UNIT 5

Design Principles for WSNs, Gateway Concepts, Single-node Architecture, Operating Systems

Syllabus

Design Principles for WSNs, Gateway Concepts Need for Gateway, WSN to Internet Communication, and Internet to WSN Communication.

Single-node Architecture, Hardware Components & Design Constraints, Operating Systems and Execution Environments, Introduction to TinyOS and nesC.

Learning Objectives

- ✓ Various Design Principles of Wireless Sensor Networks
- ✓ Gateway and its Need
- ✓ WSN to Internet Communication
- ✓ Various Hardware Components for Wireless Sensor Nodes
- ✓ Transceiver and its Characteristics
- ✓ Categories of Sensors
- ✓ Energy Consumption in Sensor Nodes, Microcontrollers
- ✓ Radio Transceiver and its Models
- ✓ Introduction to Embedded Operating Systems
- ✓ Various Programming Paradigms
- ✓ Brief Introduction to TinyOS and nesC.

Introduction

Basic design principles have been developed, which can be useful when designing networking protocols such as Distributed organization, In-network processing, Adaptive fidelity and accuracy, Data centricity etc. Gateway is a device used for translating information among different network data formats. They provide a physical connection to the internet in order to enable the wireless sensor networks exchange the data with other information devices.

In wireless sensor nodes, the hardware components are selected based on the requirements of an application. These requirements represent the key or deciding factor with respect to size, costs and energy consumption of the nodes. A sensor node consists of hardware components such as Controller, Memory, Sensors and Actuators, Communication device and Power Supply.

TinyOS is an open-source operating system designed for wireless embedded sensor networks. The intent of TinyOS is to support applications of sensor networks on hardware platforms. nesC is an imperative language which is an extension of C. It is used to support tiny OS through various components.

PART-B ESSAY QUESTIONS WITH SOLUTIONS

5.1 DESIGN PRINCIPLES FOR WSNs

Q9. Explain in brief about design principles of wireless sensor networks.

Model Paper-I, Q10(a)

Answer :

Design Principles of Wireless Sensor Networks

The design principles for wireless sensor networks are as follows,

1. **Distributed Organization**

To achieve the optimization goals, a centralized entity is required to control the operations of the network which might reveal the points of failure. To avoid this, WSN uses distributed algorithms which cooperatively organizes the network without creating a centralized entity. This approach is called as self-organization (or) distributed organization.

2. **In-network Processing**

In distributed organization, nodes execute programs, handle flow of packets and take decisions but limit the information to the network only. Therefore in-network processing is used to extend this information with concrete data.

Some of the example techniques of in-network processing are,

(a) Aggregation

(b) Distributed source coding and distributed compression

(c) Distributed and collaborative signal processing

(d) Mobile code/agent-based networking.

3. **Data Centricity**

In a wireless sensor network, the property of concerning about data rather than identity of the sensor node is called as data centric networking. This means, in an application, requests are addressed according to the data it carries.

In comparison of traditional communication networks, data centric networking allows very difficult architecture. Participating nodes are defined implicitly in such away to avoid addressing about various relationships like one-to-one, one-to-many etc.

4. **Exploit Location Information**

For various applications, it is necessary to get informed about various location of sensor nodes on which the information is available. Therefore, some techniques must be used in order to exploit location information.

5. **Adaptive Fidelity and Accuracy**

The computational results must be allotted with certain degree of exactness (accuracy) for the entire network to avoid high consumption of energy. This property makes the wireless network energy efficient.

6. **Exploit Activity Patterns**

When an event happens, large number of sensors detect it which causes burst of traffic and causes event shower effect. Therefore, an intermediate protocol must be designed in such a way that it can control the flow of traffic through various modes depending on the activity pattern.

7. **Exploit Heterogeneity**

In a wireless sensor networks, sensor nodes could be heterogeneous either by evolution heterogeneity means that few sensor nodes perform more tasks whereas, some perform less tasks. Similarly, some nodes uses extra energy and some save the energy.

Construction heterogeneity means that some nodes require large batteries, some can reach further communication devices etc. Therefore, a balanced protocol must be developed to manage this heterogeneity.

8. ***Component-based Protocol Stacks and Cross Layer Optimization***

The protocol must use selective number of components (rather than all) that should be made available on all sensor nodes.

Cross layer exchange of information provides protocol optimization without any issue.

Q10. Explain in brief about requirements for WSN service interfaces.

Answer :

Requirements for WSN Service Interfaces

A service interface should be perform the following functionalities,

1. **Support for Synchronous Interactions**

Simple request/response interactions should be managed by setting certain parameters with a measured value for getting quick responses irrespective of its periodicity.

2. **Support for Asynchronous Interactions**

At a particular point, if the condition of the request node becomes true, it should be informed to the request node through a notification.

3. **Well-define Addressing Nodes**

Addressing nodes should be well-defined through several ways irrespective of type of interactions. This can be done by considering an explicit enumeration of requested peers.

To allow data-centric networking, peers are defined implicitly by a membership function of group of nodes.

4. **Accessibility to In-networking Processing**

Service interface should be able to specify in-networking processing method in the case when an operation read values from the network by making use of each of the node group.

Application can also use an ideal in-networking processing functions by infusing its own method into the network.

5. **Use of Aggregation Function**

Use of aggregation function i.e., limiting the number of group members provide the results with required accuracy and with expected consumption of energy.

6. **Explicit Information of Timeliness Requirement**

Service interface should provide a detailed information on the requirements of time-based delivery of data i.e., quick delivery of data results in increased energy cost whereas slow delivery results in reduced cost.

Service interface should be able to access timing, location and network status information.

7. **Explicit Information on Capabilities of Network**

The capabilities of node (or) network should be explicitly described inorder to support uninterrupted connection between nodes and can also provide access to unknown network.

It should provide security related requirements and properties.

5.2 GATEWAY CONCEPTS

5.2.1 Need for Gateway

Q11. Define gateway. Explain in brief about the need for gateways.

Model Paper-II, Q10(a)

Answer :

Gateway

Gateway is a device used for translating information among different network data formats. TCP/IP can be translated to AppleTalk by using the gateway. Hence, they are used to promote the communication between computers supporting TCP/IP with Apple brand computers. The operation of gateways status at the lower level i.e., at application layer where it removes the information till it reaches the target level. At this point, it will recollect the information for distributing among the networks. Two types of gateways are used for internet working, they are,

(a) **Transport Gateway:** This is used for connecting two networks in the transport layer.

(b) **Application Gateway:** This gateway is used for connecting two different parts of an application in the application layer.

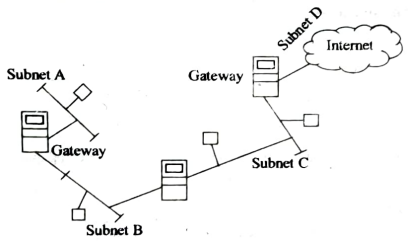

Figure: Connection of Networks Using Gateways

Need for Gateways

Gateways provide a physical connection to the internet in order to enable the wireless sensor networks exchange the data with other information devices (such as mobile device). Gateway is a relatively straight forward network that functions on the layers like physical, MAC and link layer. That is, either the mobile device or the gateway is equipped with the radio transceiver or the fewer nodes present in wireless sensor network support standard wireless communication technologies like IEEE 802.11. Both the options are beneficial on the basis of application and use-case.

The logical design of gateways is quite challenging. A gateway can be considered as a simple router between internet and sensor network which would imply the use of internet protocols inside the sensor network. However, since the wireless sensor networks require specific heavily optimized protocols, therefore the simple router cannot be sufficed by the gateways adequately.

Finally, the other possibility is to design the gateway in the form of application-level gateway. If a gateway is designed on the basis of the application level information, it will have to decide its own action. The problems associated with gateways can be estimated according to the source from where the communication is started.

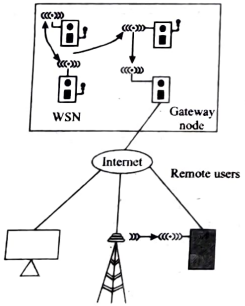

Figure: WSN with Gateway Node

Q12. Write short notes on WSN tunneling.

Answer :

WSN tunneling is a process where a large virtual network is created from various separate parts in which all protocol messages are transferred from one WSN to the other WSN using internet as a transport network. Though this process appears to be interesting and simple, it must be ensured that the virtual link between two gateway nodes is not mixed up with the real link because it may confuse the protocols that are dependent on the physical properties of a communication link.

It is not necessary for these tunnels, which are even identified as the means for intermediate interconnection of WSN's, to be in the form of fixed network connection, they can also be in the form of mobile nodes that can be carried out by people.

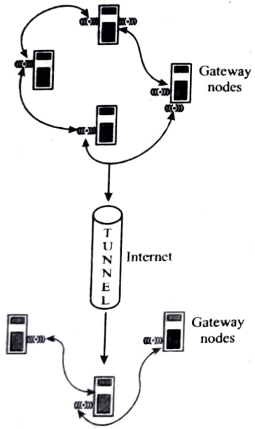

Figure: WSN Tunneling

5.2.2 WSN to Internet Communication

Q13. Explain in brief about WSN to Internet communication.

Answer :

There are various gateways present in the network which makes it difficult for the sensor nodes to choose the required route/gateway in order to reach the destination. Besides this, complexity arises when there are various gateways each capable of enabling communication between the nodes and IP networking. This problem can be overcome by constructing an IP overlay network above the sensor network.

But, even after constructing an IP overlay, there is a possibility of few problems to arise. It is quite difficult for the sensor nodes to identify the address of the internet host to which the message is to be sent. However, it becomes necessary for the sensor node to include enough information in its protocol even if it does not have the necessity to process the IP protocol. This information is then retrieved by the gateway which is later translated into IP packet. The gateway here, performs the tasks identical to the Network Address Translation (NAT) device.

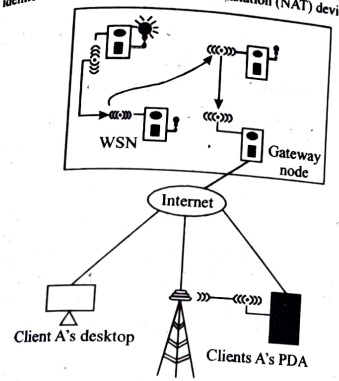

Figure: WSN to Internet Communication

5.2.3 Internet to WSN Communication

Q14. Discuss in brief about how communication is done from internet to WSN communication.

Answer :

When an internet based entity tries to access services of WSN, it must be capable to communicate with the WSN. It can also be a part of WSN. If the requesting entity is very far and is unable to communicate with any of the sensor nodes, then a gateway node can be used for assistance. To continue the communication, the below information is required.

(i) Whether any sensor network exists at the desired location.

(ii) Existence of gateway node

(iii) Access to actual services.

Addressing a sensor node becomes difficult if the data provided by it is irrelevant and if it does not have even an IP address. So, the requesting terminal can send the properly formatted request to the gateway which is an application level gateway or atleast a proxy for individual/set of a sensor node that have the ability to respond. This gateway will then translate the request into intra sensor network protocol interactions. It is assumed that there exists an application level protocol which can be used by the remote requester and gateway. It is also assumed that it is most relevant so that it can be used for communication over the internet rather than actual sensor network protocols. Then the gateway can mask a data centric data exchange in the network following the identity centric exchange in the internet, for example. Such an application level protocol not much clearly provides solution for our problem, rather so called web service protocols provide what services and way of accessing them, more clearly as well from far.

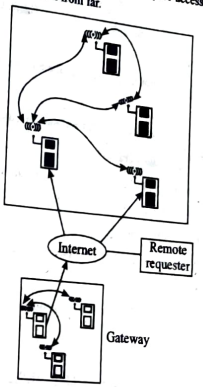

Figure: Internet to WSN Communication

5.3 SINGLE-NODE ARCHITECTURE

5.3.1 Hardware Components

Q15. Explain in detail about the hardware components for a wireless sensor node.

Answer : Model Paper-I, Q10(b)

The hardware components for a wireless sensor nodes are selected based on the requirements of an application. These requirements represents the key or deciding factor with respect to size, costs and energy consumption of the nodes.

Basically, a sensor node consists of the following five hardware components,

1. Controller
2. Memory
3. Sensors and Actuators
4. Communication device
5. Power supply.

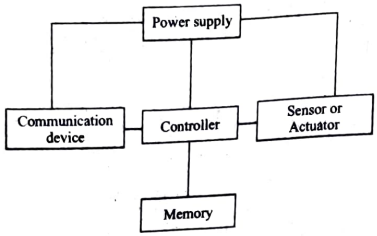

Figure: Overview of Hardware Components

1. **Controller**

   A controller is used for processing all the relevant data that are efficient to execute arbitrary code.

2. **Memory**

   It is used for storing codes and intermediate data. Generally, distinct types of memory are provided.

3. **Sensors and Actuators**

   These are the devices that forms the actual interface to the physical world. They have the capability of observing or controlling physical parameters of the environment available.

4. **Communication Device**

   It is a device that can turn nodes into a network in order to send and receive data in a wireless channel.

5. **Power Supply**

   Energy is the basic need of a sensor node. It is difficult to provide through tethered power supply. Hence, some kind of source such as batteries are required to supply energy.

   Also, some form of recharging can be provided, through which energy can be obtained from the environment such as solar energy. Example of such power supply can be solar cells.

Q16. What are the differences between microprocessors and microcontroller.

Answer :

	Microprocessors		Microcontrollers
1.	In microprocessor, I/O ports, timers, ROM, RAM are considered as external devices to the processor.	1.	In microcontroller, I/O ports, timers, ROM, RAM are available on a single chip.
2.	Memory to memory transfer is not possible.	2.	Memory to memory transfer is possible
3.	Bit manipulations are very less in microprocessors.	3.	Bit manipulations are very high in microcontrollers.
4.	In microprocessors special In, Out instructions are needed to transfer data between processor and external devices.	4.	Data transfer between microcontroller and external device is possible with 'MOV' instruction.
5.	ROM, RAM capacities are higher than that of microcontrollers.	5.	ROM, RAM capacities are less.
6.	By the execution of 'PUSH' instruction, the SP (Stack Pointer) is decremented.	6.	By the execution of 'PUSH' instruction, the SP (Stack Pointer) is incremented.
7.	The 'SP' will be incremented by performing POP operation.	7.	The 'SP' will be decremented by performing POP operation.
8.	Sign and zero flags are available.	8.	Sign and zero flags are not available.
9.	Clock frequency is less than microcontrollers.	9.	Clock frequency in greater than microprocessors.

Q17. Compare FPGA's and ASIC's.

Answer :

	FPGA		ASIC
1.	FPGA (Field-Programmable Gate Array) is a semiconductor device which contains programmable logic components (called "logic blocks") and programmable interconnects. Example: Decoders, Logic gates, etc.	1.	ASIC (Application-Specific Integrated Circuit) is an integrated circuit which is designed for a specific use, but not for general-purpose use. Example: RAM, ROM etc.
2.	Its design consumes more power.	2.	Its design consumes less power.
3.	Its design cycle is smaller and simpler, because most of the routing, placement, and timing are handled by software.	3.	Its design cycle is relatively larger and complex than FGPA'S design.
4.	Tools used for FPGA designs are cheaper.	4.	Tools used for ASIC designs are costly than FPGA.
5.	For small applications, FPGA products are cost effective.	5.	For complex and large volume designs, products of ASIC are cheaper than FPGA products.
6.	It has low speed.	6.	It has high speed.
7.	It can be reused, by simply reprogramming.	7.	It is application specific, hence cannot be reused.

Q18. What is a transceiver? Explain the tasks and characteristics of transceiver.

Answer :

Transceiver

Model Paper-II, Q10(b)

A transceiver is a device which is capable of performing the tasks of both a transmitter and a receiver. In other words, it is the combination of a transmitter and a receiver.

In a wireless communication, a sensor node requires both a transmitter and a receiver. Because, it is necessary to convert a bit stream (a sequence of bytes/frame obtained from microcontroller) into a radio waves, and vice versa.

Practically, using of both transmitter and receiver as a separate component will make a network complex.

Hence, this complexity can be reduced by using a single component called transceiver.

But in a wireless network, most of the situations does not support transmitting and receiving operations at the same time, due to which most of the time half-duplex operation is noticed.

Tasks and Characteristics of Transceiver

There are many characteristics which are to be considered while selecting transceiver. Some of them are as follows.

1. **Servicing the Upper Layer**

   The receiver needs to offer services such as packet transmission, byte interface, or bit interface to its upper layer, most probably to MAC layer.

2. **Data Rates**

   The gross data rate is specified by the carrier frequency and bandwidth that is used with both modulation and coding. For example, data rates can be few tens of kilobits/sec, which is considered to be less than that of in broadband wireless communication. But it is sufficient for WSN. Change in symbol rate and different modulation can be used to obtain different data rates.

3. **Coding**

   Selection of several coding schemes are allowed by some of the transceivers.

4. **Modulation**

   Various modulations such as ASK, FSK, on/off keying, or any of them which are similar are supported by the transceiver.

5. **Carrier Frequency and Multiple Channels**

   Transceivers can be used for different carrier frequencies to match the requirements of application and regulatory restrictions. Transceivers need to provide help by allowing to select among several carrier frequencies. So that, the congestion problem can be reduced in dense networks. These channels are appropriate for certain MAC protocols.

6. Gain

The gain can be obtained by dividing the output signal power by the input signal power. It is measured in terms of dB.

An amplifier with high gain are likely to obtain good energy efficiency.

7. Noise Figure

Noise Figure (NF) of an element can be obtained by dividing the SNR (Signal-to-Noise Ratio) at the element's input, SNR_i, by the SNR at the element's output, SNR_o. It is given by,

$$NF = \frac{SNR_i}{SNR_o}$$

8. Transmission Power Control

Certain transceivers have the ability of controlling the transmission power, where as certain need some external circuitry for this season. Generally, power levels are available only in discrete number. From this, the actual transmission of power can be selected. The maximum output power can be known by regulations.

9. Voltage Range

A transceiver requires a particular range of voltage for its operation. But, if the voltage supplied is not sufficient, then a voltage stabilization circuitry is required for the transceiver to work.

10. Frequency Stability

The frequency stability defines the degree of variation from nominal center frequencies. This variation is due to the environmental changes such as temperature or pressure observed in the oscillators. In some cases, poor frequency stability can even disconnect the communication links.

Q19. Explain the structure of transceivers.

Answer :

The structure of a transceiver mainly consists of two parts, they are,

1. Radio Frequency (RF) Front End

This part of the transceiver performs the analog signal processing within the specified radio frequency band.

2. Base Band Processor

This part of the transceiver performs the processings on all types of signals within the specified digital domain. It also interacts with other digital circuitry such as the processor within a sensor node.

The conversion of frequency is carried out between these two parts. The front end of the transceiver structure is shown in the figure below.

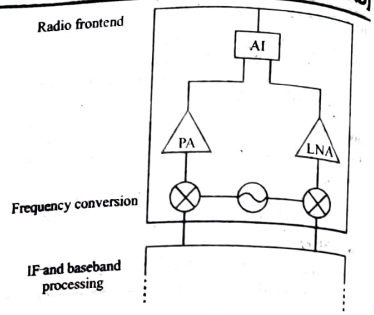

Figure: Overview of RF Front End

The RF front end structure consists of following elements,

(i) PA

The task of PA (Power Amplifier) is to accept the upconverted signals received from the Intermediate Frequency (IF) or baseband part. And it performs amplification, and then transmit it over the antenna.

(ii) LNA

The task of LNA (Low Noise Amplifier) is to amplify the incoming signals upto a level appropriate for further processing, without any significant reduction in SNR. The range of powers of the incoming signal differs from very weak signals (obtained from nodes near to the boundary of reception) to very strong signals (obtained from neighbouring nodes). This range value is upto 100 dB. LNA is always active, and consumes some part of transceiver's energy.

(iii) Frequency Conversion

The conversion of frequency from RF spectrum to intermediate frequencies (or to the baseband) is done using the components such as oscillators or voltage-controlled oscillators and mixers.

Q20. Write in detail the four operational states of transceiver.

Answer :

Operational States of Transceiver

A transceiver of consists of four operational states. They are as follows,

(i) Transmit

In this state, the transmitting part of a transceiver (i.e., transmitter) will be active. Suppose if a node is in transit state then it sends data to its neighbouring nodes.

(ii) Receive

In this state, the receiving part of a transceiver (i.e., receiver) will be active. Suppose if a node is in receive state then it is always ready to receive data from its neighbouring nodes.

(iii) Idle

In this state, the receiver will be ready to receive but it will not receive anything at that moment. Almost all the parts of a receiver will be active where as remaining parts will be switched off.

(iv) Sleep

In this state, all the important parts of a transceiver will be switched off. In this sleep state there are various other sub states.

These substates usually depend upon the number of switched off parts in a transceiver and they differ accordingly.

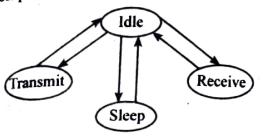

Figure: Transceiver States

So, a transceiver can be in any of these four states as it operates within these states only. A transceiver can easily move from one state to any other state.

Q21. Discuss briefly about the advanced concepts for radio communication.

Answer :

Some of the advanced concepts for radio communication are as follows,

1. Spread-Spectrum Transceivers (SST)

SST overcomes the interface problems in ASK or FSK. However, they face complex hardware and high price issues due to which they failed to gain much importance in WSNs.

2. Ultra Wide-Band Communication

Ultra Wide Band (UWB) is a large bandwidth that transmits digital sequence as short impulses. These impulses are spread over a large spectrum ranging from a few hertz to several GHz. However, such a large bandwidth can overlap with the spectrum of conventional radio system. Since it is widely spread, a small transmission power is sufficient. UWB communication are impermeable to narrow band radio waves. They are capable of measuring the distances accurately.

3. Wakeup Radio

A node is said to be a wakeup receivers when it is turned on only when the packet starts towards it. It is necessary to turn on the node when it is only waiting for the packet.

The purpose of the wakeup receivers is to wakeup only the necessary receiver without wasting much of the power. The wakeup receivers overcomes the problems related to WSNs. The main receiver is waked up by specifying the address of the intended receiver at the beginning of the message packet.

Q22. What are the three categories of sensors?

Answer : **Model Paper-I, Q11(a)**

Sensors are categorized into three groups. They are,

1. Passive, omnidirectional sensors
2. Passive, narrow-beam sensors
3. Active sensors.

1. Passive, Omnidirectional Sensors

This group of sensors are used for measuring a physical quantity at the point of the sensor node without the actual manipulation of the environment by active probing, in this context they are passive.

Some of these sensors are self-powered, this means they can recharge themselves by obtaining energy from the environment. They require energy only for the amplification of analog signal. The idea of direction is not included in these measurements.

Examples

Thermometer, smoke detectors, light sensors, humidity, mechanical stress tension in materials.

2. Passive, Narrow-beam Sensors

This group of sensors are passive, as well as involves well-defined idea of direction in their measurement.

Example

Camera is a good example. It can take measurement in a specified direction and can be rotated when required.

3. Active Sensors

This group of sensors are capable of manipulating the environment by active probing.

Examples

Radar or sonar sensor, or seismic sensor. These are capable of generating shock waves by small explosions.

Q23. Explain how energy is stored using batteries in power supply of sensor node.

Answer :

Energy can be stored using batteries in order to supply power for untethered wireless sensor nodes in the required form. A battery can be either non rechargeable called primary battery or rechargeable called secondary battery. The batteries in either of the form are electro-chemical stores for energy.

Some of the requirements of a battery to store energy are as follows,

1. **Capacity**

   A battery even with small size, light weight and low cost should have high capacity of storing energy. The measure for energy per volume is Joules /Cubic cm.

2. **Capacity Under Load**

   It should have the capability of surviving in different patterns of usage. Because, the consumption of energy levels by a sensor node keeps on varying based on the requirement. And, some times it may consume high amount of energy to perform certain modes of operation.

3. **Self Discharge**

   The ability of self discharging should be low in order to increase the life time.

4. **Efficient Recharging**

   A battery should get recharge efficiently even on low power availability. As a result, it should not show any "memory effect".

5. **Relaxation**

   An empty or unused battery when recharges itself by performing the chemical diffusion within the cell, then the effect obtained is known as relaxation. The lifetime and usable capacity of a battery supporting relaxation effect is significantly extendable.

5.3.2 Design Constraints

Q24. Explain how energy consumption can be reduced in sensor nodes.

Model Paper-II, Q11(a)

Answer :

The components that consumes huge amount of energy in sensor nodes are,

1. Micro controller
2. Radio front ends
3. Memory
4. Sensors.

Hence, to avoid such consumption, the sensor node in the network should be tightly controlled. In order to reduce the power consumption from the above components, a low-power chip that can provide efficient energy for sensor node should be designed.

The major disadvantage of this design is that it can be destroyed when devices are not operated properly.

Since, nodes in wireless sensor network are free most of the time, the other important contribution that can made to reduce power consumption is to turn off the sensor node when they are free and wake them up when,

(i) External stimuli occur

(ii) Depending on time consideration.

The disadvantage of this technique is that complete turning off of node is not always possible. Similarly the other core technique adapted for providing efficient energy consumption is to make use of those state that represents multiple operations simultaneously.

ACPI (Advanced Configuration and Power Interface) is one such technique in which a single state is used for representing fully operational machine where as other four states are used for representing power consumption and wake-up time etc.

Therefore, different operation modes are introduced on various components of sensor nodes. So that different number of sleep states with distinct characteristics act on different model. Thus, for this reason states like idle, active and sleep has been defined for microcontroller. Sleep state of microcontroller keep the sensor nodes in sleep mode. Thereby reducing the power consumptional than required. This energy saving strategy can be explained with the help of the following figure,

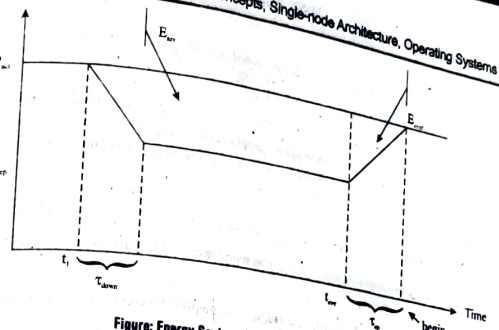

Figure: Energy Saving and Overheads for Sleep Modes

Consider,

1. 't_1' as a decision, which helps in deciding whether a sensor node should be put into sleep mode so as to reduce the power consumed from P_{act} to P_{sleep}.
2. 't_{eve}' as the time by next event to occur if controller is in active state.
3. 't_{act}' = P_{act} $(t_{eve} - t_1)$ as the total energy sent by nodes for being idle.
4. τ_{down} as the total time taken to keep the component in sleep mode.
5. $(P_{act} + P_{sleep})/2$ as average power consumption.

∴ The total energy required to keep node in sleep state is given as $\tau_{down}\,(P_{act} + P_{sleep})/2 + (t_{eve} - t_1 - \tau_{down})\,P_{sleep}$.

The total energy saving is given as,

$$E_{sav} = (t_{eve} - t_1)\,P_{act} - (\tau_{down}\,(P_{act} + P_{sleep})/2 + (t_{eve} - t_1 - \tau_{down})\,P_{sleep})$$

The additional overhead is,

$$E_{over} = \tau_{up}\,(P_{act} + P_{sleep})/2.$$

Q25. Discuss energy consumption in microcontroller.

Answer :

Energy consumption in microcontroller can be clearly understood by following examples,

1. **Intel StrongArm**

The intel strong arm involve three sleep modes. They are as follows,

(a) **Normal Mode**

In this mode, the power consumed is about 400 milliwatt. Each part of processor are completely powered.

(b) **Idle Mode**

In this mode, the power consumed is about 100 milliwatt. Here, CPU clocks are not made to work but the clocks that are relevant to the peripherals are made to work. Upon any interruption, it is returned to normal mode.

(c) **Sleep Mode**

In this mode, the power consumed is about 50 μwatt. Here, only the real-time clock is made to work. A timer interrupt results in occurring a timer wakeup of time period 160 milliseconds.

2. **Texas Instruments MSP430**

The texas instrument MSP430 presents extensive range of operation modes.

(i) A fully operational mode

(ii) Sleep modes.

(i) Fully Operational Mode

It consumes upto 1.2 milliwatt. The total power value provided is 1 MHz with voltage 3 volts.

(ii) Sleep Modes

There are four sleep modes in MSP430 i.e., LMP4, LMP3, ... etc. The LMP4 sleep mode is at the lower level which takes power upto 0.3 μwatt. In this mode, the external interrupt is only responsible for awaking of controller. The LMP3 consumes power upto 6 μwatt. It has a clock that runs always which is used to scheduled wakeups.

3. Atmel ATmega

The Atmel ATmega 128 L comprises of six different modes which consumes the power as follows,

The idle and active modes consumes power in the range of 6 milliwatt to 15 milliwatt and powerdown mode consumes power upto 75 μ watt.

Dynamic voltage scaling is another way of efficient energy consumption wherein lesser amount of voltage is provided at lower clock rates with the assurance of correct operations.

Q26. Discuss classification of memory from the perspective of energy.

Answer :

Memory can be classified into two types based on the energy perspective,

1. On-chip memory of microcontroller.
2. Off-chip RAM i.e., flash memory.

On-chip memory of microcontroller is operated by power which is a part of the power consumption of microcontroller.

OFF-chip RAM such as flash memory is relevant because the construction and utilization of it effects the lifespan of a node. The relevant metrics such as readtimes, write times and energy consumption which can be easily accessed from manufacture data sheets. This data sheet can vary based on various factors.

When these metrics are compared,

(i) Read time and read energy are almost found to be same among various flash memory types.

(ii) Write is quite different because it is based on granularity of accessing data. The difference can be known by counting the number of times the chip is overwritten.

(iii) The energy consumption of erase and write also differ with the ratio of 900:1 among various memory types.

Consider an example of Mica nodes that consumes energy of 1.111 nAh and 83.333 nAh to read and write flash memory respectively. Hence, the task of writing to a flash memory consumes both the time and memory which can be avoided if necessary.

Q27. Explain in detail about radio transceivers along with its different models.

Answer :

Radio transceiver is the device which support both transmission as well as reception of data between the pair of nodes. It can be operated like microcontroller in several modes wherein simpler modes are turning on or turning off. The transceiver should be turned off in order to have low energy consumption and should be turned on only when it is necessary. However this leads to overheads in time and power considerations. The models for the energy consumption per bit for sending as well as receiving specifies the energy consumption behaviour of the transceiver.

The energy consumption models required are,

1. *Modeling energy consumption during transmission.*
2. *Modeling energy consumption during reception.*

1. Modeling Energy Consumption During Transmission

Here, models of various levels make various assumptions as discussed below,

(a) The desired transmission power is known which is the function of system aspects such as, energy per bit over noise (E_b/N_o), efficiency of bandwidth (η_{bw}), distance (d) and path loss coefficient (γ). Here, the amplifier of the transmitter generates the transmitter power. The own power consumption of the transmitter relies on its architecture. But mostly power consumed by them relies on their power generation. The values of these two types of power are proportional to each other.

(b) A particular fixed power level is always needed which is independent of radiated power and a proportional offset. i.e.,

$$P_{amp} = \alpha_{amp} + \beta_{amp} P_{tx}$$

Here,

α_{amp} and β_{amp} are constants.

However, this model states that the maximum output power will give the best amplifier efficiency.

Apart from the amplifier other device such as baseband processor need to be powered up during transmission. The power consumed by other devices is referred to P_{txElec}. However, the energy required for a packet transmission n-bit long relies on the following,

(a) Duration to send the packet which is find out by nominal rate 'R' and the coding rate R_{code}

(b) The total power consumed during transmission.

(c) Startup cost if the transceiver is turned on.

The equation below illustrates the things discussed above,

$$E_{tx}(n, R_{code}, P_{amp}) = T_{start} P_{start} + \frac{n}{R.R_{code}} (P_{txElec} + P_{amp})$$

The above equation do not rely on modulation selected for transmission. This model also assumes that,

(i) Coding overhead only depends on the coding rate and

(ii) Antenna efficiency is perfect.

The Forward Error Correction (FEC) coding can be employed to this model for its improvement.

2. Modeling Energy Consumption During Reception

Here, the energy consumed by a receiver when it is idle is negligible and can be considered as zero.

Consider when the receiver is active, the energy needed to it for receiving a packet is E_{recv}, the startup component for this energy is $T_{start} P_{start}$. This model posses a component which is proportional to packet time $\frac{R}{R.R_{code}}$. This is time which is required for actual reception. However, even during reception the receiver circuitry is required which is referred to as P_{rxElec}.

This energy is utilized for each bit (decoding overhead). Thus, the equation used by this model is given below,

$$E_{rcvd} = T_{start} P_{start} + \frac{n}{R.R_{code}} P_{rxElec} + nE_{decBit}$$

The decoding step utilized in the model is complex. This is because it rely on various system parameters and hardware. The above equation rely on the following factors,

(i) Voltage supply

(ii) Decoding time per bit

(iii) Restricted length (k) of code to be used.

Q28. What is the relation in energy consumption between sending data and computing?

Answer :

The relation in energy consumption between sending data and computing is substantially dependent on the hardware such as microcontroller that is being used. Microcontroller might require 1nJ accounts for single sample in a radio transceiver. And bluetooth transceiver might require nearly 100 nJ for transmitting a single bit.

Apart from microcontroller, the other hardware whose ratio of sending one bit in comparison with computation of single instruction is as given below:

(a) The ratio of communication versus computation lies between 1500 to 2700.

(b) The ratio for MEDUSA II nodes lies between 220 to 2900.

(c) The ratio for WINS NG 2.0 nodes is about 1400.

The RFM TR1000 radio transceiver requires 1 μJ for transmitting a single bit and 0.5 μJ for its reception. The processor of it consumes 8 nJ for a single instruction which results in the ratio for communication to computation cost as 190.

In other context, the same amount of energy is consumed in communicating 1 kB of data across 100 m and computation of three million instructions.

When the details are ignored, it can be noticed that the communication is rather more expensive than computation. But the energy for computation cannot be ignored eventhough it is less than communication which is based on the computational task.

Therefore, this observation inspires network architecture of various methods and design decisions of wireless sensor network. The idea to invest the computation of network at the time of investing on communication that results to the notion of aggregation and in network processing.

5.4 OPERATING SYSTEMS AND EXECUTION ENVIRONMENTS

Q29. Explain in detail about embedded operating systems.

Answer :

Conventionally, an operating system is responsible for controlling and protecting the access of resources and managing the allocation to different users. It allows several processes to communicate with each other and also to be executed concurrently.

In addition to this, resources which support full-blown operating system (OS) are also not made available for such systems whereas in embedded system, the execution code is more restricted and balanced than that of used in a general purpose system. The more simpler term than the operating system is execution environment will be more suitable because requirements of such systems need to be fulfilled by Wireless Sensor Network (WSN). Energy efficient execution needs to be supported by the energy management in the form of Dynamic Voltage Scaling (DVS) techniques.

The external components such as sensors, radio modem or timers and asynchronous information need to be handled in an easier and efficient manner.

In order to do this, the below things are required with less usage of memory and execution time.

(a) Programming paradigms

(b) Structure of protocol stack and

(c) Additional support for energy management.

Q30. Explain the different programming paradigms and their application programming interfaces.

Answer : Model Paper-I, Q11(b)

Programming Paradigms

The different programming paradigms are,

1. Concurrent programming paradigm
2. Process-based programming paradigm
3. Event-based programming paradigms.

1. Concurrent Programming Paradigm

Programming paradigm that support concurrent execution plays a significant role for wireless sensor network because they need to handle the data communication by random sources such as multiple sensors or radio transceiver at arbitrary points.

Consider an example of sequential programming paradigm which shows the need of concurrent programming paradigm. A system should either poll a sensor or radio transceiver in order to know the data availability. If the data is available then it should be processed in a correct way. Afterwards, it must poll the transceiver to know the packet availability. If the packet is available then it should be processed immediately. This can be shown in figure below,

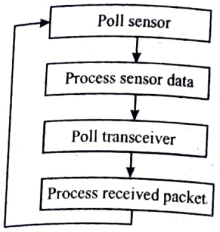

Figure: Sequential Programming Model

This sequential process can have risk of missing data or packet during packet or sensor information is being processed. If the sensor data processing or incoming packet utilize large amount of time then the risk associated with it will be large. Thus, this insufficiency can be overcome by concurrent programming model.

2. Process-based Programming Paradigm

Basically, modern general purpose operating system allows execution of multiple processes concurrently upon a single CPU. So, such process based approach supports the concurrency in a sensor node. This can be shown in figure below,

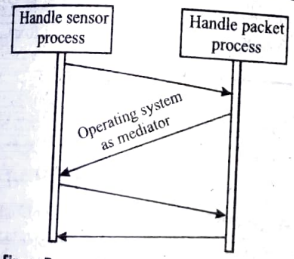

Figure: Process-based Programming Model

Thus, the mapping of an execution model of concurrent processes with a sensor node exhibits granularity mismatch.

The comparison of individual protocol functions with an individual processes causes large overhead of process switching (i.e., switching from one process to another process).

If the small tasks need to be executed more frequently then this problem gets severe in terms of the overhead which is caused by task switching usually done in sensor networks. In addition to this, all the processes need their own stack space in memory that holds it along with the stringent memory constraints of sensor nodes.

3. Event-based Programming Paradigm

The event-based programming paradigm is preferable because it overcomes the problems of process-based programming paradigm. In this, the system is allowed to wait until an event occurs. The event can be data from a sensor node, packet arrival, timer expiration and so on.

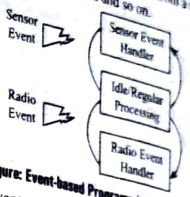

Figure: Event-based Programming Paradigm

An event can be handled by a short sequence of instruction which stores the name of event and its necessary information. The actual information processing of these events should not be performed in event handler routine. Instead it is decoupled from the actual appearance and processed separately.

The event handler can be sensor event handler and radio event handler. These event handlers are responsible for interrupting the normal processing of code. Because the event is simple and short, it can be run in any situation without informing the interruption of code. The event handlers execute sequentially without interrupting each other.

However, this event-based programming paradigm results in differentiating among two different contexts. One of the context is time-critical event handler in which execution can be triggered only by the event handler in order to process normal code.

It provides the following advantages,

1. The event based programming paradigm performance improved by a factor of '8' by comparing it with process based programming paradigm.
2. Its power consumption is minimized by a factor of 12.
3. Its instruction memory requirement is minimized by a factor of 30.

Application Programming Interfaces

Apart from programming models, the interfaces provided to the operating system are used to inquire the internal state of the system. They are also used to set the internal state of the system. This interface must be accessible by both the application programs and the protocol stack and it must be tied with both of them. The Application Programming Interface (API) consists of functional interface, object abstraction and also detailed behavioral semantics. Abstraction indicates wireless links, nodes, etc. And the functions indicate state inquiry and manipulation, accessing hardware and setting of policies.

Q31. Discuss in detail structure of an operating system and protocol stack.

Answer :

The structuring of communication protocols was done traditionally by using layered approach. In this approach, each protocol are stacked on top of each other and each layer in the protocol make use of the functions defined by the layer below it. This layered approach provides the following advantages,

1. Manageable protocol stack
2. Promoting modularity and reuse.
3. Presence of complexity.

Despite of the above advantages, this layered approach is not adequate enough in providing functionality to wireless sensor network. This scenario can be better explained with the following example.

Example

Consider the information that specify the strength of received signal by a communication partner. This information at physical layer can be used for the following,

1. To provide support to networking protocols so that the routing alteration can be decided.
2. To determine the location where the information is stored by evaluating the distance from signal strength.

To provide support to link layer protocols either in channel adaptive or hybrid FEC/ARQ schemes.

Thus, only single information source can be used for benefitting various other protocols which are not directly linked with this source of information. However this type of information exchange is called cross-layer information exchange which is responsible to weaken the restrictions of layered approach.

Moreover, a communication protocol can become flexible manageable and efficient by enabling the following,

1. Supporting wide networking protocol into wireless system.
2. Migrating functionalities into the backbone.
3. Supporting handover mechanisms provided by physical layer in cellular network.

Hence, to avoid such situation, wireless sensor network use component model as communication protocol in order to structure the protocol stack. The advantage of this model is that it breaks the monolithic layers into small self contained components or modules or building block. The component helps in attaining single well defined function depending on which two components interact with each other over an interface.

The component model not only helpful in solving the structing problem associated with protocol stack but also helpful in programming wireless sensor nodes. This is done by embedded component model along with event based approach. In addition to component model, it wraps hardware communication primitives, in network processing functionalities so as to design and implement it as a single unit called component.

5.5 INTRODUCTION TO TinyOS AND nesC

Q32. Explain in detail about TinyOS. Model Paper-II, Q11(b)

Answer :

TinyOS is an open-source operating system designed for wireless embedded sensor networks. The intent of TinyOS is to support applications of sensor networks on hardware platforms, where resources are constrained to achieve applications code minimum size. TinyOS should implement a simple task model and should support minimal device and network abstractions, static memory allocation, language-based development approach and also it does not contain file system which allows to compile only the required parts of the operating system with the application. TinyOS also arranges components in the form of layer as other operating systems. The lower layer is nearer to the hardware and the upper layer is nearer to the application TinyOS provides component architecture that is unique and offers a library as a set of system software components. The system components of the application can be connected to the other components of the application by writing the application code in nesC language.

Consider an example of TinyOS application, Field Monitor. All the nodes in a sensor field send their temperatures and photo sensor readings to a base station using ad hoc routing mechanism. The following figure illustrates Field Monitor application for sensing and sending measurements.

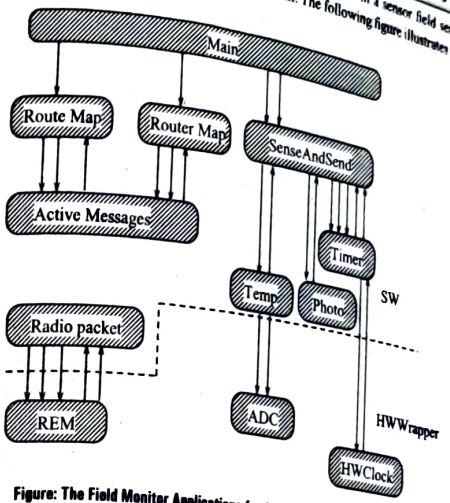

Figure: The Field Monitor Applications for Sensing and Sending Measurement

In the above figure, blocks correspond to TinyOS components and arrows correspond to functional calls. Consider the Timer component to understand the semantics of TinyOS components. Timer component works with a clock, which is a software wrapper around a hardware clock responsible for producing periodic interrupts. The arrowheads indicate the method calls of the Timer component. The method of the component can be called by other component if arrowhead points into the component on the other hand, if arrowhead points outward then it represents that method of the component requires another layer component to provide. The components relationship with other layers are represented by the directions of the arrows.

A program executed in TinyOS contains two components namely tasks and events.

1. Tasks

Tasks are created by components to a task scheduler. The default implementation of the TinyOS scheduler is responsible for maintaining a task queue. The tasks in the queue are called as per the order in which they are created. Tasks are nonpreemptive i.e., tasks are always completed without preemption by other tasks. The task from the task queue will be called by the scheduler only if the prior task has completed. If the tasks are not available in the task queue then scheduler keeps the CPU into sleep mode.

2. Events

The sources of triggered execution are events from hardware like clock, digital inputs and so on. The execution of interrupt handler is referred to as event context. Unlike tasks, the processing of events can be prempted by other events. As events prempt tasks, programmers are required to split the code into small execution sections. This makes events not to block other task for a long period of time.

Q33. Define nesC. Discuss in detail the component interface as well as component implementation in nesC.

Answer :

nesC

nesC is an imperative language which is an extension of C. It is used to support TinyOS through various components.

Component Interface in nesC

A component in nesC contains both interface specification as well as component implementation. The interfaces of a nesC component are divided into provides or uses interfaces inorder to reflect the layered structure of TinyOS.

Provides Interface

It is a collection of method calls exposed to the upper layer.

Uses Interface

It is a collection of method calls hiding the lower layers. For instance, the interface specification of the Timer component is shown below. The interface which in turn independent of the Timer component is referred to as TimerModule.

```
module TimerModule
{
    provides
        {
            interface StdCtrl;
            interface Timer;
        }
    uses interface Clock as Clk;
    }
interface StdCtrl
{
    command result init( );
}
interface Timer
{
    command result start(char type, uint32 interval,);
    command result stop( );
    event result fire( );
}
interface Clock
{
    command result setRate(char interval, char scale);
    event result fire( );
}
```

Method call semantics of the interfaces are same but nesc differentiates the directions of the interface calls between layers as event calls and command calls.

Event Call

A method call from lower layer component to higher layer component is referred to as event call.

Command Call

A method call from upper-layer component to lower layer component is referred to as command call. User must know the type of the interface and also the direction of the method call to check whether an interface method is implemented by the component or is required by the component.

A component can use or provide the same interface multiple number of times by providing each interface a separate name through 'as' notation.

Component Implementation in nesC

nesC components can be classified into two types based on their implementation. The two components are,

1. Modules
2. Configurations.

1. **Modules**

Application code is responsible for implementing modules. The module implementation section-U is written in C-like code. A command or an event bar in an interface foo is represented by foo.bar. The invocation of a command is represented by a keyword *call* and triggering of an event is represented by a keyword *signal*. The following code illustrates the implementation definition the Timer component.

```
module Timer
{
    provides
    {
        interface StdControl;
        interface Timer01;
    }
    uses interface Clock as Clk;
}
implementation
{
    bool evenFlag;
    command result_t StdControl.init( )
    {
        evenFlag=0;
        return call Clk_setRate(128,4);
    }
    event result_t Clk.fire( )
{
evenFlag=!evenFlag;
if(evenFlag)
{
    signal Timer01.timer0Fire( );
    }
        else
        {
            signal Timer01.timer1Fire( );
        }
        return SUCCESS;
    }
}
```

Configurations

Connecting interfaces of existing components are responsible for implementing configurations for instance, a timer service (called TimerC) can be obtained by connecting Timer component and a hardware clock Wrapper (HWClock).

The following figure illustrates how the components like Timer and HWClock are connected,

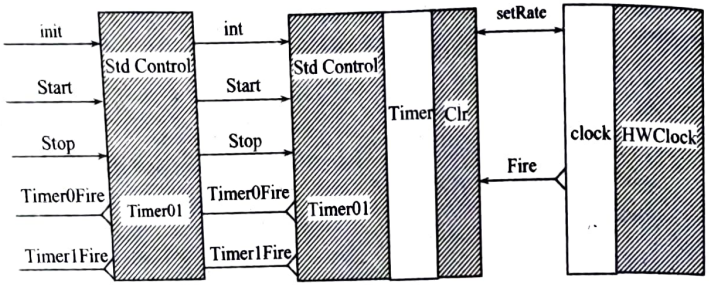

Figure: TimerC Configuration Implemented by Connecting Timer with HWClock

The keyword configuration in the specification represents that component cannot be directly implemented as a module. In configuration implementation, the code first includes the two components and then specifies that interface stdcontrol of the TimerC component is the StdControl interface of the Timer module. The operator is used to specify the connection between the clock interfaces.

An application refers to a configuration instead of a module. A main module must be involved in an application which is responsible for linking the code to the scheduler at run time.

Q34. Explain how concurrency is supported in nesC.

Answer :

The execution model of TinyOS directly reflected in the imperative language. nesC using command and event contexts. The following section illustrates the component SenseAndSend to not only support concurrency in nesC but also to limit race conditions.

The construction of SenseAndSend component is made on top of the Timer component, an ADC component, which is responsible for providing sensor readings and a communication component, which is responsible for sending a packet.

```
module SenseAndSend
{
    provides interface stdControl;
    uses interface ADC;
    uses interface Timer;
    uses interface send;
}
implementation
{
    bool busy;
    norace uint16_t sensorReadings;
    command result_t stdControl.init( )
    {
        busy=FALSE;
    }
```



```
    event result_t Timer.timerOFire( )
    {
        bool localBusy;
        atomic
        {
            localBusy=busy;
            busy=TRUE;
        }
        if{!localBusy}
        {
            call ADC.getData( );
            return SUCCESS;
        }
        else
        {
            return FAILED;
        }
    }
    task void SendData( )
    {
        adcPacket_data=SensorReading;
        callSend.Send{&adcPacket,sizeof adcpacket data};
        return SUCCESS;
    }
    event result_t ADC.dataReady(uint16_t data)
    {
        sensorReading=data;
        post SendData( );
        atomic
        {
            busy=FALSE;
        }
        return SUCCESS;
    }
}
```

The SenseAndSend components responds to TimerOFire event by calling the ADC to poll sensor readings. But, getting a sensor readings takes a long time. So, a split-phase operation is implemented for getting sensor readings.

ADC.getData( ) returns immediately by calling it and a signal by ADC.dataReady( ) event indicates the completion of the operation. When ADC is busy in handling requests than all the new requests are rejected explicitly by using a busy flag. The ADC.getData( ) method sets the busy flag to true and the ADCdataReady( ) method sets the busy flag to false.

In the SenseAndSend, the race condition is the updating of the busy flag both scheduled tasks and event-triggered interrupts. To avoid race conditions several language facilities are provided by nesC. In nesC, code can be classified into two types,

1. **Synchronous Code (SC):** It is the code that is reachable from tasks.
2. **Asynchronous Code (AC):** It is the code that is reachable from atleast one interrupt handler.

IMPORTANT QUESTIONS

SHORT QUESTIONS

Q1. What are the hardware components for a wireless sensor nodes?

Ans: For answer refer Unit-V, Q2.

Important Qu

Q2. Explain in details about transceivers.

Ans: For answer refer Unit-V, Q3.

Important Que

Q3. What are the communication devices used for exchanging data between nodes?

Ans: For answer refer Unit-V, Q5.

Important Que

Q4. Write about application programming interfaces.

Ans: For answer refer Unit-V, Q7.

Important Que

ESSAY QUESTIONS

Q5. Explain in brief about design principles of wireless sensor networks.

Ans: For answer refer Unit-V, Q9.

Important Ques

Q6. Define gateway. Explain in brief about the need for gateways.

Ans: For answer refer Unit-V, Q11.

Important Ques

Q7. Explain in brief about WSN to Internet communication.

Ans: For answer refer Unit-V, Q13.

Important Quest

Q8. Discuss in brief about how communication is done from internet to WSN communication.

Ans: For answer refer Unit-V, Q14.

Important Quest

Q9. Explain in detail about the hardware components for a wireless sensor node.

Ans: For answer refer Unit-V, Q15.

Important Questi

Q10. Explain how energy consumption can be reduced in sensor nodes?

Ans: For answer refer Unit-V, Q24.

Important Questi

Q11. Explain in detail about embedded operating systems.

Ans: For answer refer Unit-V, Q29.

Important Questio

Q12. Explain in detail about TinyOS.

Ans: For answer refer Unit-V, Q32.

Important Questio

Q13. Define nesC. Discuss in detail the component interface as well as component implementation in nesC

Ans: For answer refer Unit-V, Q33.

Important Questio