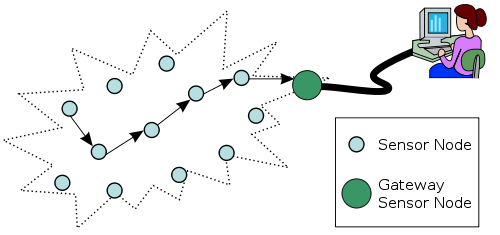
***EXPERIMENT NO:09***

***Aim:*** *Case study of WSN.*

***Theory:***

## → Wireless Sensor Network

**Wireless Sensor Networks** (**WSN**), sometimes called **Wireless Sensor and Actuator Networks** (**WSAN**), are spatially distributed autonomous sensors to *monitor* physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling *control* of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.



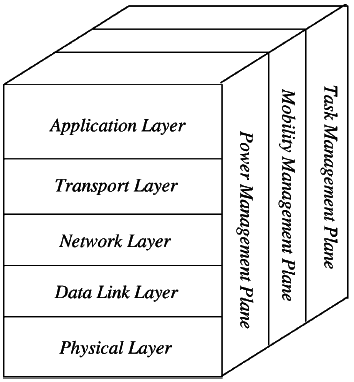
## → Characteristics

WSN is currently used for real-world unattended physical environment to measure numerous parameters. So, the characteristics of WSN must be considered for efficient deployment of the network. The significant characteristics of WSN are described as follows:

* Low cost: in the WSN normally hundreds or thousands of sensor nodes are deployed to measure any physical environment. In order to reduce the overall cost of the whole network the cost of the sensor node must be kept as low as possible.
* Energy efficient: energy in WSN is used for different purpose such as computation, communication and storage. Sensor node consumes more energy compared to any other for communication. If they run out of the power they often become invalid as we do not have any option to recharge. So, the protocols and algorithm development should consider the power consumption in the design phase.
* Computational power: normally the node has limited computational capabilities as the cost and energy need to be considered.
* Communication Capabilities: WSN typically communicate using radio waves over a wireless channel. It has the property of communicating in short range, with narrow and dynamic bandwidth. The communication channel can be either bidirectional or unidirectional. With the unattended and hostile operational environment it is difficult to run WSN smoothly. So, the hardware and software for communication must have to consider the robustness, security and resiliency.
* Security and Privacy: Each sensor node should have sufficient security mechanisms in order to prevent unauthorized access, attacks, and unintentional damage of the information inside of the sensor node. Furthermore, additional privacy mechanisms must also be included.
* Distributed sensing and processing: the large number of sensor node is distributed uniformly or randomly. WSNs each node is capable of collecting, sorting, processing, aggregating and sending the data to the sink. Therefore the distributed sensing provides the robustness of the system.
* Dynamic network topology: in general WSN are dynamic network. The sensor node can fail for battery exhaustion or other circumstances, communication channel can be disrupted as well as the additional sensor node may be added to the network that result the frequent changes in the network topology. Thus, the WSN nodes have to be embedded with the function of reconfiguration, self-adjustment.
* Self-organization: the sensor nodes in the network must have the capability of organizing themselves as the sensor nodes are deployed in an unknown fashion in an unattended and hostile environment. The sensor nodes have work in collaboration to adjust themselves to the distributed algorithm and form the network automatically.
* Multi-hop communication: a large number of sensor nodes are deployed in WSN. So, the feasible way to communicate with the sinker or base station is to take the help of an intermediate node through routing path. If one need to communicate with the other node or base station which is beyond its radio frequency it must me through the multi-hop route by intermediate node.
* Application oriented: WSN is different from the conventional network due to its nature. It is highly dependent on the application ranges from military, environmental as well as health sector. The nodes are deployed randomly and spanned depending on the type of use.
* Robust Operations: Since the sensors are going to be deployed over a large and sometimes hostile environment. So, the sensor nodes have to be fault and error tolerant. Therefore, sensor nodes need the ability to self-test, self-calibration, and self-repair.
* Small physical size: sensor nodes are generally small in size with the restricted range. Due to its size its energy is limited which makes the communication capability low.

## → Wireless Sensor Network Architecture

The most common WSN architecture follows the OSI architecture Model. The architecture of the WSN includes five layers and three cross layers. Mostly in sensor n/w we require five layers, namely application, transport, n/w, data link & physical layer. The three cross planes are namely power management, mobility management, and task management. These layers of the WSN are used to accomplish the n/w and make the sensors work together in order to raise the complete efficiency of the network.



*Wireless Sensor Network Architecture*

## → Application Layer

The application layer is liable for traffic management and offers software for numerous applications that convert the data in a clear form to find positive information. Sensor networks arranged in numerous applications in different fields such as agricultural, military, environment, medical, etc.

## → Transport Layer

The function of the transport layer is to deliver congestion avoidance and reliability where a lot of protocols intended to offer this function are either practical on the upstream. These protocols use dissimilar mechanisms for loss recognition and loss recovery. The transport layer is exactly needed when a system is planned to contact other networks.

Providing a reliable loss recovery is more energy efficient and that is one of the main reasons why TCP is not fit for WSN. In general, Transport layers can be separated into Packet driven, Event driven. There are some popular protocols in the transport layer namely STCP (Sensor Transmission Control Protocol), PORT (Price-Oriented Reliable Transport Protocol and PSFQ (pump slowly fetch quickly).

## → Network Layer

The main function of the network layer is routing, it has a lot of tasks based on the application, but actually, the main tasks are in the power conserving, partial memory, buffers, and sensor don’t have a universal ID and have to be self-organized.

The simple idea of the routing protocol is to explain a reliable lane and redundant lanes, according to a convinced scale called metric, which varies from protocol to protocol. There are a lot of existing protocols for this network layer, they can be separated into; flat routing and hierarchical routing or can be separated into time driven, query-driven & event driven.

## → Data Link Layer

The data link layer is liable for multiplexing data frame detection, data streams, MAC, & error control, confirm the reliability of point–point (or) point– multipoint.

## → Physical Layer

The physical layer provides an edge for transferring a stream of bits above physical medium. This layer is responsible for the selection of frequency, generation of a carrier frequency, signal detection, Modulation & data encryption. IEEE 802.15.4 is suggested as typical for low rate particular areas & wireless sensor network with low cost, power consumption, density, the range of communication to improve the battery life. CSMA/CA is used to support star & peer to peer topology. There are several versions of IEEE 802.15.4.V.

# Os in wsn

## TinyOS

TinyOS is an open source, flexible, component based, and application-specific operating system designed for sensor networks. TinyOS can support concurrent programs with very low memory requirements. The OS has a footprint that fits in 400 bytes. The TinyOS component library includes network protocols, distributed services, sensor drivers, and data acquisition tools. The following subsections survey the TinyOS design in more detail.

### ***Features:***

* **File System:** TinyOS provides a single level file system. The rationale behind providing a single level file system is the assumption that only a single application runs on the node at any given point in time. As node memory is scarce, having a single level file system is therefore sufficient.
* **Database Support:** The purpose of sensor nodes is to sense, perform computations, store and transmit data, therefore TinyOS provides database support in the form of TinyDB. Further details on TinyDB can be found in.
* **Security Support:** Communication security in wireless broadcast medium is always required. TinyOS provides its communication security solution in the form of TinySec.
* **Simulation Support:** TinyOS provides simulation support in the form of TOSSIM. The simulation code is written in NesC and consequently can also be deployed to actual motes.
* **Language Support:** TinyOS supports application development in the NesC programming language. NesC is a dialect of the C language.
* **Supported Platforms:** TinyOS supports the following sensing platforms: Mica, Mica2, Micaz, Telos, Tmote and a few others.

## Contiki

Contiki, is a lightweight open source OS written in C for WSN sensor nodes. Contiki is a highly portable OS and it is build around an event-driven kernel. Contiki provides preemptive multitasking that can be used at the individual process level. A typical Contiki configuration consumes 2 kilobytes of RAM and 40 kilobytes of ROM. A full Contiki installation includes features like: multitasking kernel, preemptive multithreading, proto-threads, TCP/IP networking, IPv6, a Graphical User Interface, a web browser, a personal web server, a simple telnet client, a screensaver, and virtual network computing.

### ***Features:***

In this section, we briefly discuss additional features provided by the Contiki OS.

* **Coffee File System:** Contiki provides file system support for flash-based sensor devices in the form of the Coffee file system. The purpose of the Coffee file system is to provide a programming interface for building efficient and portable storage abstractions. Coffee provides a platform independent storage abstraction through an expressive programming interface. Coffee uses a small and constant RAM footprint per file, making it scalable.
* **Security Support:** Contiki does not provide support for secure communication. A proposal and implementation of a secure communication protocol with the name Contiki Sec has been provided in.
* **Simulation Support:** Contiki provides sensor network simulations through Cooja.
* **Language Support:** Contiki supports application development in the C language.
* **Supported Platforms:** Contiki supports the following sensing platforms: Tmote, AVR series MCU.

## Mantis

The MultimodAl system for NeTworks of In-situ wireless Sensors (MANTIS) provides a new multithreaded operating system for WSNs. MANTIS is a lightweight and energy efficient operating system. It has a footprint of 500 bytes, which includes kernel, scheduler, and network stack. The MANTIS Operating System (MOS) key feature is that it is portable across multiple platforms, *i.e*., we can test MOS applications on a PDA or a PC. Afterwards, the application can be ported to the sensor node. MOS also supports remote management of sensor nodes through dynamic programming. MOS is written in C and it supports application development in C. The following subsections discuss the design features of MOS in more detail.

### 

### ***Features:***

In this section we discuss the additional features provided by the MANTIS OS.

* **Simulation Support:** MANTIS supports wireless sensor network simulation through AVRORA.
* **Language Support:** MANTIS supports application development in the C language.
* **Supported Platforms:** MANTIS supports the following sensing platforms: Mica2, MicaZ , and Telos .
* **Shell:** An implementation of a Unix-like shell comes with MANTIS that runs on the sensor node.

## Nano-RK

Nano-RK is a fixed, preemptive multitasking real-time OS for WSNs. The design goals for Nano-RK are multitasking, support for multi-hop networking, support for priority-based scheduling, timeliness and schedulability, extended WSN lifetime, application resource usage limits, and small footprint. Nano-RK uses 2 Kb of RAM and 18 Kb of ROM. Nano-RK provides support for CPU, sensors, and network bandwidth reservations. Nano-RK supports hard and soft real-time applications by the means of different real-time scheduling algorithms, e.g., rate monotonic scheduling and rate harmonized scheduling. Nano-RK provides networking support through socket-like abstraction. Nano-RK supports FireFly and MicaZ sensing platforms.

### ***Features:***

In this section we discuss additional features provided by the Nano-RK OS.

* Language Support.
* Nano-RK supports application development in the C language.
* Supported Platforms.
* Nano-RK supports the following sensing platforms: MicaZ, and FireFly.

## LiteOS

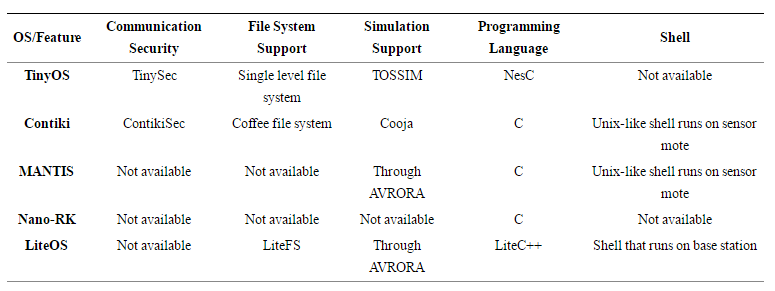
LiteOS is a Unix-like operating system designed for WSNs at the University of Illinois at Urbana-Champaign. The motivations behind the design of a new OS for WSN are to provide a Unix-like OS for WSN, provide system programmers with a familiar programming paradigm (thread-based programming mode, although it provides support to register event handlers using callbacks), a hierarchical file system, support for object-oriented programming in the form of LiteC++, and a Unix-like shell. The footprint of LiteOS is small enough to run on MicaZ nodes having an 8 MHz CPU, 128 bytes of program flash, and 4 Kbytes of RAM. LiteOS is primarily composed of three components: LiteShell, LiteFS, and the Kernel. In the following subsections we discuss these three components and other features of LiteOS in detail.

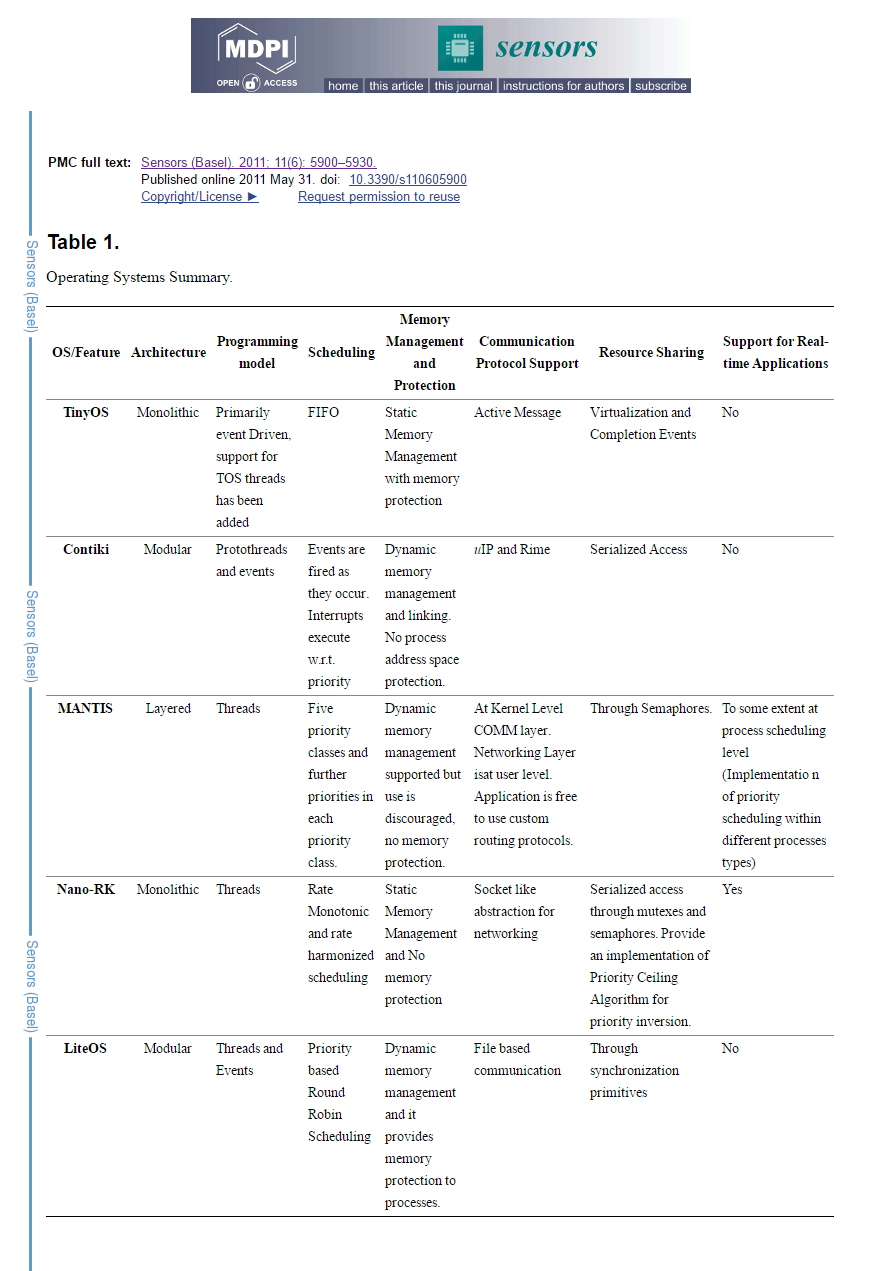
### 

### ***Features:***

In the section, we discuss additional features provided by the LiteOS.

* Lite File System (LiteFS).
* LiteOS provides support for a hierarchical file system called LiteFS. LiteFS provides support for both files and directories. LiteFS is partitioned in three modules. It keeps open file descriptors, memory allocation bit vectors, and information about flash memory layout in RAM. A second module resides in EEPROM and this module contains information about the hierarchical directory structure. Thirdly, it uses flash to store data. As in the Unix file system, files in LiteOS represent different entities, such as data, application binaries, and device drivers.
* The latest version of LiteFS supports eight file handlers in RAM and each handler consumes eight bytes. Therefore, at most eight files can be opened simultaneously. Two bit vectors are used to keep track of the current allocations in EEPROM and flash. The bit vector corresponding to the EEPROM consumes eight bytes and the bit vector corresponding to the data flash consumes 32 bytes. In total LiteFS requires 104 bytes inside the RAM.
* LiteFS mounts all single hop nodes to the file system just like mounting a USB device. It is important to note that since all single hop neighbors are mounted on a PC running some version of Linux OS, a user with access to the PC can copy or delete files on these nodes. The user can copy a new binary executable from the PC to a particular node and then it can issue an exec command to instruct the node to start executing the file.
* Simulation Support.
* LiteOS supports wireless sensor networks simulations through AVRORA.
* Language Support.
* LiteOS supports application development in the LiteC++ language.
* Supported Platforms.
* LiteOS supports the following sensing platforms: MicaZ and AVR series MCU.



****

***Conclusion*:**

Wireless sensor networks become more popular these days due to its less power requirement, low cost, performance and high potential application areas. In this paper we have presented wireless sensor network architectures, their applications and various design issues. The application of WSN in the areas of military, vehicle parking, event detection, greenhouse surveillance, environmental, home applications have been briefed. We have analyzed and enlisted analyzed various design issues that are faced in the design of WSN monitoring strategy. In order to use wireless technology in different applications like we have presented in the paper, needs a well understanding of the network architecture. The application is not restricted to the areas elaborate in this paper. The future prospects of WSN applications are highly promising to revolutionize our everyday lives.