masters on the same bus. The following bus interface diagram shown in Fig. 2.26 illustrates the connection of master and slave devices on the I2C bus.

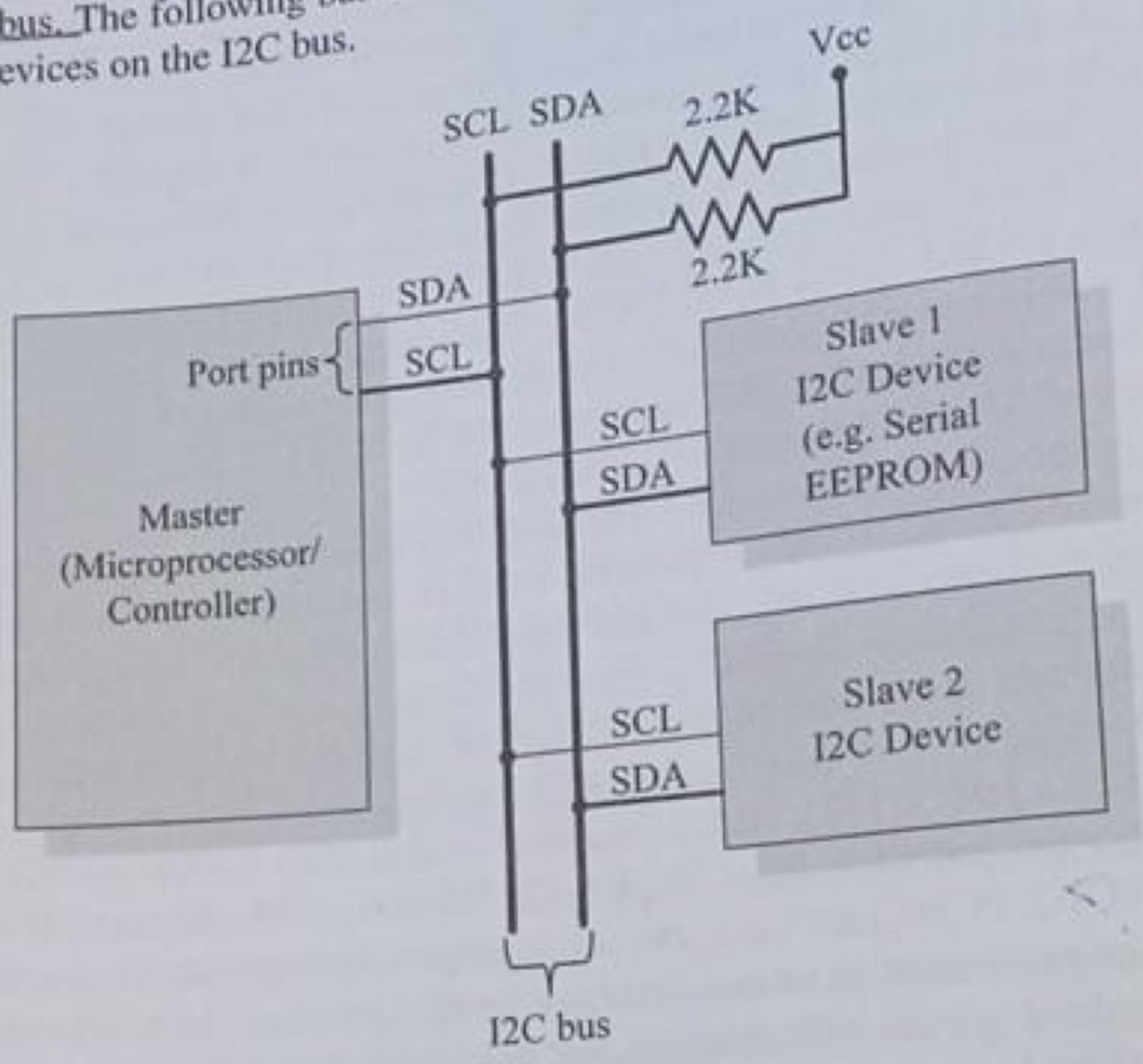

Fig. 2.26 I2C Bus Interfacing

The I2C bus interface is built around an input buffer and an open drain or collector transistor. When the bus is in the idle state, the open drain/collector transistor will be in the floating state and the output lines (SDA and SCL) switch to the 'High Impedance' state. For proper operation of the bus, both the bus lines should be pulled to the supply voltage (+5V for TTL family and +3.3V for CMOS family devices) using pull-up resistors. The typical value of resistors used in pull-up is 2.2K. With pull-up resistors, the output lines of the bus in the idle state will be 'HIGH'.

The address of a I2C device is assigned by hardwiring the address lines of the device to the desired logic level. The address to various I2C devices in an embedded device is assigned and hardwired at the time of designing the embedded hardware. The sequence of operations for communicating with an I2C slave device is listed below:

1. The master device pulls the clock line (SCL) of the bus to 'HIGH'
2. The master device pulls the data line (SDA) 'LOW', when the SCL line is at logic 'HIGH' (This is the 'Start' condition for data transfer)
3. The master device sends the address (7 bit or 10 bit wide) of the 'slave' device to which it wants to communicate, over the SDA line. Clock pulses are generated at the SCL line for synchronising the bit reception by the slave device. The MSB of the data is always transmitted first. The data in the bus is valid during the 'HIGH' period of the clock signal
4. The master device sends the Read or Write bit (Bit value = 1 Read operation; Bit value = 0 Write operation) according to the requirement
5. The master device waits for the acknowledgement bit from the slave device whose address is sent on the bus along with the Read/Write operation command. Slave devices connected to the bus compares the address received with the address assigned to them

masters on the same bus. The following bus interface diagram shown in Fig. 2.26 illustrates the connection of master and slave devices on the I2C bus.

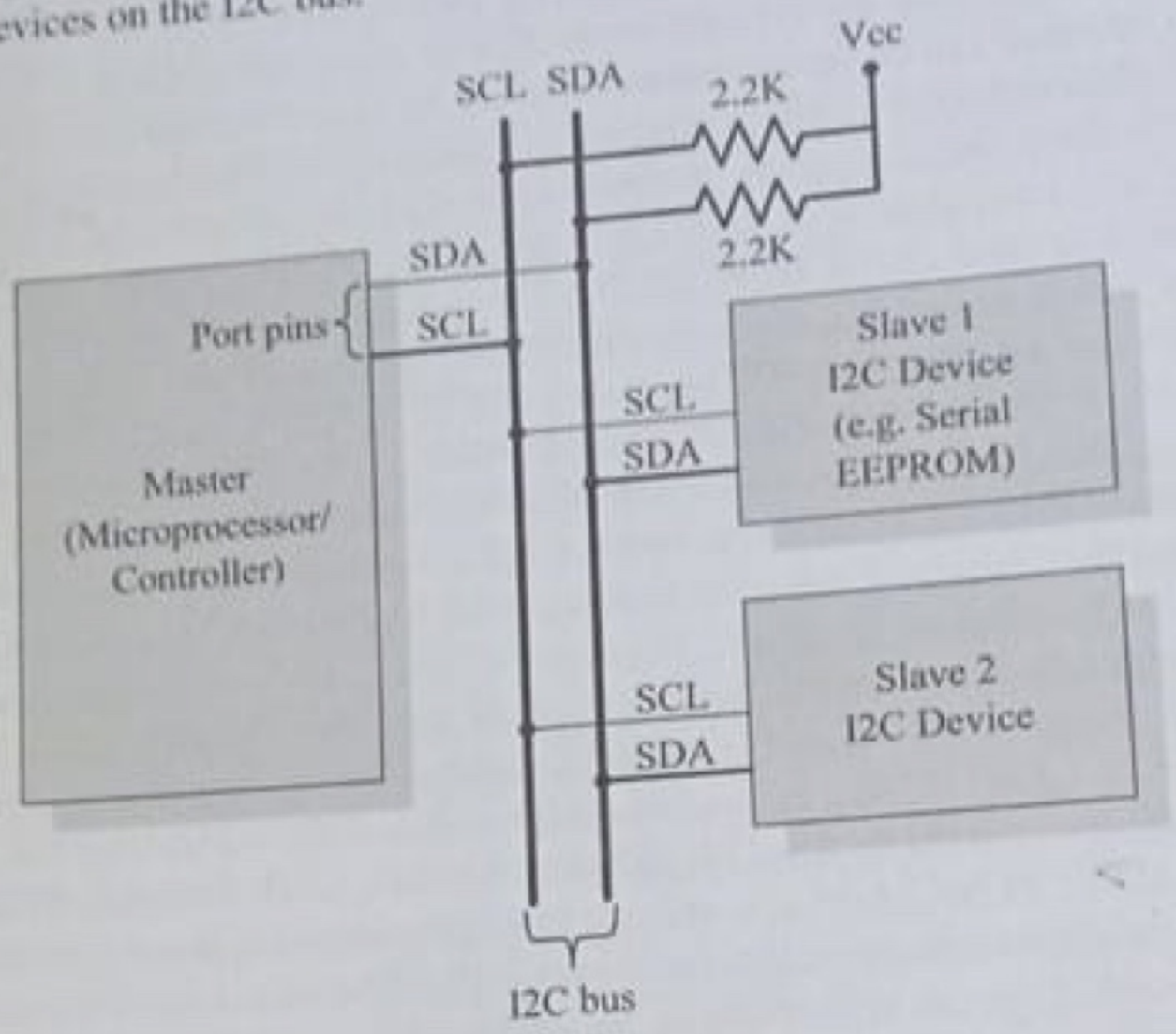

Fig. 2.26 I2C Bus Interfacing

The I2C bus interface is built around an input buffer and an open drain or collector transistor. When the bus is in the idle state, the open drain/collector transistor will be in the floating state and the output lines (SDA and SCL) switch to the 'High Impedance' state. For proper operation of the bus, both the bus lines should be pulled to the supply voltage (+5V for TTL family and +3.3V for CMOS family devices) using pull-up resistors. The typical value of resistors used in pull-up is 2.2K. With pull-up resistors, the output lines of the bus in the idle state will be 'HIGH'.

The address of a I2C device is assigned by hardwiring the address lines of the device to the desired logic level. The address to various I2C devices in an embedded device is assigned and hardwired at the time of designing the embedded hardware. The sequence of operations for communicating with an I2C slave device is listed below:

1. The master device pulls the clock line (SCL) of the bus to 'HIGH'
2. The master device pulls the data line (SDA) 'LOW', when the SCL line is at logic 'HIGH' (This is the 'Start' condition for data transfer)
3. The master device sends the address (7 bit or 10 bit wide) of the 'slave' device to which it wants to communicate, over the SDA line. Clock pulses are generated at the SCL line for synchronising the bit reception by the slave device. The MSB of the data is always transmitted first. The data in the bus is valid during the 'HIGH' period of the clock signal
4. The master device sends the Read or Write bit (Bit value = 1 Read operation; Bit value = 0 Write operation) according to the requirement
5. The master device waits for the acknowledgement bit from the slave device whose address is sent on the bus along with the Read/Write operation command. Slave devices connected to the bus compares the address received with the address assigned to them

may lead to the failure of the product in the market. Proper market study and cost benefit analysis should be carried out before taking a decision on the per-unit cost of the embedded product. From a designer/product development company perspective the ultimate aim of a product is to generate marginal profit. So the budget and total system cost should be properly balanced to provide a marginal profit.

The Product Life Cycle (PLC) Every embedded product has a product life cycle which starts with the design and development phase. The product idea generation, prototyping, Roadmap definition, actual product design and development are the activities carried out during this phase. During the design and development phase there is only investment and no returns. Once the product is ready to sell, it is introduced to the market. This stage is known as the Product Introduction stage. During the initial period the sales and revenue will be low. There won't be much competition and the product sales and revenue increases with time. In the growth phase, the product grabs high market share. During the maturity phase, the growth and sales will be steady and the revenue reaches at its peak. The Product Retirement/Decline phase starts with the drop in sales volume, market share and revenue. The decline happens due to various reasons like competition from similar product with enhanced features or technology changes, etc. At some point of the decline stage, the manufacturer announces discontinuing of the product. The different stages of the embedded products life cycle–revenue, unit cost and profit in each stage–are represented in the following Product Life-cycle graph (Fig. 3.1).

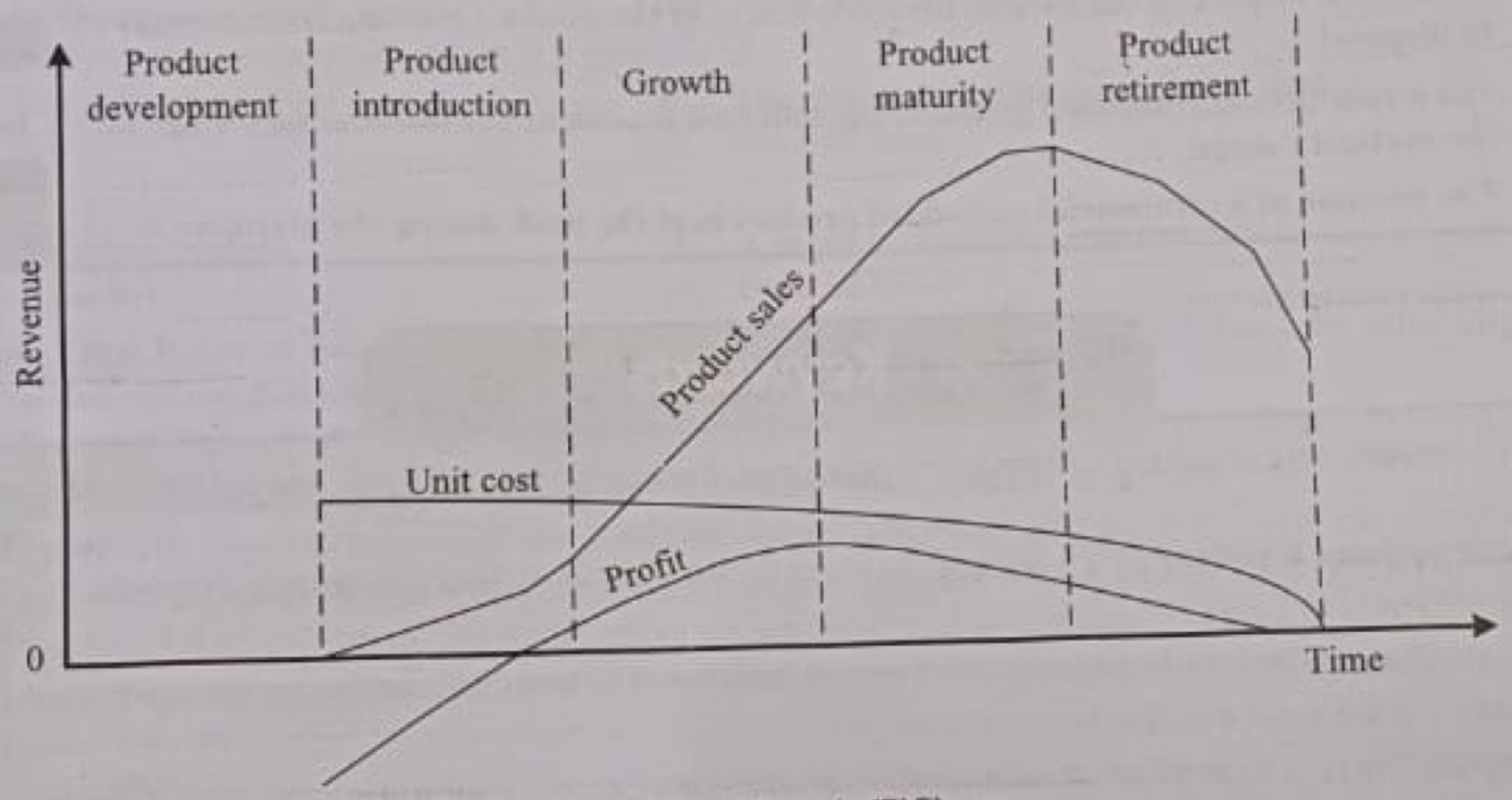

Fig. 3.1 Product life cycle (PLC) curve

From the graph, it is clear that the total revenue increases from the product introduction stage to the product maturity stage. The revenue peaks at the maturity stage and starts falling in the decline/retirement stage. The unit cost is very high during the introductory stage (a typical example is cell phone; if you buy a new model of cell phone during its launch time, the price will be high and you will get the same model with a very reduced price after three or four months of its launching). The profit increases with increase in sales and attains a steady value and then falls with a dip in sales. You can see a negative value for profit during the initial period. It is because during the product development phase there is only investment and no returns. Profit occurs only when the total returns exceed the investment and operating cost.

(5) Security
(6) Safety

3.2.1.1 Response Response is a measure of quickness of the system. It gives you an idea about how fast your system is tracking the changes in input variables. Most of the embedded systems demand fast response which should be almost Real Time. For example, an embedded system deployed in flight control application should respond in a Real Time manner. Any response delay in the system will create potential impact to the safety of the flight as well as the passengers. It is not necessary that all embedded systems should be Real Time in response. For example, the response time requirement for an electronic toy is not at all time-critical. There is no specific deadline that this system should respond within this particular timeline.

3.2.1.2 Throughput Throughput deals with the efficiency of a system. In general it can be defined as the rate of production or operation of a defined process over a stated period of time. The rates can be expressed in terms of units of products, batches produced, or any other meaningful measurements. In the case of a Card Reader, throughput means how many transactions the Reader can perform in a minute or in an hour or in a day. Throughput is generally measured in terms of 'Benchmark'. A 'Benchmark' is a reference point by which something can be measured. Benchmark can be a set of performance criteria that a product is expected to meet or a standard product that can be used for comparing other products of the same product line.

3.2.1.3 Reliability Reliability is a measure of how much % you can rely upon the proper functioning of the system or what is the % susceptibility of the system to failures.

Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR) are the terms used in defining system reliability. MTBF gives the frequency of failures in hours/weeks/months. MTTR specifies how long the system is allowed to be out of order following a failure. For an embedded system with critical application need, it should be of the order of minutes.

3.2.1.4 Maintainability Maintainability deals with support and maintenance to the end user or client in case of technical issues and product failures or on the basis of a routine system checkup. Reliability and maintainability are considered as two complementary disciplines. A more reliable system means a system with less corrective maintainability requirements and vice versa. As the reliability of the system increases the chances of failure and non-functioning also reduces, thereby the need for maintainability is also reduced. Maintainability is closely related to the system availability. Maintainability can be broadly classified into two categories, namely, 'Scheduled or Periodic Maintenance (preventive maintenance)' and 'Maintenance to unexpected failures (corrective maintenance)'. Some embedded products may use consumable components or may contain components which are subject to wear and tear and they should be replaced on a periodic basis. The period may be based on the total hours of the system usage or the total output the system delivered. A printer is a typical example for illustrating the two types of maintainability. An inkjet printer uses ink cartridges, which are consumable components and as per the printer manufacturer the end user should replace the cartridge after each '*n*' number of printouts to get quality prints. This is an example for '**Scheduled Periodic maintenance**'. If the paper feeding part of the printer fails the printer fails to print and it requires immediate repairs to rectify this problem. This is an example of '**Maintenance to unexpected failure**'. In both of the maintenances (scheduled and repair), the printer needs to be brought offline and during that time it will not be available for the user. Hence it is obvious that maintainability is simply an indication of the availability of the product for use. In any embedded system design, the ideal value for availability is expressed as

$$A_i = \text{MTBF}/(\text{MTBF} + \text{MTTR})$$

- Understand the Product Life Cycle (PLC)

No matter whether it is an embedded or a non-embedded system, there will be a set of charact describing the system. The non-functional aspects that need to be addressed in embedded system des commonly referred as quality attributes. Whenever you design an embedded system, the design shou into consideration of both the functional and non-functional aspects. The following topics give an o of the characteristics and quality attributes of an embedded system.

3.1 CHARACTERISTICS OF AN EMBEDDED SYSTEM

LO 1 Unders the characte describing a embedded

Unlike general purpose computing systems, embedded systems possess certain specific characteristics and these characteristics are unique to each embedded system. Some of the important characteristics of an embedded system are as follows:

(1) Application and domain specific
(2) Reactive and Real Time
(3) Operates in harsh environments

(4) Distributed
(5) Small size and weight
(6) Power concerns

3.1.1 Application and Domain Specific

If you closely observe any embedded system, you will find that each embedded system is designed to perform a set of defined functions and they are developed in such a manner to do the intended functions only. They cannot be used for any other purpose. It is the major criterion which distinguishes an embedded system from a general purpose computing system. For example, you cannot replace the embedded control unit of your microwave oven with your air conditioner's embedded control unit, because the embedded control units of microwave oven and airconditioner are specifically designed to perform certain specific tasks. Also, you cannot replace an embedded control unit developed for a particular domain say telecom with another control unit designed to serve another domain like consumer electronics.

3.1.2 Reactive and Real Time

As mentioned earlier embedded systems are in constant interaction with the Real world through sensors and user-defined input devices which are connected to the input port of the system. Any changes happening in the Real world (which is called an Event) are captured by the sensors or input devices in Real Time and the control algorithm running inside the unit reacts in a designed manner to bring the controlled output variables to the desired level. The event may be a periodic one or an unpredicted one. If the event is an unpredicted one then such systems should be designed in such a way that it should be scheduled to capture the events without missing them. Embedded systems produce changes in output in response to the changes in the input. So they are generally referred as Reactive Systems.

Real Time System operation means the timing behaviour of the system should be deterministic; meaning the system should respond to requests or tasks in a known amount of time. A Real Time system should not miss any deadlines for tasks or operations. It is not necessary that all embedded systems should be Real Time in operations. Embedded applications or systems which are mission critical, like flight control systems, Antilock Brake Systems (ABS), etc. are examples of Real Time systems. The design of an embedded Real time system should take the worst case scenario into consideration.

3.1.3 Operates in Harsh Environment

It is not necessary that all embedded systems should be deployed in controlled environments. The environment in which the embedded system deployed may be a dusty one or a high temperature zone or an area subject to vibrations and shock. Systems placed in such areas should be capable to withstand all these adverse operating conditions. The design should take care of the operating conditions of the area where the system is going to implement. For example, if the system needs to be deployed in a high temperature zone, then all the components used in the system should be of high temperature grade. Here we cannot go for a compromise in cost. Also proper shock absorption techniques should be provided to systems which are going to be commissioned in places subject to high shock. Power supply fluctuations, corrosion and component aging, etc. are the other factors that need to be taken into consideration for embedded systems to work in harsh environments.

3.1.4 Distributed

The term distributed means that embedded systems may be a part of larger systems. Many numbers of such distributed embedded systems form a single large embedded control unit. An automatic vending machine

is a typical example for this. The vending machine contains a card reader (for pre-paid vending systems), a vending unit, etc. Each of them are independent embedded units but they work together to perform the overall vending function. Another example is the Automatic Teller Machine (ATM). An ATM contains a card reader embedded unit, responsible for reading and validating the user's ATM card, transaction unit for performing transactions, a currency counter for dispatching/vending currency to the authorised person and a printer unit for printing the transaction details. We can visualise these as independent embedded systems. But they work together to achieve a common goal.

Another typical example of a distributed embedded system is the Supervisory Control And Data Acquisition (SCADA) system used in Control & Instrumentation applications, which contains physically distributed individual embedded control units connected to a supervisory module.

3.1.5 Small Size and Weight

Product aesthetics is an important factor in choosing a product. For example, when you plan to buy a new mobile phone, you may make a comparative study on the pros and cons of the products available in the market. Definitely the product aesthetics (size, weight, shape, style, etc.) will be one of the deciding factors to choose a product. People believe in the phrase "Small is beautiful". Moreover it is convenient to handle a compact device than a bulky product. In embedded domain also compactness is a significant deciding factor. Most of the application demands small sized and low weight products.

3.1.6 Power Concerns

Power management is another important factor that needs to be considered in designing embedded systems. Embedded systems should be designed in such a way as to minimise the heat dissipation by the system. The production of high amount of heat demands cooling requirements like cooling fans which in turn occupies additional space and make the system bulky. Nowadays ultra low power components are available in the market. Select the design according to the low power components like low dropout regulators, and controllers/ processors with power saving modes. Also power management is a critical constraint in battery operated application. The more the power consumption the less is the battery life.

3.2 QUALITY ATTRIBUTES OF EMBEDDED SYSTEMS

LO 2 Explain the non-functional requirements that needs to be addressed in the design of an embedded system

Quality attributes are the non-functional requirements that need to be documented properly in any system design. If the quality attributes are more concrete and measurable it will give a positive impact on the system development process and the end product. The various quality attributes that needs to be addressed in any Embedded System development are broadly classified into two, namely 'Operational Quality Attributes' and 'Non-Operational Quality Attributes'.

3.2.1 Operational Quality Attributes

The operational quality attributes represent the relevant quality attributes related to the Embedded System when it is in the operational mode or 'online' mode. The important quality attributes coming under this category are listed below:

(1) Response
(2) Throughput
(3) Reliability
(4) Maintainability

lighting controls, environmental controls, etc., are examples for applications which can make use of the ZigBee technology.

ZigBee PRO offers full wireless mesh, low-power networking capable of supporting more than 64,000 devices on a single network. It provides standardised networking designed to connect the widest range of devices, in any industry, into a single control network. ZigBee PRO offers an optional new and innovative feature, 'Green Power' to connect energy harvesting or self-powered devices into ZigBee PRO networks. The 'Green Power' feature of ZigBee PRO is the most eco-friendly way to power battery-less devices such as sensors, switches, dimmers and many other devices and allows them to securely join ZigBee PRO networks. ZigBee 3.0 delivers all the features of ZigBee while unifying the ZigBee application standards found in ZigBee devices. ZigBee 3.0 standard enables communication and interoperability among devices for smart homes, connected lighting, and other markets.

The specifications for ZigBee is developed and managed by the ZigBee alliance (www.zigbee.org), a non-profit consortium of leading semiconductor manufacturers, technology providers, OEMs and end-users worldwide.

2.4.2.8 General Packet Radio Service (GPRS), 3G, 4G, LTE General Packet Radio Service (GPRS), 3G, 4G and LTE are cellular communication technique for transferring data over a mobile communication network like GSM and CDMA. Data is sent as packets in GPRS communication. The transmitting device splits the data into several related packets. At the receiving end the data is re-constructed by combining the received data packets. GPRS supports a theoretical maximum transfer rate of 171.2kbps. In GPRS communication, the radio channel is concurrently shared between several users instead of dedicating a radio channel to a cell phone user. The GPRS communication divides the channel into 8 timeslots and transmits data over the available channel. GPRS supports Internet Protocol (IP), Point to Point Protocol (PPP) and X.25 protocols for communication. GPRS is mainly used by mobile enabled embedded devices for data communication. The device should support the necessary GPRS hardware like GPRS modem and GPRS radio. To accomplish GPRS based communication, the carrier network also should have support for GPRS communication. GPRS is an old technology and it is being replaced by new generation cellular data communication techniques like 3G (3rd Generation), High Speed Downlink Packet Access (HSDPA), 4G (4th Generation), LTE (Long Term Evolution) etc. which offers higher bandwidths for communication. 3G offers data rates ranging from 144Kbps to 2Mbps or higher, whereas 4G gives a practical data throughput of 2 to 100+ Mbps depending on the network and underlying technology.

2.5 EMBEDDED FIRMWARE

LO 5 Describe what embedded firmware is and its role in embedded systems

Embedded firmware refers to the control algorithm (Program instructions) and or the configuration settings that an embedded system developer dumps into the code (Program) memory of the embedded system. It is an un-avoidable part of an embedded system. There are various methods available for developing the embedded firmware. They are listed below.

(1) Write the program in high level languages like Embedded C/C++ using an Integrated Development Environment (The IDE will contain an editor, compiler, linker, debugger, simulator, etc. IDEs are different for different family of processors/controllers. For example, Keil micro vision3 IDE is used for all family members of *8051* microcontroller, since it contains the generic *8051* compiler C51).

(2) Write the program in Assembly language using the instructions supported by your application's target processor/controller.

The instruction set for each family of processor/controller is different and the program written in either of the methods given above should be converted into a processor understandable machine code before loading it into the program memory.

typical interfacing of devices in a Wi-Fi network. For communicating with devices over a Wi-Fi network, the device when its Wi-Fi radio is turned ON, searches the available Wi-Fi network in its vicinity and lists out the Service Set Identifier (SSID) of the available networks. If the network is security enabled, a password may be required to connect to a particular SSID. Wi-Fi employs different security mechanisms like Wired Equivalency Privacy (WEP) Wireless Protected Access (WPA), etc. for securing the data communication. Wi-Fi supports data rates ranging from 1Mbps to 1300Mbps (Growing towards higher rates as technology progresses), depending on the standards (802.11a/b/g/n/ac) and access/modulation method. Depending on the type of antenna and usage location (indoor/outdoor), Wi-Fi offers a range of 100 to 1000 feet.

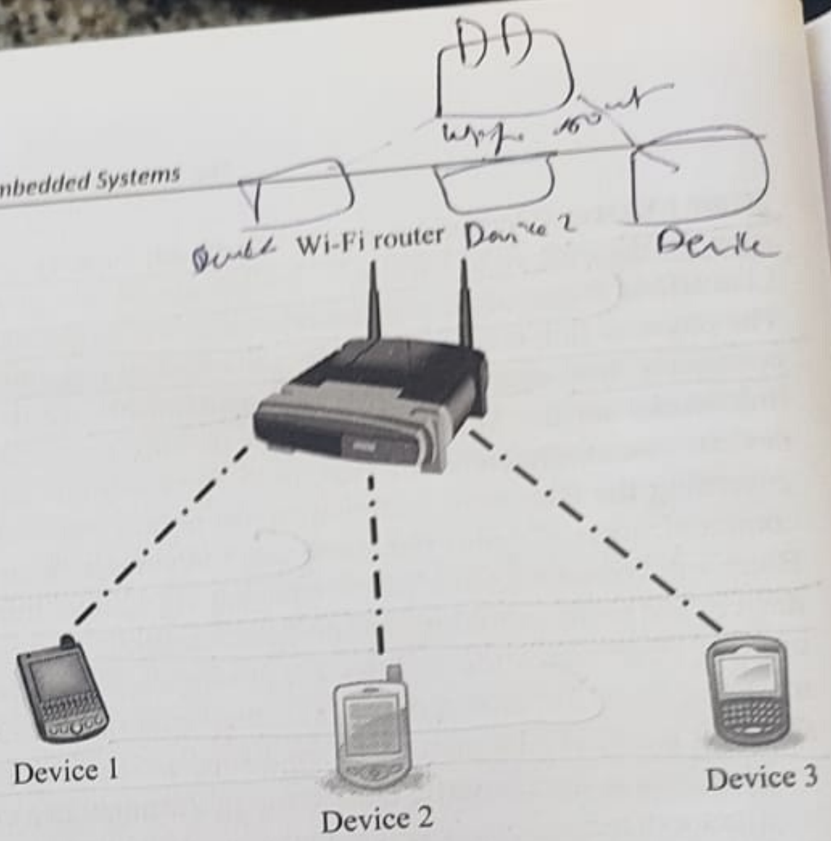

Fig. 2.33 Wi-Fi network

2.4.2.7 ZigBee ZigBee is a low power, low cost, wireless network communication protocol based on the IEEE 802.15.4-2006 standard. ZigBee is targeted for low power, low data rate and secure applications for Wireless Personal Area Networking (WPAN). The ZigBee specifications support a robust mesh network containing multiple nodes. This networking strategy makes the network reliable by permitting messages to travel through a number of different paths to get from one node to another.

ZigBee operates worldwide at the unlicensed bands of Radio spectrum, mainly at 2.400 to 2.484 GHz, 902 to 928 MHz and 868.0 to 868.6 MHz. ZigBee Supports an operating distance of up to 100 metres and a data rate of 20 to 250Kbps.

In the ZigBee terminology, each ZigBee device falls under any one of the following ZigBee device category.

ZigBee Coordinator (ZC)/Network Coordinator The ZigBee coordinator acts as the root of the ZigBee network. The ZC is responsible for initiating the ZigBee network and it has the capability to store information about the network.

ZigBee Router (ZR)/Full function Device (FFD) Responsible for passing information from device to another device or to another ZR.

ZigBee End Device (ZED)/Reduced Function Device (RFD) End device containing ZigBee functionality for data communication. It can talk only with a ZR or ZC and doesn't have the capability to act as a mediator for transferring data from one device to another.

The diagram shown in Fig. 2.34 gives an overview of ZC, ZED and ZR in a ZigBee network.

ZigBee is primarily targeting application areas like home & industrial automation, energy management, home control/security, medical/patient tracking, logistics & asset tracking and sensor networks & active RFID. Automatic Meter Reading (AMR), smoke detectors, wireless telemetry, HVAC control, heating control,

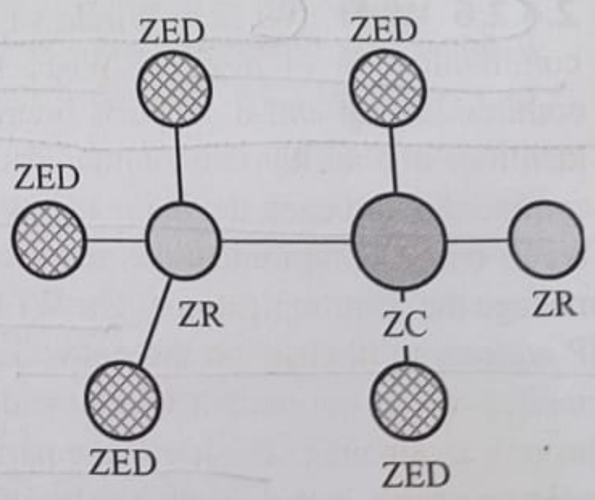

Fig. 2.34 A ZigBee network model

6. The slave device with the address requested by the master device responds by sending an acknowledge bit (Bit value = 1) over the SDA line
7. Upon receiving the acknowledge bit, the Master device sends the 8bit data to the slave device over SDA line, if the requested operation is 'Write to device'. If the requested operation is 'Read from device', the slave device sends data to the master over the SDA line
8. The master device waits for the acknowledgement bit from the device upon byte transfer complete for a write operation and sends an acknowledge bit to the Slave device for a read operation
9. The master device terminates the transfer by pulling the SDA line 'HIGH' when the clock line SCL is at logic 'HIGH' (Indicating the 'STOP' condition)

The first generation I2C devices were designed to support data rates only up to 100kbps. Over time there have been several additions to the specification so that there are now five operating speed categories; Namely, Standard mode (Sm - Data rate up to 100kbit/sec), Fast mode (Fm - Data rate up to 400kbit/sec), Fast mode Plus (Fm+ - Data rate up to 1Mbit/sec), and High-speed mode (Hs-mode - Data rate up to 3.4Mbit/sec) and an Ultra Fast-mode (UFm), with a bit rate up to 5 Mbit/s for unidirectional I2C bus.

2.4.1.2 Serial Peripheral Interface (SPI) Bus

The Serial Peripheral Interface Bus (SPI) is a synchronous bi-directional full duplex four-wire serial interface bus. The concept of SPI was introduced by Motorola. SPI is a single master multi-slave system. It is possible to have a system where more than one SPI device can be master, provided the condition only one master device is active at any given point of time, is satisfied. SPI requires four signal lines for communication. They are:

Master Out Slave In (MOSI): Signal line carrying the data from master to slave device. It is also known as Slave Input/Slave Data In (SI/SDI)

Master In Slave Out (MISO): Signal line carrying the data from slave to master device. It is also known as Slave Output (SO/SDO)

Serial Clock (SCLK): Signal line carrying the clock signals

Slave Select (SS): Signal line for slave device select. It is an active low signal

The bus interface diagram shown in Fig. 2.27 illustrates the connection of master and slave devices on the SPI bus.

The master device is responsible for generating the clock signal. It selects the required slave device by asserting the corresponding slave device's slave select signal 'LOW'. The data out line (MISO) of all the slave devices when not selected floats at high impedance state.

The serial data transmission through SPI bus is fully configurable. SPI devices contain a certain set o registers for holding these configurations. The serial peripheral control register holds the various configuratio parameters like master/slave selection for the device, baudrate selection for communication, clock sign control, etc. The status register holds the status of various conditions for transmission and reception.

SPI works on the principle of 'Shift Register'. The master and slave devices contain a special shift regis for the data to transmit or receive. The size of the shift register is device dependent. Normally it is a multi of 8. During transmission from the master to slave, the data in the master's shift register is shifted ou the MOSI pin and it enters the shift register of the slave device through the MOSI pin of the slave de At the same time the shifted out data bit from the slave device's shift register enters the shift register o master device through MISO pin. In summary, the shift registers of 'master' and 'slave' devices fo circular buffer. For some devices, the decision on whether the LS/MS bit of data needs to be sent out f configurable through configuration register (e.g. LSBF bit of the SPI control register for Motorola's 68 controller).

When compared to I2C, SPI bus is most suitable for applications requiring transfer of data in 'str The only limitation is SPI doesn't support an acknowledgement mechanism.

as the non-heritable variation. For an embedded system, the quality attribute 'Evolvability' refers to the ease with which the embedded product (including firmware and hardware) can be modified to take advantage of new firmware or hardware technologies.

3.2.2.3 Portability Portability is a measure of 'system independence'. An embedded product is said to be portable if the product is capable of functioning 'as such' in various environments. target processors/ controllers and embedded operating systems. The ease with which an embedded product can be ported on to a new platform is a direct measure of the re-work required. A standard embedded product should always be flexible and portable. In embedded products, the term 'porting' represents the migration of the embedded firmware written for one target processor (e.g. Intel x86) to a different target processor (say an ARM Cortex M3 processor from Freescale). If the firmware is written in a high level language like 'C' with little target processor-specific functions (operating system extensions or compiler specific utilities), it is very easy to port the firmware for the new processor by replacing those 'target processor-specific functions' with the ones for the new target processor and re-compiling the program for the new target processor generates the new target processor-specific Re-compiling the program for the new target processor-specific machine codes. If the firmware is written in Assembly Language for a particular family of processor (say x86 family), it will be very difficult to translate the assembly language instructions to the new target processor specific language and so the portability is poor.

If you look into various programming languages for application development for desktop applications, you will see that certain applications developed on certain languages run only on specific operating systems and some of them run independent of the desktop operating systems. For example, applications developed using Microsoft technologies (e.g. Microsoft Visual C++ using Visual studio) is capable of running only on Microsoft platforms and may not function on other operating systems; whereas applications developed using 'Java' from Sun Microsystems works on any operating system that supports Java standards.

3.2.2.4 Time-to-Prototype and Market Time-to-market is the time elapsed between the conceptualisation of a product and the time at which the product is ready for selling (for commercial product) or use (for non-commercial products). The commercial embedded product market is highly competitive and time to market the product is a critical factor in the success of a commercial embedded product. There may be multiple players in the embedded industry who develop products of the same category (like mobile phone, portable media players, etc.). If you come up with a new design and if it takes long time to develop and market it, the competitor product may take advantage of it with their product. Also, embedded technology is one where rapid technology change is happening. If you start your design by making use of a new technology and if it takes long time to develop and marke the product, by the time you market the product, the technology might have superseded with a ne technology. Product prototyping helps a lot in reducing time-to-market. Whenever you have a produ idea, you may not be certain about the feasibility of the idea. Prototyping is an informal kind of rap product development in which the important features of the product under consideration are develope The time to prototype is also another critical factor. If the prototype is developed faster, the act estimated development time can be brought down significantly. In order to shorten the time to prototy make use of all possible options like the use of off-the-shelf components, re-usable assets, etc.

3.2.2.5 Per Unit Cost and Revenue Cost is a factor which is closely monitored by both end user (t who buy the product) and product manufacturer (those who build the product). Cost is a highly sens factor for commercial products. Any failure to position the cost of a commercial product at a nominal

where A_i = Availability in the ideal condition, MTBF = Mean Time Between Failures, and MTTR = Mean Time To Repair

3.2.1.5 Security 'Confidentiality', 'Integrity', and 'Availability' (The term 'Availability' mentioned here is not related to the term 'Availability' mentioned under the 'Maintainability' section) are the three major measures of information security. Confidentiality deals with the protection of data and application from unauthorised disclosure. Integrity deals with the protection of data and application from unauthorised modification. Availability deals with protection of data and application from unauthorised users. A very good example of the 'Security' aspect in an embedded product is a Personal Digital Assistant (PDA). The PDA can be either a shared resource (e.g. PDAs used in LAB setups) or an individual one. If it is a shared one there should be some mechanism in the form of a user name and password to access into a particular person's profile–This is an example of 'Availability'. Also all data and applications present in the PDA need not be accessible to all users. Some of them are specifically accessible to administrators only. For achieving this, Administrator and user levels of security should be implemented –An example of Confidentiality. Some data present in the PDA may be visible to all users but there may not be necessary permissions to alter the data by the users. That is Read Only access is allocated to all users–An example of Integrity.

3.2.1.6 Safety 'Safety' and 'Security' are two confusing terms. Sometimes you may feel both of them as a single attribute. But they represent two unique aspects in quality attributes. Safety deals with the possible damages that can happen to the operators, public and the environment due to the breakdown of an embedded system or due to the emission of radioactive or hazardous materials from the embedded products. The breakdown of an embedded system may occur due to a hardware failure or a firmware failure. Safety analysis is a must in product engineering to evaluate the anticipated damages and determine the best course of action to bring down the consequences of the damages to an acceptable level. As stated before, some of the safety threats are sudden (like product breakdown) and some of them are gradual (like hazardous emissions from the product).

3.2.2 Non-Operational Quality Attributes

The quality attributes that needs to be addressed for the product 'not' on the basis of operational aspects are grouped under this category. The important quality attributes coming under this category are listed below.

(1) Testability & Debug-ability
(2) Evolvability
(3) Portability
(4) Time to prototype and market
(5) Per unit and total cost.

3.2.2.1 Testability & Debug-ability Testability deals with how easily one can test the design, application and by which means he/she can test it. For an embedded product, testability is applicable to both the embedded hardware and firmware. Embedded hardware testing ensures that the peripherals and the total hardware functions in the desired manner, whereas firmware testing ensures that the firmware is functioning in the expected way. Debug-ability is a means of debugging the product as such for figuring out the probable sources that create unexpected behaviour in the total system. Debug-ability has two aspects in the embedded system development context, namely, hardware level debugging and firmware level debugging. Hardware debugging is used for figuring out the issues created by hardware problems whereas firmware debugging is employed to figure out the probable errors that appear as a result of flaws in the firmware.