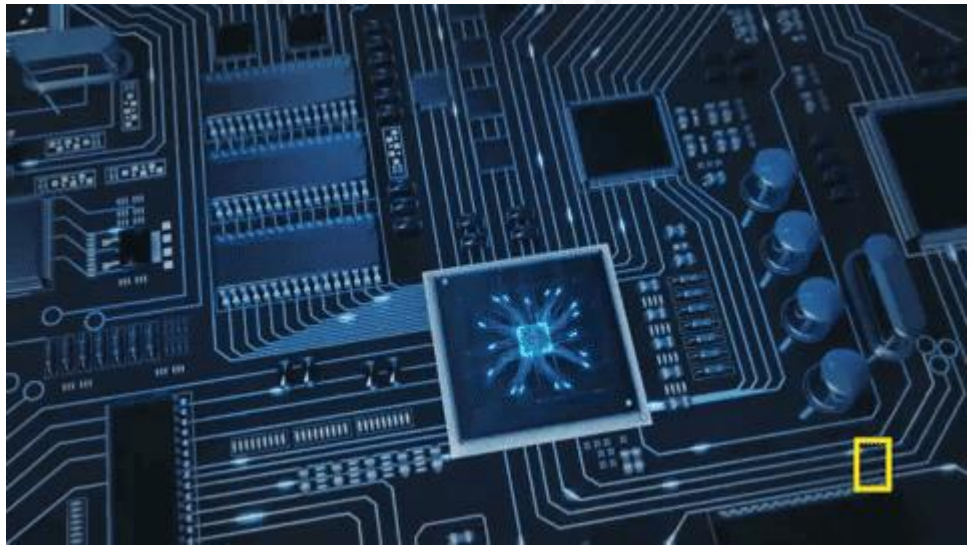
ATME College of Engineering

Department of EEE
Emitting Elite Energy

BEE613B
Embedded Systems Design
Module-2: Characteristics and Quality attributes of Embedded systems



Presented by,

Mr.Shreeshayana R

Assistant Professor

Electrical and Electronics Engineering

ATME College of Engineering, Mysuru

ATME College of Engineering

Department of EEE
Emitting Elite Energy

Contents

2.1 Characteristics and quality attributes of embedded systems: Characteristics
2.2 Operational and nonoperational quality attributes,
2.3 Application specific embedded system - washing machine,
2.4 Application specific embedded system domain specific – automotive

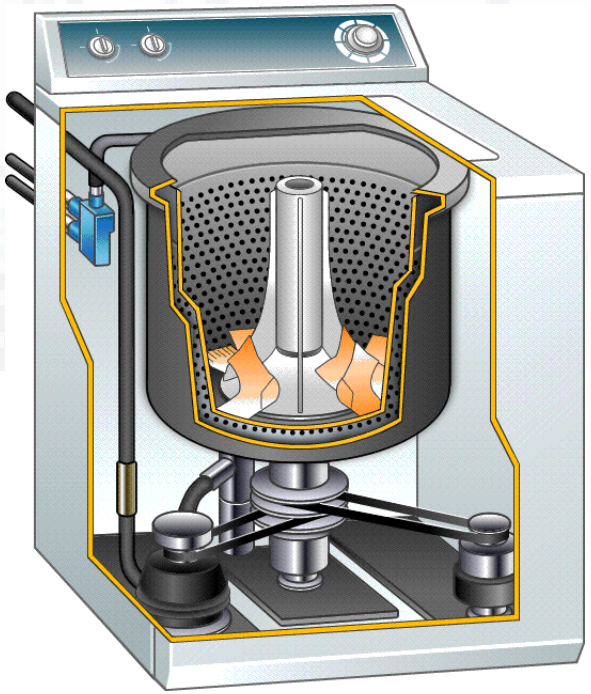
Learning Objectives

1. Understand Characteristics of Embedded Systems – Real-time, application-specific, power-efficient.
2. Analyze Quality Attributes – Differentiate operational (response, reliability) and non-operational (testability, cost).
3. Explore Application-Specific Systems – Study washing machine automation.
4. Examine Domain-Specific Systems – Learn automotive embedded systems (ECUs, ABS, infotainment).

2.1 Characteristics of Embedded Systems

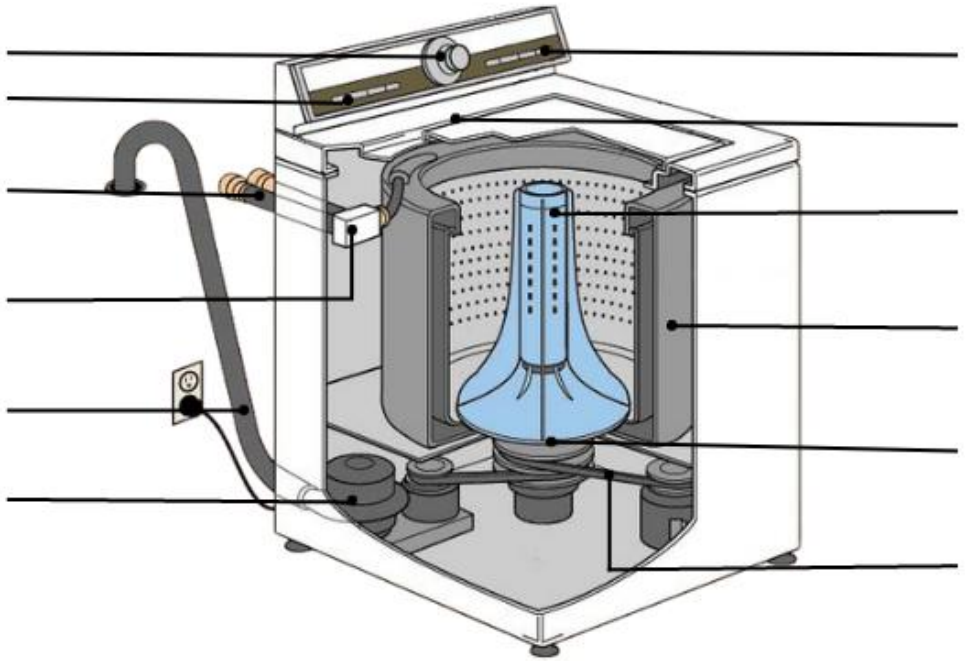
1. Application & Domain Specific

- **Embedded systems** are designed for a **particular function** or domain and cannot be repurposed.
- **Example**: A washing machine control unit cannot be used for an air conditioner.



HnK Parts
Home & Kitchen Appliance Parts

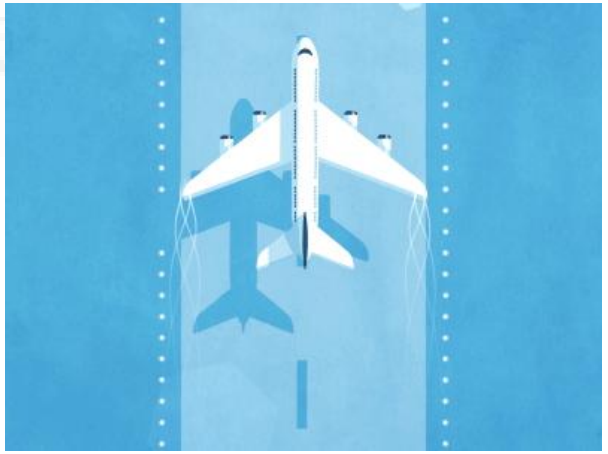
Parts of Washing Machine

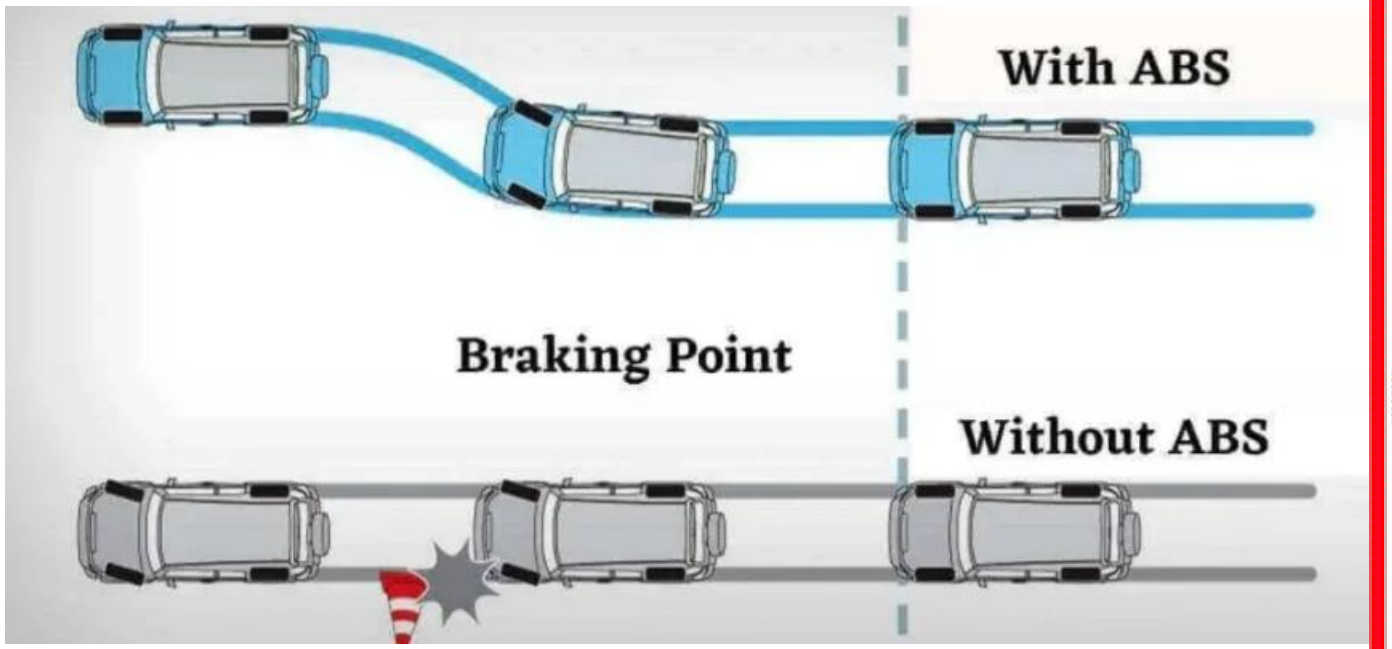


2.1 Characteristics of Embedded Systems

2. Reactive & Real-Time

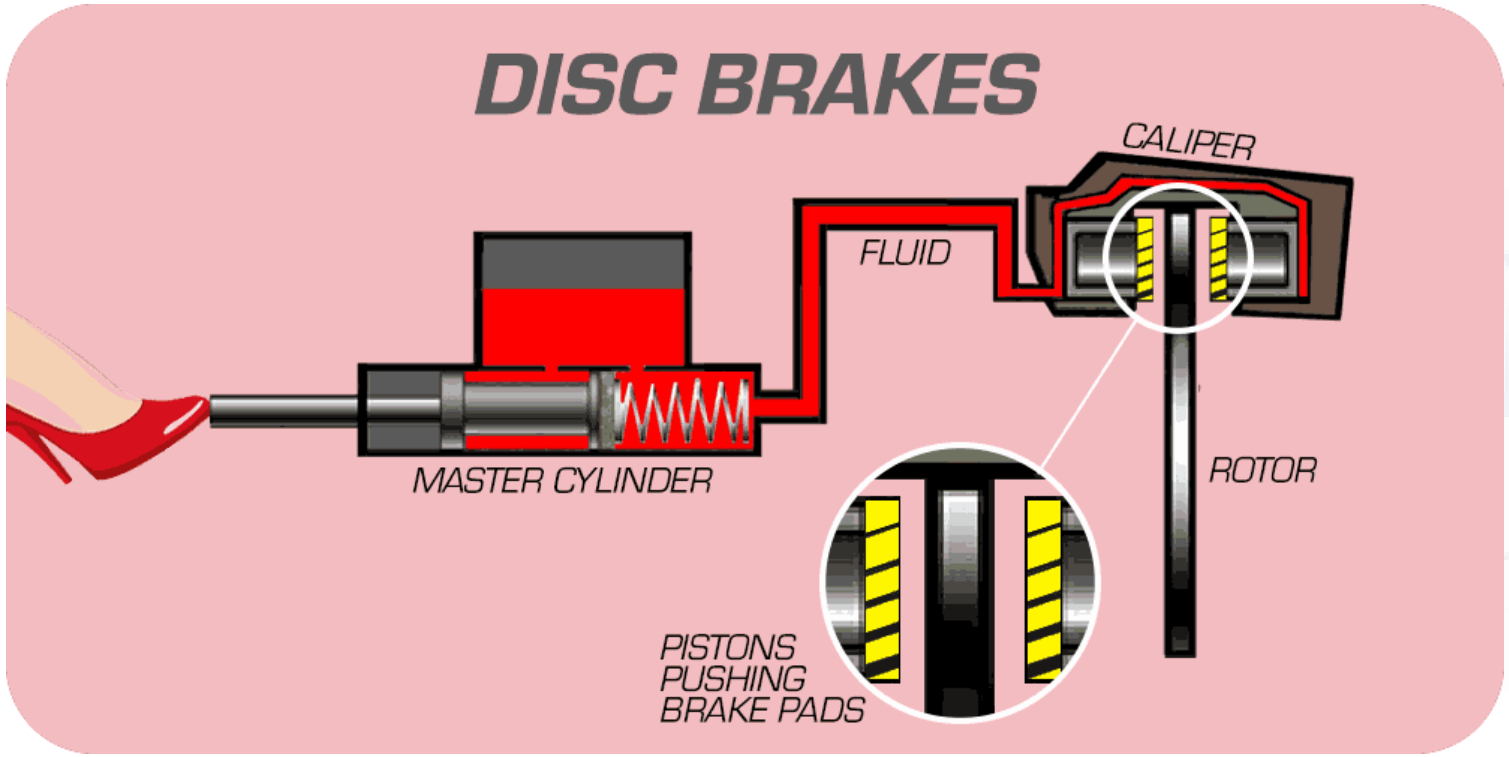
- Embedded systems interact with the real world using sensors and must react instantly to changes.
- Real-Time Systems ensure deterministic responses, critical in applications like **flight control and ABS (Anti-lock Braking Systems) in vehicles.**

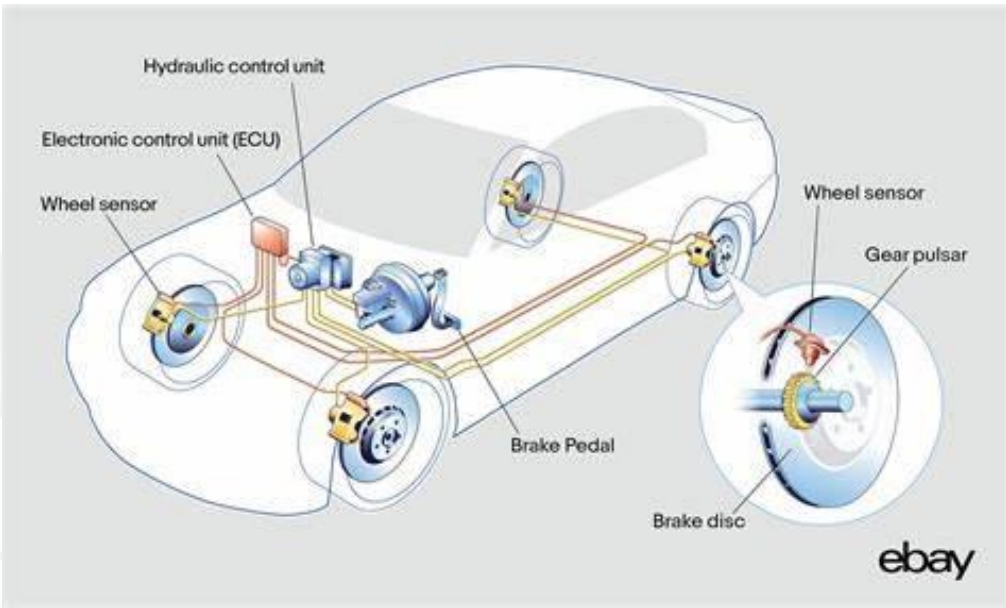




Anti-Locking Breaking System (ABS)

The **ABS** prevents the wheels from locking up and helps them maintain grip with the road below. **ABS** was first introduced as an *anti-skid system* for aircraft use in the 1950s. And in the 1970s, *Ford and Chrysler* proved that it can also be used in cars.

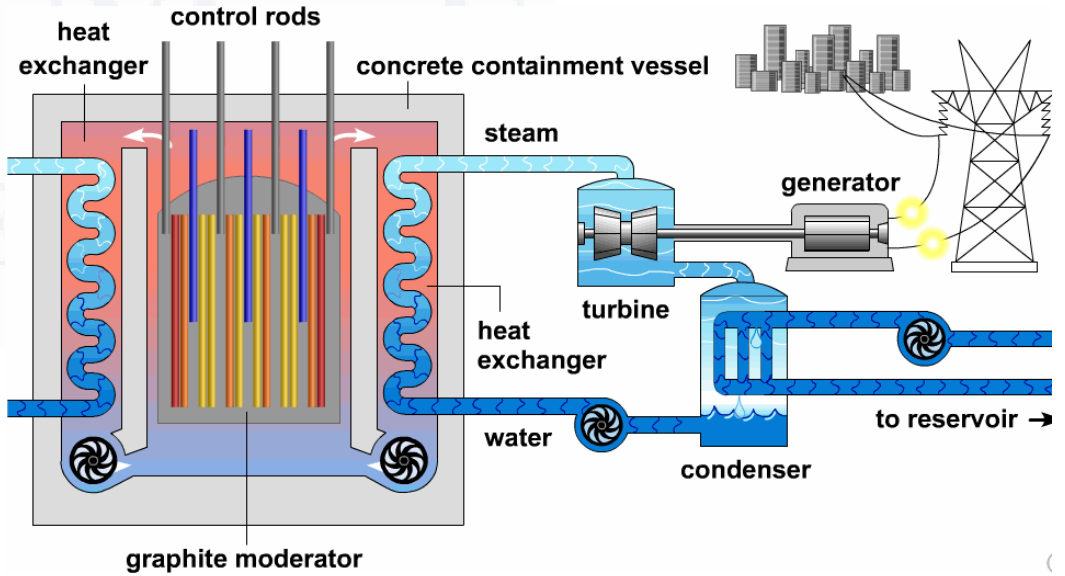

DISC BRAKES
CALIPER
FLUID
MASTER CYLINDER
ROTOR
PISTONS
PUSHING
BRAKE PADS


Hydraulic control unit
Electronic control unit (ECU)
Wheel sensor
Wheel sensor
Gear pulsar
Brake Pedal
Brake disc
ebay

2.1 Characteristics of Embedded Systems

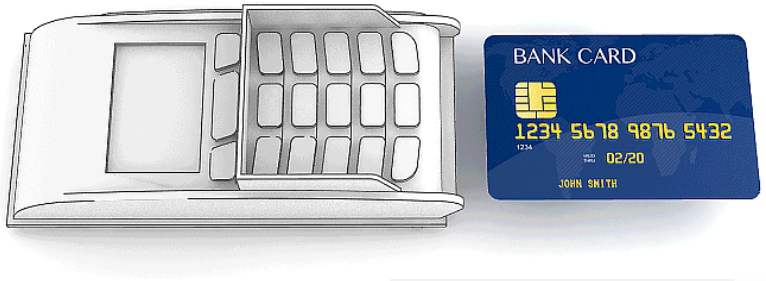
3. Operates in Harsh Environments

- Some embedded systems function in extreme conditions (e.g., high temperatures, vibrations, dust).
- **Example:** Industrial automation systems deployed in high-temperature zones.



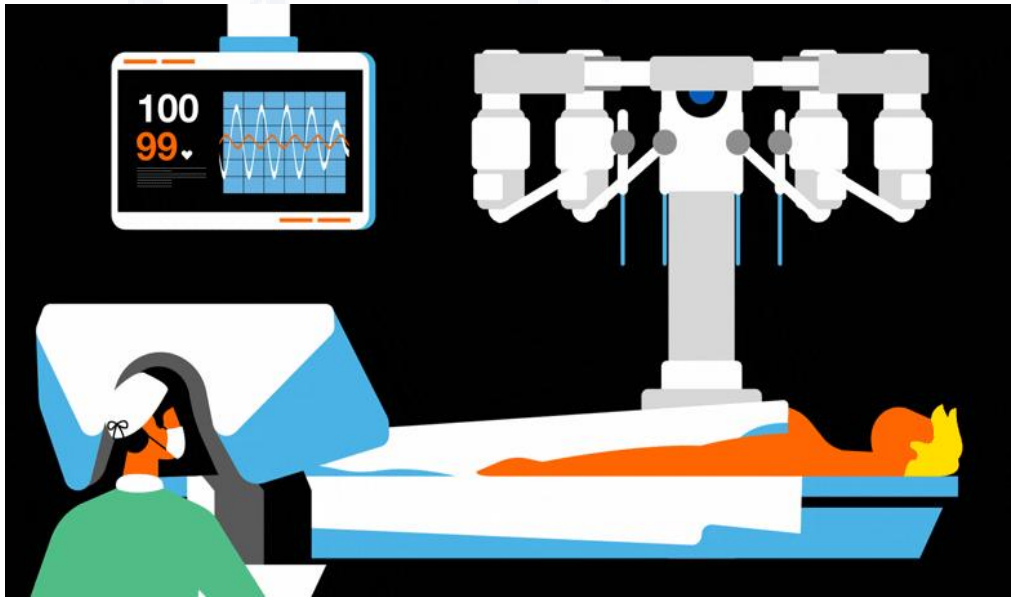
4. Distributed Systems

- Multiple embedded systems work together in a network to achieve a common function.
- **Example:** ATM machines consist of independent embedded units (card reader, cash dispenser, printer) working together.



5. Small Size & Lightweight

- Compact and aesthetically pleasing designs are important for usability and portability.
- **Example:** Mobile phones, smartwatches, and medical devices.



6. Power Efficiency

- Optimized for low power consumption to enhance battery life and reduce heat dissipation.
- **Example:** Ultra-low power controllers and energy-efficient components in IoT devices.

2.2 Quality Attributes of Embedded Systems

Operational Quality Attributes

1. Response

- Defines how quickly an embedded system reacts to input changes.
- **Example**: Flight control systems must respond instantly to pilot inputs.

Operational Quality Attributes

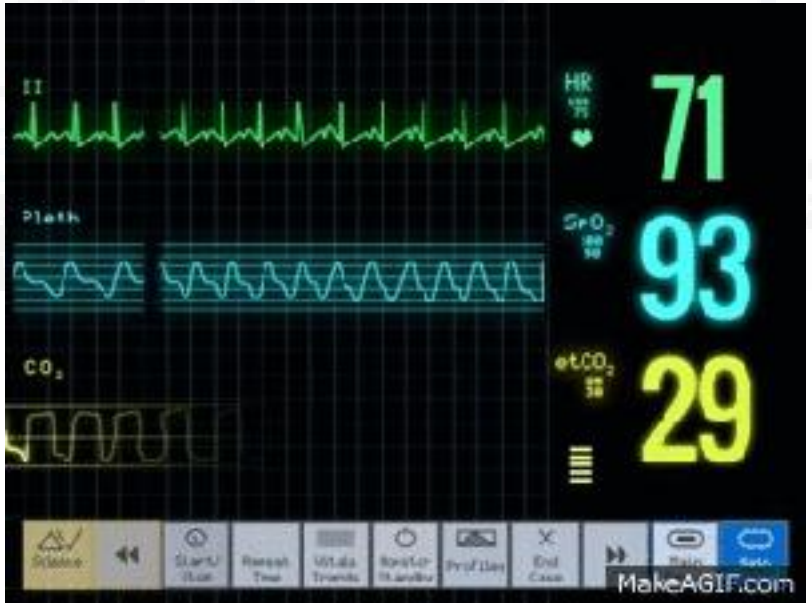
2. Throughput

- Measures system efficiency in terms of completed tasks per unit time.
- **Example:** Card readers measure throughput as transactions per second.

Operational Quality Attributes

3. Reliability

- Indicates system stability and performance over long durations.
- Measured using MTBF (Mean Time Between Failures) and MTTR (Mean Time to Repair).
- **Example:** Medical monitoring systems must function reliably 24/7.



4. Maintainability

- Deals with ease of repairing and updating the system.
- Includes scheduled maintenance (e.g., printer ink replacement) and corrective maintenance (e.g., ATM repair after failure).

5. Security

- Ensures data protection and prevents unauthorized access.
- **Example**: Password-protected PDAs, encryption in IoT devices.

.

6. Safety

- Prevents damage to users or the environment due to failures.
- **Example:** Airbag systems, industrial robots with emergency stop mechanisms.

Non-Operational Quality Attributes

1. Testability & Debug-ability

- Determines how easily the system can be tested for faults.
- Includes **hardware testing** (e.g., checking sensors) and **firmware debugging.**

How to test for hardware failures in a computer

Sensors used in Embedded Systems

1. Hardware Testing in Smart Home Automation System

Scenario 1: Temperature Sensor (DHT22) Calibration

•**Problem**: The temperature readings from the DHT22 sensor seem inconsistent and inaccurate in certain rooms of the house.

•**Testing Process:**

- **Visual Inspection**: Check for **loose connections** or damaged pins on the DHT22 sensor.
- **Signal Output Test**: Use an oscilloscope to verify the signal integrity between the sensor and microcontroller. The digital signal should have clear high and low pulses.
- **Calibration**: Place the DHT22 sensor in a known environment (e.g., a lab with a calibrated temperature sensor) to compare the readings. After calibration, the DHT22 sensor should give readings within 0.5°C of the reference sensor.
- **Environmental Testing**: Place the sensor in different areas of the house (living room, bedroom, kitchen) and monitor how it responds to temperature changes. Ensure that it doesn't give erratic readings.

•**Result**: The sensor was found to be working within acceptable tolerances after calibration. Inconsistent readings were resolved by ensuring the sensor was placed correctly (avoiding direct sunlight or airflow) and by improving signal connections.

Scenario 2: Motion Detection Sensor (PIR)

•**Problem**: The PIR sensor occasionally fails to detect motion in certain parts of the house, causing security lapses.

•**Testing Process**:

- **Visual Inspection**: Ensure the PIR sensor is securely mounted and connected. Check for obstructions in its field of view (such as furniture, curtains).
- **Signal Output Test**: Use a logic analyzer to capture the signal from the PIR sensor. The output should be a high signal when motion is detected and a low signal otherwise.
- **Environmental Testing**: Test the sensor in various lighting conditions (day/night) and temperatures to see if the sensor performance changes. PIR sensors are often sensitive to ambient light and heat sources.

•**Result**: The PIR sensor was found to be functioning well under test conditions but had issues detecting motion in areas with heavy furniture blocking its field of view. After repositioning the sensor, detection accuracy improved.

2. Firmware Debugging in Smart Home Automation System

Scenario 1: Temperature Control Algorithm (Heating and Cooling)

•**Problem**: The firmware for controlling the heating and cooling system doesn't maintain a stable room temperature. It frequently turns the heater on/off unnecessarily.

•**Debugging Process**:

- **Serial Monitoring**: Monitor the temperature readings from the DHT22 and the decision-making process in the firmware. Check for the thresholds that trigger the heater to turn on/off.
- **Edge Case Handling**: Simulate situations where the temperature reading fluctuates rapidly due to external factors (e.g., a window being opened or closed). Observe how the system responds in the serial monitor.
- **Logic Analysis**: Set breakpoints to check how the algorithm processes temperature readings. The code might be turning the heater on/off too quickly based on small fluctuations in the temperature.
- **Fix**: Adjust the logic to include a "hysteresis" effect—where the system needs a larger change in temperature before activating the heater/cooler again. This prevents constant toggling of the device.

•**Result**: After adding a hysteresis buffer (e.g., only turn the heater on when the temperature drops below 18°C and off when it exceeds 21°C), the temperature control became stable, and the heating system operated smoothly.

2. Evolvability

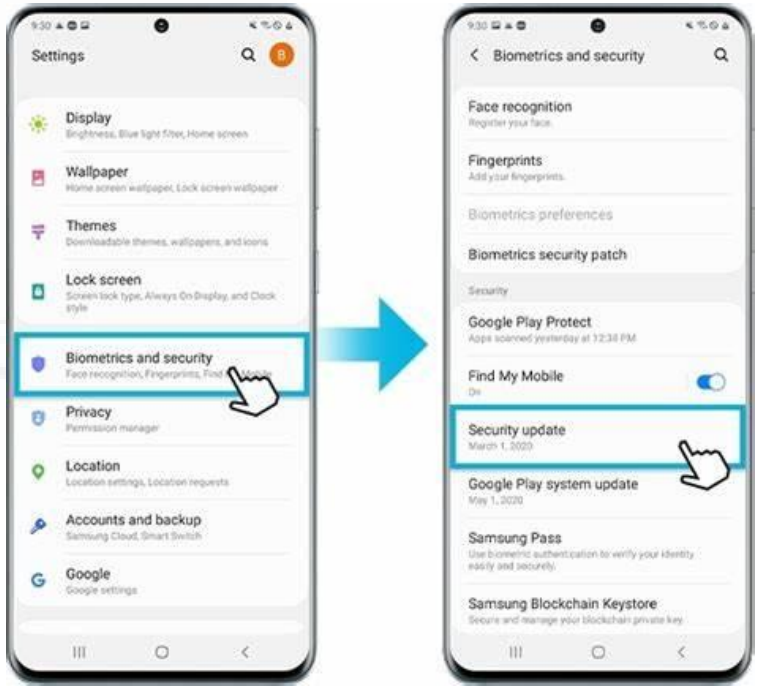
- The ability of an embedded system to integrate new features or hardware over time.
- **Example:** Firmware updates in smartphones.

How to Update HUAWEI Phone - Enrolling a Software Update

Apple Warns 1.5 Billion iPhone and Mac Users to Update Their Software Immediately Another zero-day bug could allow an attacker to take control of your device.

EXPERT OPINION BY JASON ATEN, TECH COLUMNIST @JASONATEN

AUG 19, 2022



Firmware Updates in Smartphones: A Brief Overview

Purpose: Firmware updates in smartphones improve performance, fix bugs, enhance security, and add new features. They affect the low-level software that controls hardware components like the bootloader, camera, modem, and sensors.

Types of Updates:

1. **Bootloader**: Manages the phone's startup process.
2. **Baseband/Modem**: Controls network communication (e.g., 4G/5G).
3. **System Firmware**: Updates the operating system (Android/iOS).
4. **Device-Specific**: Targets specific hardware, such as the camera or battery

3. Portability

- The capability of running on multiple platforms with minimal modifications.
- **Example:** C-based firmware can be recompiled for different processors.

Firmware is typically **low-level code that controls** the operation of hardware components, and in the case of C-based firmware, it interacts closely with the hardware to perform specific tasks like sensor control, data acquisition, and communication.

Key Difference:

•**Firmware**:
Software that is permanently programmed into a **device's hardware**, usually in non-volatile memory (e.g., ROM, Flash).
It provides **low-level control of the device's hardware**.

•**Embedded System**:
A complete system **(hardware + firmware)** designed to perform a dedicated function. The firmware is the software that operates the hardware within the embedded system.

4. Time-to-Market

- The speed from product idea to launch is crucial in competitive markets.
- **Example:** Fast prototyping using off-the-shelf components.

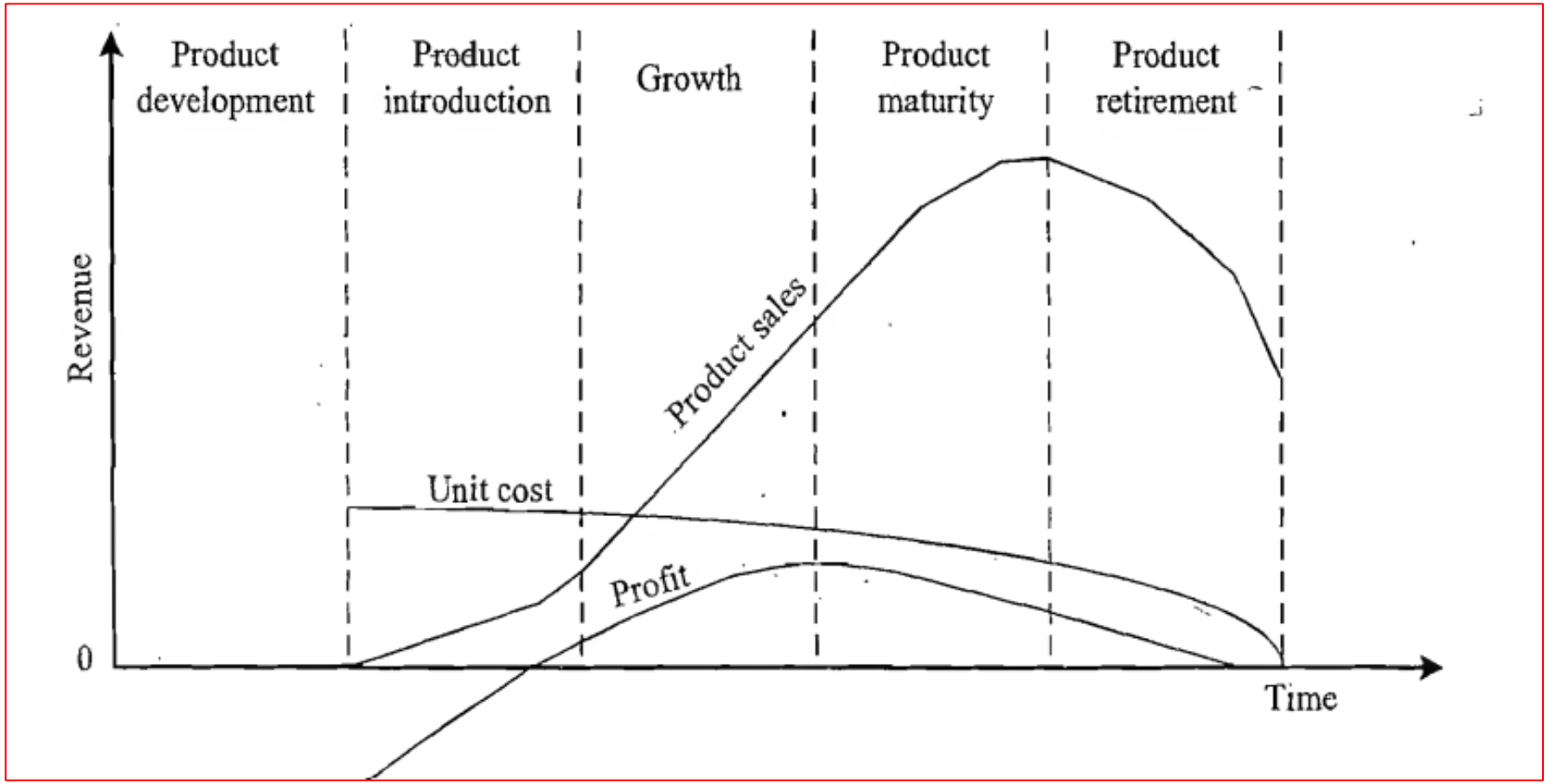
.


BEFORE
AFTER

5. Cost & Revenue

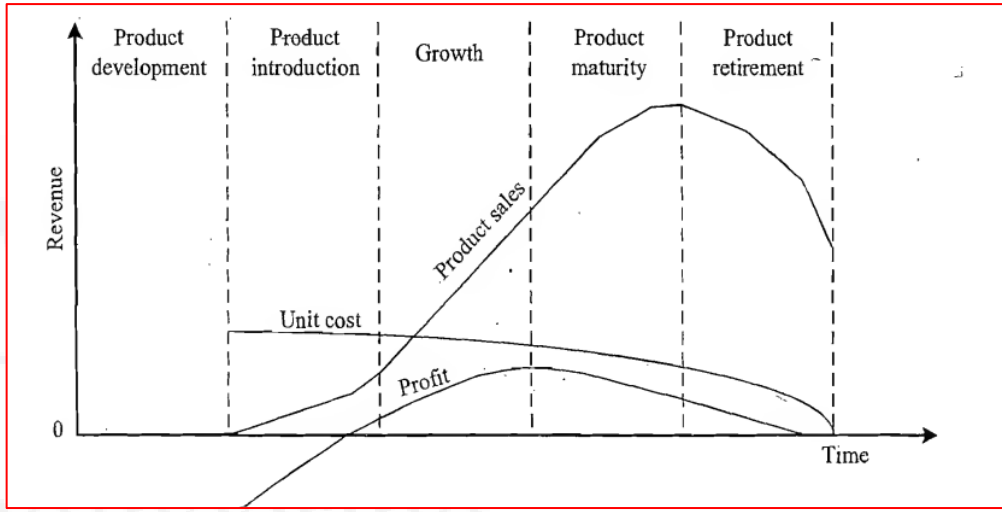
- Unit cost must be optimized to ensure profitability while maintaining quality.
- **Example:** Smartphones are expensive at launch but become cheaper over time.

1.Newer Models: Older models become cheaper as newer ones with improved features are released.
2.Economies of Scale: Increased production and manufacturing efficiency reduce the cost of components.
3.Market Competition: Competitors lower prices to remain competitive.
4.Software Support: As software updates slow down, older models lose value.
5.Retailer Discounts: Sales events and carrier promotions further reduce prices.
6.Depreciation: As demand for older models decreases, their market price drops.
In short, prices decrease as production scales, competition increases, and demand shifts toward newer models.


Product development
Product introduction
Growth
Product maturity
Product retirement
Revenue
Product sales
Unit cost
Profit
0
Time

Graph Analysis

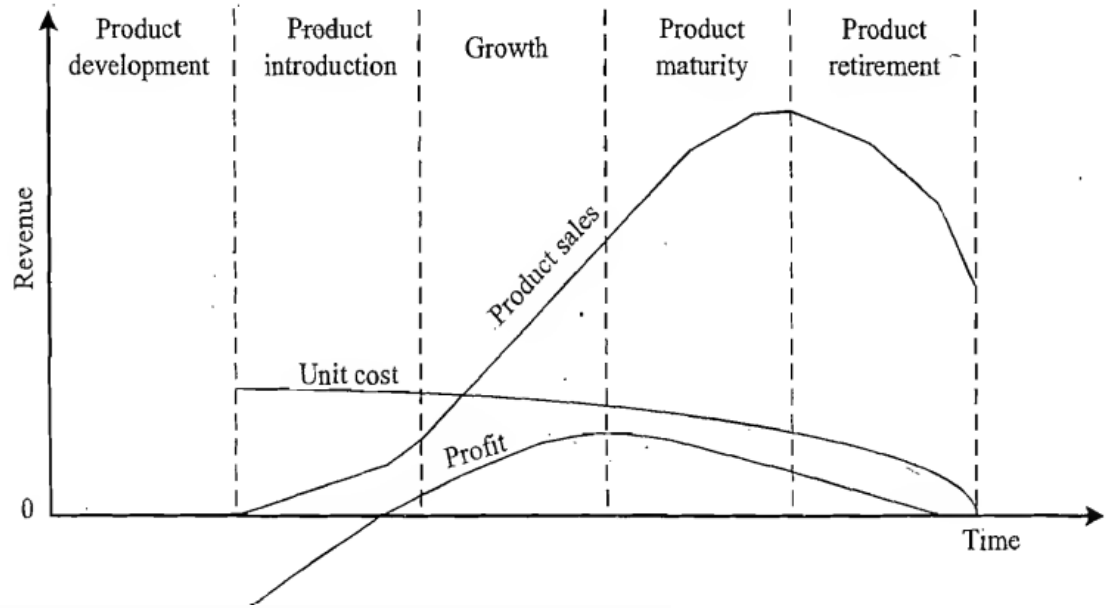
- **Product Development**: No revenue, high costs due to R&D, no profits.
- **Product Introduction**: Sales start but costs remain high, low or no profits.
- **Growth**: Sharp sales rise, increasing revenue, profits start as unit costs decrease.
- **Product Maturity**: Peak sales with slowing growth, stabilized profits, and low unit costs.
- **Product Retirement**: Sales decline due to market saturation, profits shrink, and costs may rise slightly.



Stage	Smartphone (e.g., iPhone)	DVD Player
Product Development	High development cost, no revenue	High R&D costs, no revenue
Product Introduction	Slow sales, high costs, minimal profit	Slow adoption, high costs, minimal profit
Growth	Rapid sales growth, profit increases, economies of scale	Steady sales increase, profit margins improve
Product Maturity	Sales plateau, high competition, stable profits	Sales slow, market saturation, stable or declining profits
Product Retirement	Declining sales, new models released, old models phased out	Sales decline rapidly as streaming replaces DVDs, profits fall

The product lifecycle follows a pattern where revenue grows from the introduction stage to maturity, then declines in the retirement phase.

- Unit cost is highest at launch (e.g., new smartphones are expensive initially but drop in price over time).
- Profit starts negative due to development costs, increases with sales, stabilizes, and then declines as sales drop.
- Returns exceed investment only after reaching a profitable stage.



ATME College of Engineering

Department of EEE
Emitting Elite Energy

Application specific embedded system - Washing Machine

PREPARED BY SHREESHAYANA R,ASSISTANT PROF,DEPT OF EEE,ATMECE,MYSURU

A **washing machine** is an electrical appliance used for washing clothes by automatically **agitating**, **rinsing**, and **spinning** them to remove dirt and water.

It typically consists of a **drum, motor**, **control panel**, **water inlet/outlet**, and **various washing modes**.

Washing Machine in Embedded Systems

In the context of **embedded systems**, a washing machine is an **embedded-controlled appliance** that uses a **microcontroller or microprocessor** to automate its functions.

The embedded system in a washing machine controls operations such as:

1.Water Level Control – Sensors detect the amount of water required for different loads.

2.Motor Control – The motor is controlled for different washing speeds and directions.

3.User Interface – LCD/LED displays, buttons, and touch panels allow users to select wash cycles.

4.Temperature Regulation – Sensors monitor and regulate water temperature.

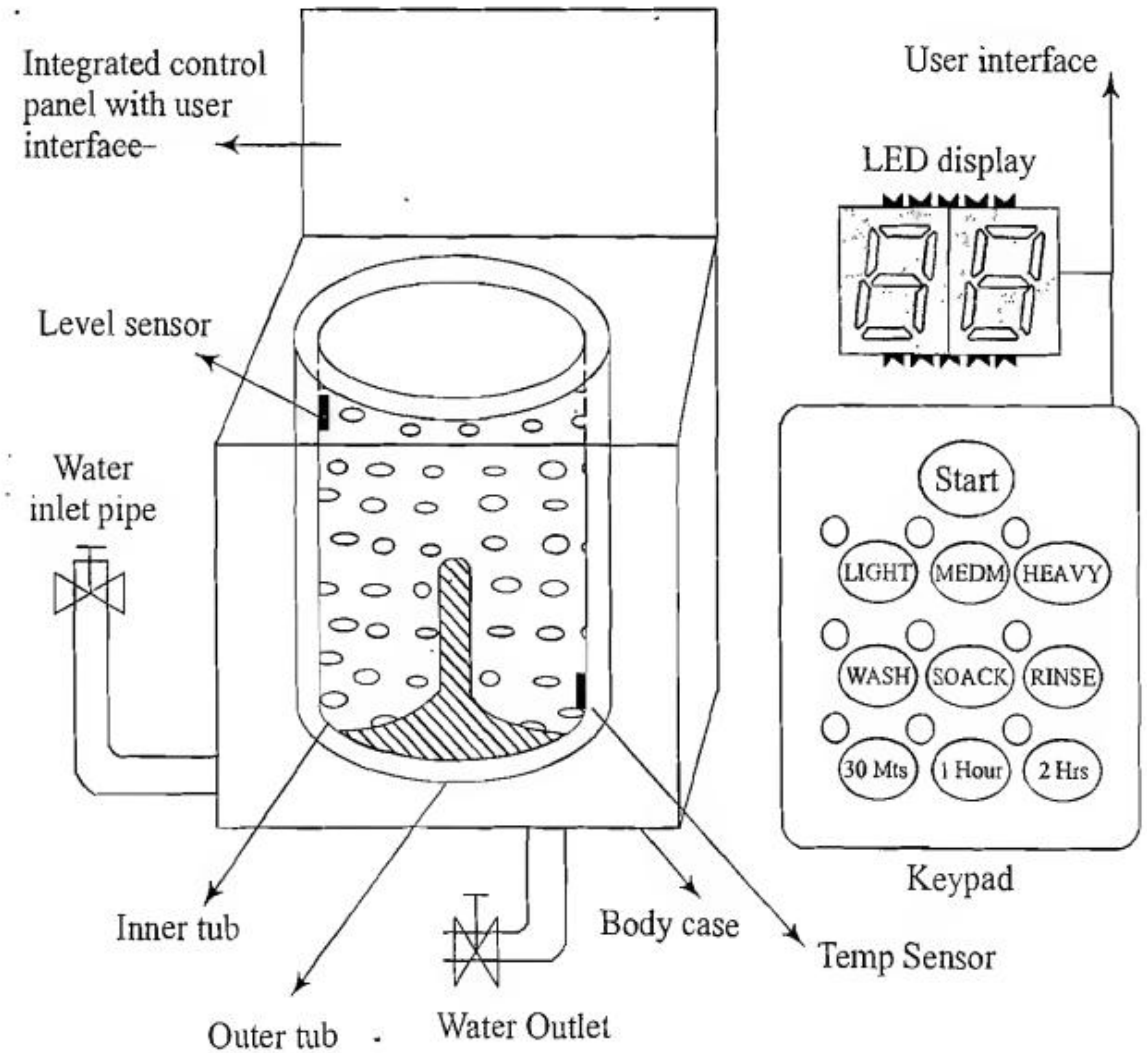
5.Spin & Drain Control – Timers and sensors control spinning and draining cycles.

6.Error Detection & Safety – Embedded systems detect unbalanced loads, overheating, and water leakage.

Water Level Control – Sensor

Motor


Cotton
Cotton Eco
Sports Wear
Mix
Delicate
Easy Care
Wool
Duvet
Intensive 60
Skin Care
Quick 30
Baby Care
Rinse+Spin
★ Press & Hold 3sec. for Tub Clean Function
12:35
Detecting
Wash
Rinse
Spin
Time Delay
Medic Rinse
Pre Wash
1400
800
400
No Spin
Rinse Hold
95°C
60
40
30
Cold
Intensive
Crease Care
Spin
Temp.
SMART DIAGNOSIS™

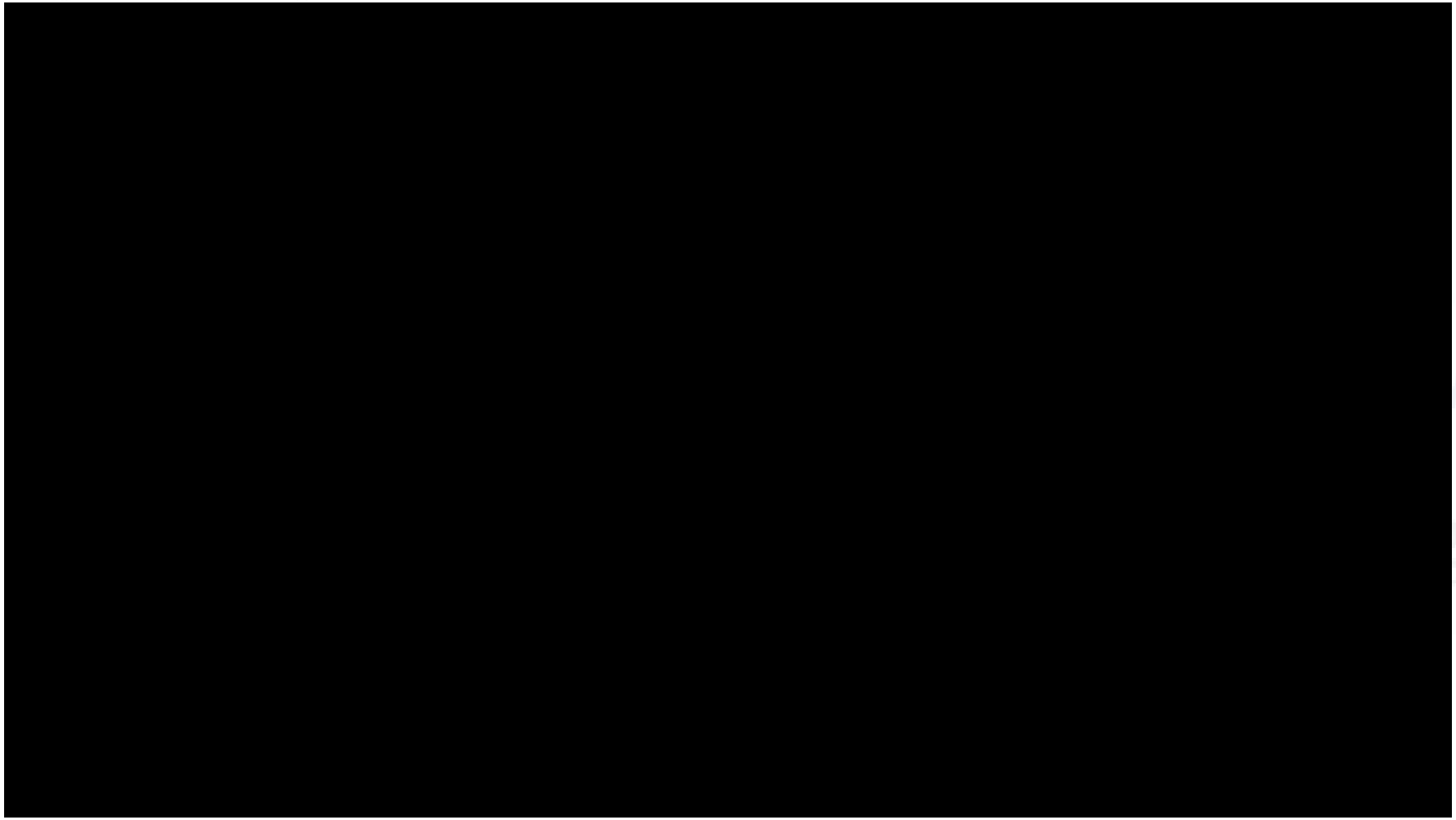


Picture not to scale

Feature	**Top Load**	**Front Load**
Drum Orientation	Vertically aligned	Horizontally aligned
Water Usage	40-60 liters per cycle	25-40 liters per cycle
Energy Consumption	Higher due to more water heating	Lower due to efficient heating and spinning
Wash Mechanism	Uses an impeller or agitator	Tumbles clothes using gravity
Spin Speed	700-900 RPM	1000-1400 RPM
Drying Time	Longer (lower water extraction)	Faster (higher water extraction)
Detergent Type	Uses more detergent (high-sudsing)	Requires HE (low-sudsing) detergent
Wash Cycle Time	Shorter (30-45 mins)	Longer (60-90 mins)
Noise & Vibration	More vibration and noise	Quieter operation
Durability	Less complex, longer lifespan	More parts, requires careful maintenance

Which One to Choose?

- **Top Load:** Best for affordability, ease of use, and quick washing.
- **Front Load:** Best for efficiency, deep cleaning, and long-term cost savings.



Washing Machine – Application-Specific Embedded System

1. Embedded System Role:

- Uses sensors, actuators, control units, and user interfaces to automate washing.
- Enhances home automation with intelligent processing.

2. Main Components:

- **Actuators**: Motorized agitator, tumble tub, water pump, inlet valve.
- **Sensors**: Water temperature sensor, water level sensor.
- **Control Unit**: Microcontroller processes sensor data and manages actuators.

3. Types of Washing Machines:

- **Top Loading**: Agitator moves clothes up and down.
- **Front Loading**: Clothes tumble repeatedly in water.

4. Washing Phases:

- **Wash**: Agitator/tub moves clothes through water and detergent.
- **Spin:** High-speed tub rotation removes excess water using centrifugal force.
- **Rinse**: Removes detergent by adding fresh water and spinning.

5. User Interface & Controls:

- **Inputs:** Wash type selector (Wash, Spin, Rinse), cloth type selector (Light, Medium, Heavy Duty), timer settings.
- **Outputs**: LED/LCD display, status indicators.

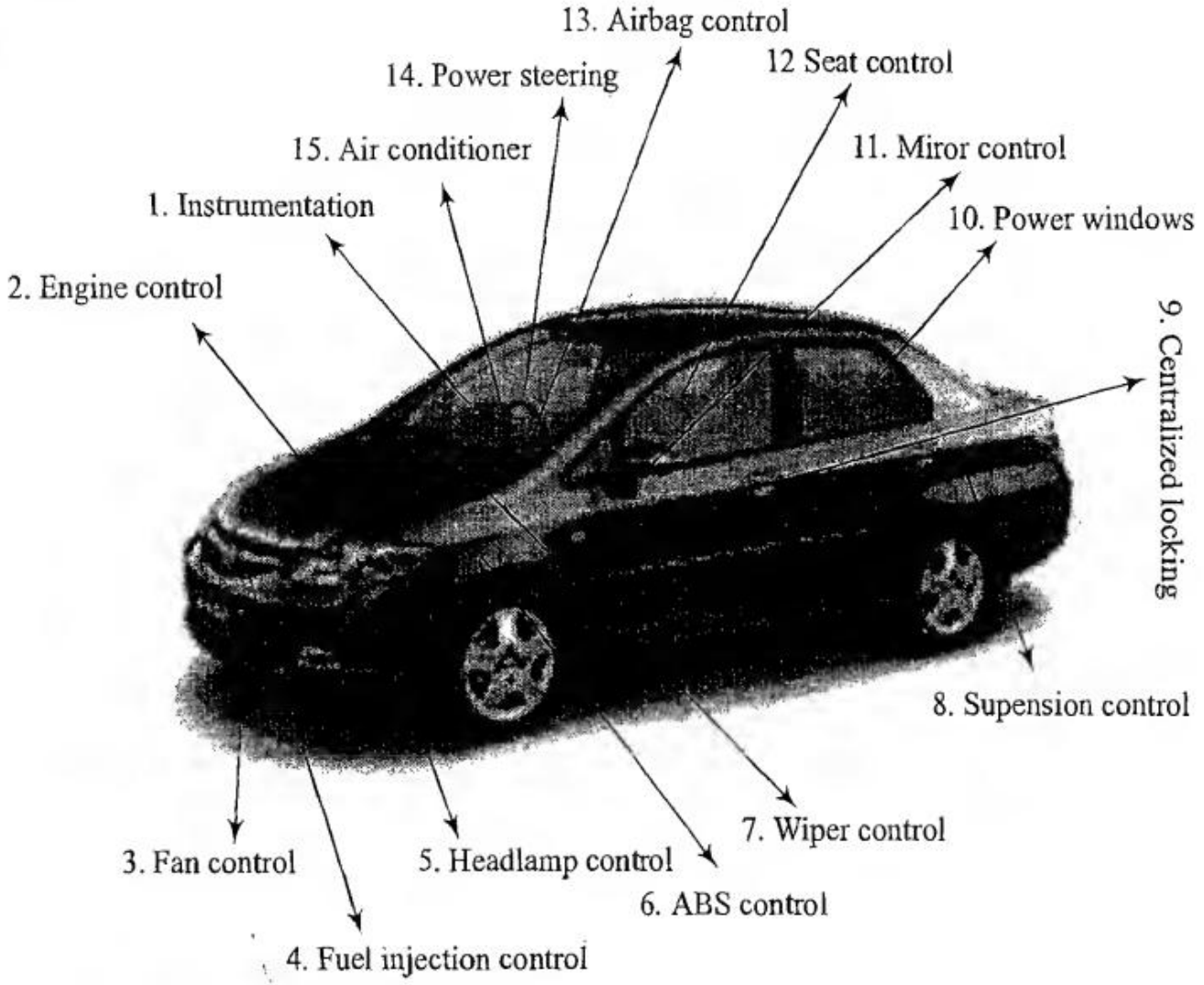
6. Working Principle:

- Sensors detect water levels & temperature → Microcontroller processes data → Actuators perform washing functions.

7. Manufacturer Variations:

- Basic principles remain the same, but interface & control panel designs vary.

Application specific embedded system domain specific – automotive


13. Airbag control
14. Power steering
12 Seat control
15. Air conditioner
11. Miror control
1. Instrumentation
10. Power windows
2. Engine control
9. Centralized locking
8. Supension control
7. Wiper control
3. Fan control
5. Headlamp control
6. ABS control
4. Fuel injection control

Embedded Systems in Automotive refer to specialized computing systems integrated into vehicles to perform dedicated functions such as **engine control, safety monitoring, infotainment, and driver assistance.**

These systems are designed for real-time operation, reliability, and efficiency, enhancing vehicle performance, safety, and user experience.

1. **Instrumentation** – Displays vehicle speed, fuel level, engine temperature, and other critical information.
2. **Engine Control** – Manages engine performance, fuel efficiency, and emissions.
3. **Fan Control** – Regulates the cooling fan to maintain optimal engine temperature.
4. **Fuel Injection Control** – Controls the precise amount of fuel injected into the engine for combustion.
5. **Headlamp Control** – Automatically adjusts headlight brightness and switching based on conditions.
6. **ABS Control** – Prevents wheel lock-up during sudden braking for improved stability.
7. **Wiper Control** – Adjusts wiper speed based on rain intensity and user settings.
8. **Suspension Control** – Adjusts suspension settings for a smooth and stable ride.

9. Centralized Locking – Controls the locking/unlocking of all doors from a single point.

10. Power Windows – Enables electronic control of window operation for convenience.

11. Mirror Control – Allows electronic adjustment of side mirrors for better visibility.

12. Seat Control – Adjusts seat position electronically for comfort and ergonomics.

13. Airbag Control – Detects collisions and deploys airbags for occupant safety.

14. Power Steering – Provides assisted steering for easy maneuverability.

15. Air Conditioner – Regulates cabin temperature for passenger comfort.


0 km
行驶里程
0:03 h
行驶时间
16:22
TrA 70%


RCC 361

Automotive Embedded Systems – Key Points

1. Overview

- Embedded systems control mechanical functions in vehicles.
- Found in simple (wipers, mirrors) to complex (ABS, airbags) systems.
- Built on microcontrollers, DSPs, or hybrid ECUs (Electronic Control Units).
- 20-40 ECUs in standard cars, 75-100 in luxury vehicles.

2. Types of ECUs (Electronic Control Units)

- **High-Speed ECUs (HECUs):** Require fast response for engine control, ABS, fuel injection, electronic throttle, transmission control.
- **Low-Speed ECUs (LECUs):** Used for non-critical functions like audio controls, door locks, mirrors, wipers, seat adjustments.

3. Automotive Communication Buses

- **CAN (Controller Area Network)**: High-speed (1 Mbps), used for safety & powertrain systems (ABS, airbags, engine control).
- **LIN (Local Interconnect Network):** Low-speed (20 Kbps), used for sensor-actuator interfacing (mirrors, fans, seats, windows).
- **MOST (Media-Oriented System Transport):** Fiber-optic bus for automotive multimedia systems (audio, video, infotainment).

4. Automotive Embedded Market Key Players

(a) Silicon Providers

- **Analog Devices**: Signal processing chips for GPS, driver assistance systems.
- **Xilinx:** FPGAs & CPLDs for collision avoidance, voice recognition.
- **Atmel:** Flash microcontrollers for car infotainment, safety, and CAN/LIN networking.
- **Maxim/Dallas, NXP, Renesas, Texas Instruments, Infineon, NEC** – Leading chip providers for automotive control systems.

(b) Tools & Platform Providers

- **Enea OSE**: Real-time OS for multi-core, fault-tolerant systems.
- **MathWorks MATLAB/SIMULINK**: Used for simulation & modeling.
- **Keil Microvision, Lauterbach, ARTiSAN, Microsoft Windows CE** – Provide debugging & development tools for automotive applications.

(c) Solution Providers (OEMs)

- Bosch Automotive: Full automotive solutions (engine control, safety, infotainment).
- DENSO: Provides engine management, hybrid vehicle solutions.
- Delphi, Infosys: Supply hardware & software automotive solutions.

Thank You