PROJECT REPORT ON

“BLUETOOTH BASED SENSOR NETWORK”

UNDER THE GUIDANCE OF

## Dr. ANAND KUMAR SOURABH



CSE (IOT) ENGINEERING DEPARTMENT

**A.C. PATIL COLLEGE OF ENGINEERING**

**2022-23**

SIGNATURE:

DATE:

REMARK:

PROJECT REPORT

ON

“BLUETOOTH BASED SENSOR NETWORK”

Submitted to:

## Dr. ANAND KUMAR SOURABH

**Project By:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SR NO | Name | PRN NO | Roll number | Class | SIGN |
| 1. | PRIYUSH KHOBRAGADE | 211112018 | 52 | T. E. |  |
| 2. | AYUSHI MASKE | 201111002 | 15 | T. E. |  |
| 3. | KAUSTUBH DHAKATE | 201111004 | 07 | T. E. |  |
| 4. | AKSHAY LOHAR | 211112032 | 54 | T. E. |  |

**BONAFIDE CERTIFICATE**

Certified that this report **“BLUETOOTH BASED SONSOR NETWORK”** is the bonafide work of “**PRIYUSH KHOBRAGADE, AKSHAY LOHAR, AYUSHI MASKE And KAUSTUBH DHAKATE”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**HEAD OF THE DEPARTMENT SUPERVISOR**

Professor, Assistant Professor,

Dept. of Computer Science And Engineering, Dept. of Computer Science and Engineering

AC Patil Engineering College, (IOT) ,

New -Mumbai , Kharghar 410210New Mumbai , Kharghar 410210

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**DECLARATION**

We affirm that the title “BLUETOOTH BASED SONSOR NETWORK”. Begin submitted in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering is the original work carried out by me. It has not forwarded the part of any other thesis submitted for award of any degree or diploma, either in this or any other University.

**Signature of the candidate: -**

|  |
| --- |
| PRIYUSH KHOBRAGADE 52 |
| AKSHAY LOHAR 54 |
| KAUSTUBH 07 |
| AYUSHI MASKE 15 |

I certify that the declaration made above by the candidate is true.

Signature of the Guide,

**(Dr. Anand Kumar Sourabh**)

CSE(IOT), Assistant Professor,

Dept. of Computer Science and Engineering,

AC PATIL ENGINEERING COLLAGE

KHARGHAR – 410210.

**ACKNOWLEDGEMENT**

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

We greatly and profoundly thankful to our honorable GUIDE Prof. S. P. Anand Kumar for giving motivation and providing all necessary facilities for the successful completion of this project.

We greatly and profoundly thankful to our honorable HOD Prof. S. P. Pawar for giving motivation and providing all necessary facilities for the successful completion of this project.

We also record my sincere thanks to beloved PRINCIPAL **Dr.V.N. Pawar**, for their Moral support and deeds in bringing out this project successfully.

We also thankful to all the faculty members of **Department of Computer Science and Engineering (IOT)** for their kind and valuable cooperation during the course of the project.

I would like to express my special gratitude and thanks to faculties for giving me such attention and time. My thanks and appreciations also go my colleague in developing the project and people who have willingly helped me out with their abilities.

TABLE OF CONTENT

CHAPTER 1

* 1. [BLUETOOTH INTRODUCTION](#_TOC_250031)
  2. BLUETOOTH SPECIFICATION
  3. [TECHNOLOGY OVERVIEW](#_TOC_250030)

[CHAPTER 2](#_TOC_250023)

* 1. BLUETOOTH TOPOLOGY
  2. A PICONET
  3. [BTNODE](#_TOC_250020)
  4. DESIGN
  5. SCATTERNET
  6. BLUETOOTH BASED SENSOR NETWORK
  7. FEATURE OF BLUETOOTH

CHAPTER 3

* 1. [BLUETOOTH HARDWARE ARCHITECTURE](#_TOC_250017)

3.2 DESIGN

CHAPTER 4

* 1. [BLUETOOTH HARDWARE ARCHITECTURE](#_TOC_250017)

**CHAPTER 4 BLUETOOTH WORKING**

* 1. IN NETWORK QUERY PROCESSING
  2. [TOPOLOGY MANAGEMENT](#_TOC_250005)

4.3 [TDM AT THE REDIO LEVEL](#_TOC_250004)

**CHAPTER 5 SEPERATED CHANNEL**

**5.1 CALIBRATION OF BLNODES**

**5.2 CODE FOOTPRINT**

**5.3THROUGHPUT**

**5.4 THROUGHPUT FOR MULTIPOINT**

**CHAPTER 6 ENERGY CONSUMPTION**

**6.1 THE POWER SLEEPMODE**

**6.2 NETWORK SELF ASSEMBLY**

**6.3 BLUETOOTH BASED SENSOR NETWORK**

**6.4 WSN**

**CHAPTER 7 BLUETOOTH IPLEMENTATION**

**7.1 IMPLEMENTATION**

**7.2 WIRELESS CONNECTIVITY OVER BLUETOOTH**

**7.3 NETWORK ARRENGEMENT**

**CHAPTER 8 IMPLEMETATION**

**8.1 BLUETOOTH ARCHETECTURE**

**8.2 BLUETOOTH MODULE HARDWARE ARCHETECTURE**

**8.3 DISCOVERY OF SMART NODES**

**CHAPTER 9 ADVANTAGES**

**9.1 ADVANTAGES**

**9.2 LIMITATION**

**9.3 ACKNOLDGEMENT**

**9.4 APPLICATION**

**CHAPTER 10 CONCLUSION**

10.1 CONCLUSION

**CHAPTER 11 REFERENCES**

11.1 REFERENCES

**BLUETOOTH BASED SMART SENSOR NETWORKS**

**Abstract:**

Wireless sensor networks are networks of small computers, fitted with sensors, microprocessors and wireless interfaces. This technology has achieved a lot of attention lately. For these networks are suggested the wide range of modern and fascinating applications, from personal health care to environmental monitoring and military applications. Different wireless technologies, such as simple RF, Bluetooth, UWB or infrared, may be used to communicate between sensors. The core concepts, features and problems of Bluetooth-based wireless sensor networks are outlined in this, as well as the execution of a simple Bluetooth-based sensor network. The presentation of main problems experienced during the implementation and applied solutions are also done. This is specified on the smart sensor networks using the Bluetooth topology. How smart sensor networks are used and can be implemented using the Bluetooth technology. How they are used as the purpose of communication in industrial field, how they are built, their working and concept is reviewed. Architecture, network, applications and working has been reviewed basically for the communication and research purpose.

Wireless sensor networks are networks of small computers, fitted with sensors, microprocessors, and wireless interfaces. This technology has achieved a lot of attention lately. These networks are suggested for the wide range of modern and fascinating applications, from personal health care to environmental monitoring and military applications. Different wireless technologies, such as simple RF, Bluetooth, UWB or infrared, may be used to communicate between sensors. The core concepts, features, and problems of Bluetooth-based wireless sensor networks are outlined in this, as well as the execution of a simple Bluetooth-based sensor network. The presentation of the main problems experienced during the implementation and applied solutions is also done. This is specified on the smart sensor networks using the Bluetooth topology

**Keywords:**  Bluetooth Technology, Wireless Sensor Node & Network Architecture, Wireless Sensor Node design, WSN formation.

# CHAPTER 1

# 1.1 INTRODUCTION

Wireless sensor networks are networks of small devices having sensors, microprocessors and wireless communication interfaces. This technology has become famous lately. For the purpose of communication in industrial field, WSN technology is widely used. Sensors are used for the communication in industries. In this process, the signals are sent through the wires from each field devices and are monitored on central control room. With the beginning of wiring concept, the field device is used for minimizing the wiring cost. Wireless technology is introduced to eliminate the wires as they are costly, bulky and can be easily damaged.

Ericsson Mobile communications started research to explore the usefulness of a consuming low power, low cost, low ratio interface, and to find a process to remove wires between the devices in the year 1994. Then the Bluetooth technology was invented by an electrical engineer Dr. Jaap Haartsen and named the technology Bluetooth to the honor of the 10th century king Harald “Blue tooth” of Denmark. The aim of Bluetooth is harmony and unification [6]. It also enables the different devices to communicate through wireless connectivity. Bluetooth uses frequency hopping spread spectrum technique and works in the illicit ISM band at 2.4 GHz frequency. A distinctive quality Bluetooth device holds a range of about 10 meters and can be extended to 100 meters. The total bandwidth of 1 Mb / sec is supported by communication channel. A topmost data transfer rate of 721 KBPS maximum of three channels is supported through a single channel.

* Bluetooth is wireless high speed data transfer technology over a short range (10 − 100 meters).
* Bluetooth Wireless Technology (BWT) was developed in 1994 at Ericsson in Sweden.
* Purpose − Originally it was build to eliminate the need for cable connections between PDAs and notebook PCs. Later the goals were to enable different devices through a commonly accepted standard for wireless connectivity.
* Ericsson on advent of BWT conceptualized a Radio Technology through a wireless personal area network (WPAN)*.* Group called Bluetooth Special Interest Group (SIG) was formed in 1998 to develop the standard of IEEE 802.1 . This specification standardized the Bluetooth technology world wide.

# 1.2 BLUETOOTH SPECIFICATIONS: -

1. Developed by: Jaap Haartsen and Sven Mattisson in Sweden
2. Standard: IEEE 802.15
3. ISM Band Frequency: 2.4 GHz
4. Range: 10 − 100 meters
5. Channel Bandwidth: 1 Mbps
6. Maximum Asymmetric Data Transfer Rate: 721 Kbps

**Why the name BLUETOOTH** ?

The name was adopted as a tribute to the tenth−century Viking king Harald Blåtand who peacefully united Denmark and Norway. Harald liked to eat blueberries, which gave his teeth the coloration that lead to the nickname "Bluetooth."

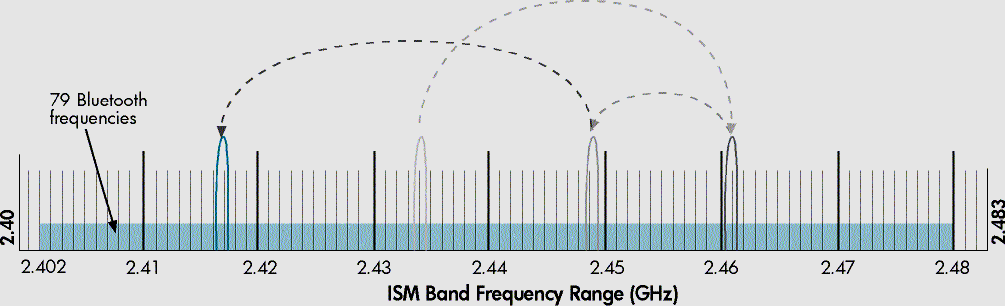
# TYPE’S OF BWT:

Depending on the *power consumption* and *range* of the device, there are 3 Bluetooth Classes as:

* **Class 1**: Max Power − 100mW ; Range − 100 m
* **Class 2**: Max Power − 2.5mW ; Range − 10 m
* **Class 3**: Max Power − 1mW; Range − 1 m

# 1.3 BLUETOOTH OPERATIONS :

* BWT− enabled devices operate in 2.4GHz ISM Band (Industrial, Science, Medical band).
* It uses 79 1−MHz frequencies in the ISM Band.
* Technique used − *frequency hopping,* to minimize interference from other networks that also use ISM Band.



**Figure No. 1** BWT−enabled devices hop between frequencies up to 1600 times per second

# CHAPTER 2

# 2.1 BLUETOOTH TOPOLOGY

Depending on the type of connections established between Bluetooth devices, 2 main topologies are as:

1. PICONET TOPOLOGY, and
2. SCATTERNET TOPOLOGY

**2.2 A Piconet:**

A *piconet* consists of upto 8 BWT−enabled devices.

When *piconet* is established, one device sets up *frequency− hopping pattern* and other devices synchronize their signals to the same pattern.

Primary Devices: Those devices which sets the frequency−

hopping pattern.

Secondary Devices: Those devices which get synchronized. Each *piconet* has a different frequency−hopping pattern. In Bluetooth, each *piconet* has 1 Master for establishment of

*piconet*, and up to 7 Slave devices. Master’s Bluetooth address is used for defining frequency− hopping sequence. Slave devices use master clock to synchronize their clocks so as to hop simultaneously. For establishing *piconet*, other Bluetooth devices in range are discovered by an inquiry procedure.

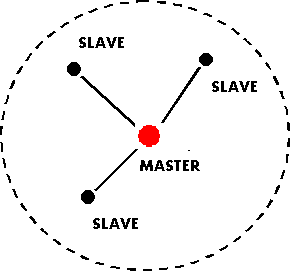


FIGURE 2.1

It is an improvised network used to link wireless devices using Bluetooth technology. It is group of up to 8 devices that shares similar frequencies. It uses the concept of master and slave. Each piconet has one master and rest of the device acts as slave. Usually, the device that starts off the piconet behaves as master. To establish a piconet, first the device searches for the other Bluetooth devices in the range. When the two devices have same frequency then required information is exchanged and to establish the connection paging procedure can be used. For more than 7 devices need to exchange information, there are two possibilities:

The initial one is by putting one or more devices into park state. The modes used in Bluetooth are sniff, hold and park which are used for low power consumptions. When a device changes to the park mode then it disconnects from Piconet, but schedule adjustment will be maintained. The master of piconet continuously transmits signals to invite the slaves to retain the piconet. If there are less than 7 slave devices in the piconet then only the slave will rejoin. If not then one of the active slave devices will be park by the master. Due to these actions, there will be delay and it can be undesirable for some applications like procedure control applications that need instant reply from the command center.

# 2.3 BTNODES

The BTnodes were developed by ETH Zurich in the con- text of the Smart-Its project. They are based on the Atmel ATmega128L microcontroller - an 8bit microcontroller (MCU) clocked at 7.4 MHz, with 4 KiB1 on chip memory and an external memory chip of up to 64 KiB. The MCU has digital and analog I/O ports that can be used to connect external sensor devices through Molex plugs on the edge of the board. The nodes are equipped with a Bluetooth module (Ericsson ROK 101 007) together with an onboard antenna. Two UARTs connect the MCU with the embedded Bluetooth chip and one of the Molex plugs. Four leds can be used for debugging purposes. The board also contains a voltage regulator: the BTnode can be plugged to power supplies ranging from 3.3 V to 12 V.

|  |  |
| --- | --- |
| MCU | Atmel ATmega128L at 7.372 MHz |
| Memory | Built in: 128 KiB Flash, 4 KiB SRAM  4 KiB EEPROM  External: 64 KiB RAM |
| I/O | 8 Channel 10-bit A/D-converter  2 programmable serial UARTS |
| Embedded  Radio | Ericsson ROK 101 007 |
| External  Radio | BTTester  (Ericsson ROK 101 007) |

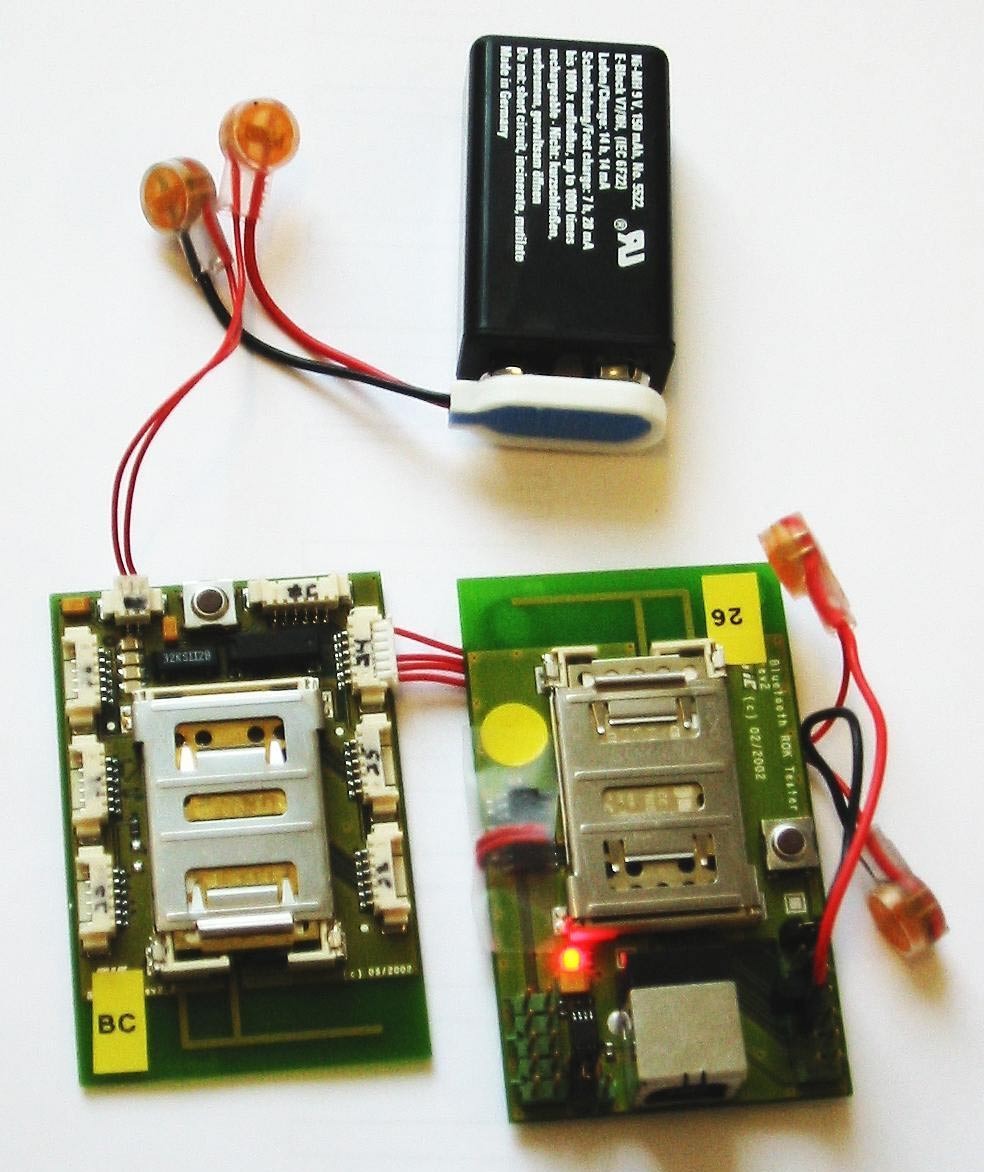
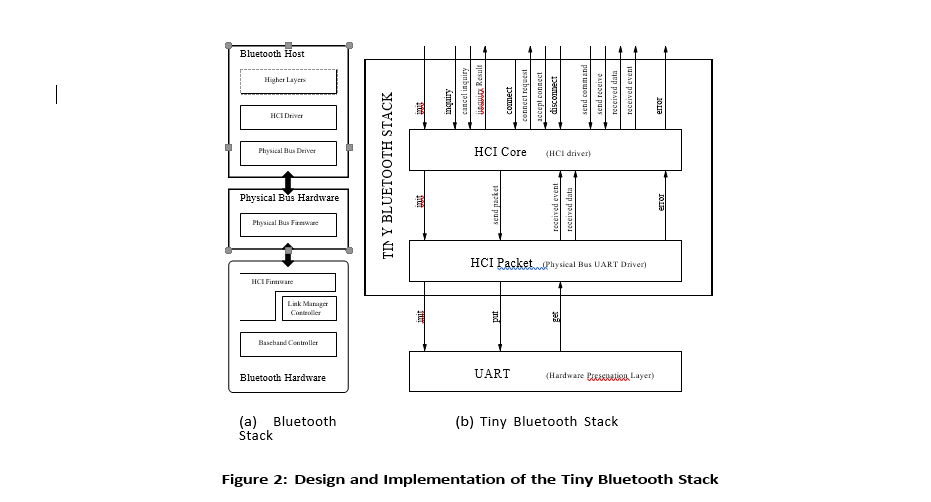


Figure 2.2 : BTnodes characteristics

As explained in the Introduction, we used dual-radio nodes for our experiments in order to assemble a multihop network.1EIC standard 60027-2 deﬁnes KiB as 1024 bytes.



**FIGURE 2.3**

We thus connected an extra Bluetooth module to the BT- node (via the Molex plug connected to an UART). For this purpose, we used the BT tester (also from ETH Zurich),a serial dongle based on the ROK 101 007 Ericsson chip (in- clouding an onboard antenna and a monitoring led).

Figure 1 summarizes the characteristics of the BTnodes. Further documentation can be obtained on the BT node project page.

The Bluetooth speciﬁcation deﬁnes two software layers abstracting the higher layers from the hardware character.

**Physical Bus Driver**:

This layer abstracts the characteristics of the physical bus; on the BTnodes the UART connecting the MCU and the Bluetooth radio.

**HCI Driver**:

This layer maps the interface of the underlying HCI layer (the upper layer embedded on the Bluetooth radio) into the programming model used for implementing the higher layers.

# TINY BLUETOOTH STACK

In this section, we report on the design and implementation of our Tiny Bluetooth stack. We compare it to the ETH Bluetooth stack for the BTnodes and to existing communication stacks developed for TinyOS.

# Design

Our goal with the Tiny Bluetooth stack was to make it possible for TinyOS programs2 to access a Bluetooth radio.2We could reuse most existing components of TinyOS except of course those abstracting hardware elements diﬀerentiating the BTnodes from the Mica and Rene motes, e.g., the u arts, the lads, and the clock.

We are not interested in implementing a full-ﬂedged Blue- tooth stack: while the three lower timing sensitive layers are embedded within the radio modules and thus a given, the higher computation intensive layers (l2cap, sdp, proﬁles) are essentially dealing with the connection of heterogeneous devices (e.g., a mobile phone and a headset) without providing solutions to problems such as cross layer optimizations or multihop routing, which characterize the sensor network regime. The ability to connect to heterogenous devices is not of paramount interest in sensor networks, which are composed of sensor nodes tailored to operate together as one unit. We thus decided not to implement those three higher layers. Rather, we focus the Tiny Bluetooth stack on the interface to the Bluetooth hardware and its mapping to TinyOS. Network assembly and multihop routing are implemented on top of the Tiny Bluetooth stack.

The Bluetooth speciﬁcation distinguishes the *Bluetooth host*, on which the three upper layers of the protocol are implemented from the *Bluetooth hardware* on which the three lower layers are implemented. The host and the hardware are connected via a physical bus (e.g., USB, UART, RS232). Figure 2(a) illustrates the connection of the Bluetooth host and the Bluetooth hardware.

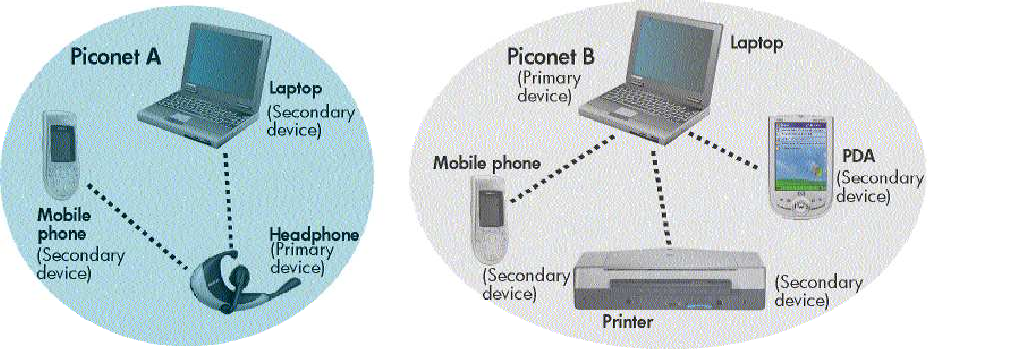
Our design is based on these hardware abstraction layers. The Tiny Bluetooth stack is composed of two components: *HCI Packet* that corresponds to the physical bus driver and *HCI Core* that corresponds to the HCI driver. The interfaces of both components are described in Figure 2(b) (we omit the events associated to each TinyOS command according to the asynchronous programming model).

TinyOS applications access the HCI Core component to communicate via Bluetooth. The HCI Core component sup- ports the main HCI commands related to (i) the initialization of the Bluetooth radio, (ii) the inquiries used for device discovery, (iii) the connection establishment, (iv) the trans- mission of data and (v) the reporting of errors.

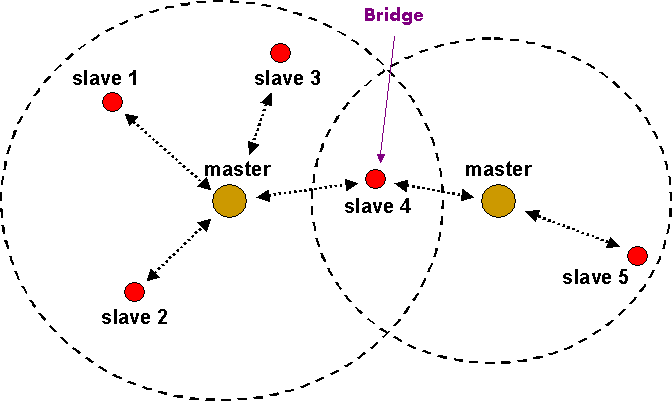
It relies on the services of the HCI Packet layer that maps send com- mands and receive events onto the UART component detail below the implementation of these components.

Piconet Topology (Modes of Bluetooth Communication) When more than 7 devices need to communicate, then one or more devices are put in park state.

* Bluetooth Low Power Modes are: *SNIFF, HOLD* and *PARK.*
* Park Mode: A devices disassociates from *piconet* when in park mode.
* The master consistently sends warnings to invite a slave to rejoin the *piconet.*
* The slaves can rejoin only if there a less than 7 slaves.

****

**FIGURE 2.6**

**2.5 Scatternet :-**

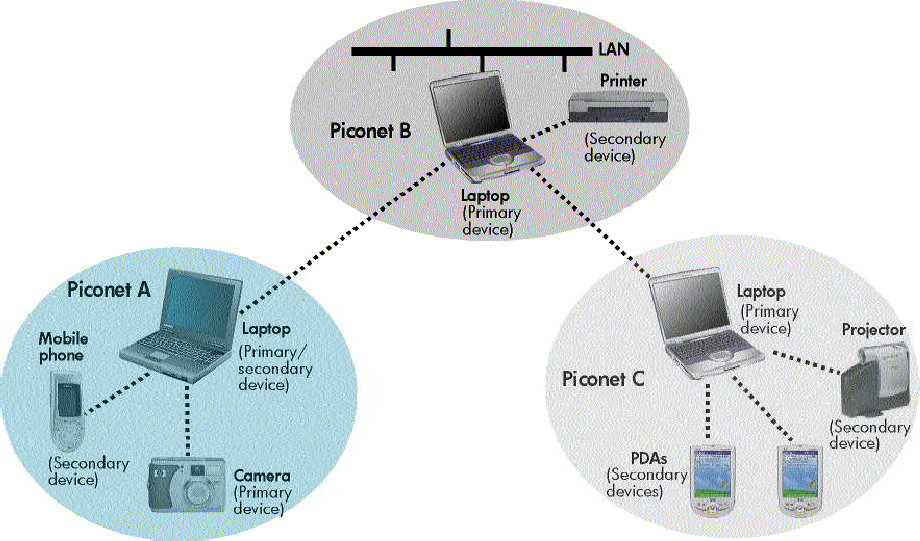
**FIGURE 2.6**

It is composed of interconnected piconets that maintain communication between more than 8 devices. Scatternets are formed when a device of one piconet that can be master or slave choose to act as a slave in second another piconet. Scatternets provide higher throughput. Also, in different piconets multiple-hop routing between devices is possible. That means at one time only single component can interface in one piconet so they hop from one to another relying upon the channel limit.

*Scatternet* consists of several *piconets* connected by devices participating in multiple *piconet.* Here, devices can be slaves in all *piconets* or master in one *piconet* and slave in other *piconets.* There is a ‘BRIDGE’ connecting 2 *piconets* which is also a slave in individual *piconets.*

**Advantages of *Scatternet*:**

* Higher throughput
* Multi-hope connection between in different piconets.



**Fig2.6.1. A Scatternet**

It is composed of interconnected piconets that maintain communication between more than 8 devices. Scatternets are formed when a device of one piconet that can be master or slave choose to act as a slave in second another piconet. Scatternets provide higher throughput. Also in different piconets multiple-hop routing between devices is possible. That means at one time only single component can interface in one piconet so they hop from one to another relying upon the channel limit.

# 2.6 BLUE TOOTH BASED SENSOR NETWORK

To produce different countless applications the developers are proving operability between different devices. Wireless sensor networks are example of such applications. It has various small devices consisting of sensing unit, microprocessor, power source and wireless communication bond. Important features of wireless sensor networks: During the task in process network nodes combines with each other. It has a particular attention on data.

As the position of junctions is not decided in the field and formation of smart sensor nodes is not planned it could be experienced that some sensor nodes close in such positions that they either can accomplish needed measurement or the error probability is more . To overcome these problems a small node of repetitive number is disposed. These nodes additionally merge and share data. And thus ensures better outcomes. The collected data is send to the users by “gateway” using multiple hop routes in the Smart sensor nodes that are scattered.

# 2.7 FUTURE OF BLUETOOTH

* BROADCAST CHANNELS: Adoption of Bluetooth into mobile phones and enable advertising models based on users pulling information from the information points.
* TOPOLOGY MANAGEMENT:

Automatic configuration of piconet topologies in scatternet situations.

* QoS IMPROVEMENTS:

Enable audio and video data transmission at higher quality, especially in best effort traffic being transmitted in the same piconet.

# CHAPTER 3

**3.1 BLUETOOTH HARDWARE ARCHITECTURE**

Bluetooth hardware has 3 prime function modules:

2.4 GHz Bluetooth radio frequency Trans receiver unit. Link controlling unit Host controller interface Host controller is made up of a digital signal processing section with link controller and central processor. Link controller composed of both hardware and software parts for the execution of base band processing and physical layer protocols. CPU core helps Bluetooth module to filter page request and to handle enquiries. Link manager is software which runs on the CPU and communicates to them with the help of link manager protocols.

**HOST CONTROLLER –**

Consists of a *Digital Signal Processing* part, having *Link Controller (LC) & CPU Core.* It interfaces to the Host environment.

LINK CONTROLLER −

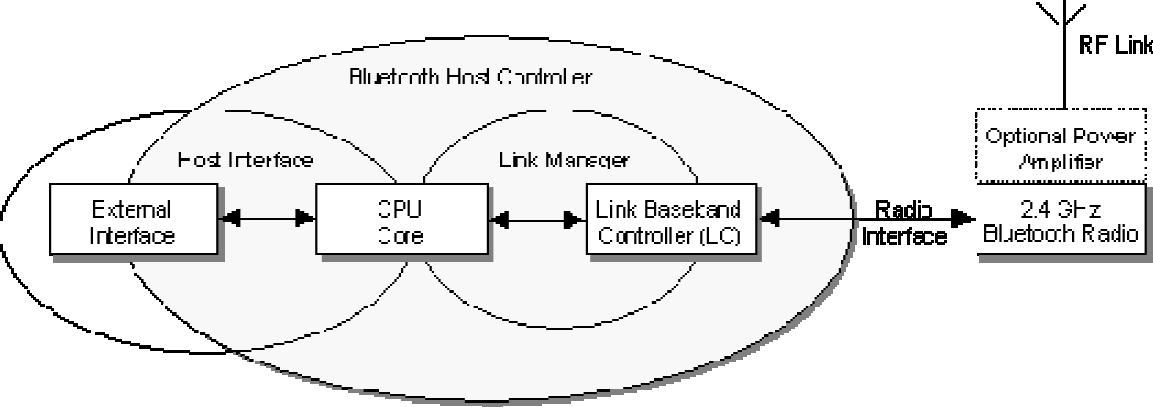
Consists of *Hardware & Software parts* to perform Base−Band Processing, and Physical Layer Protocols. Also perform slow−level digital signal processing to form connections.

CPU CORE –

Helps Bluetooth Module to handle Inquires and filter page request (not involving host device).

LINK MANAGER –

LM software runs on CPU core. LM discovers other remote LMs and communicates. Them via LMP (link manager protocols). Bluetooth Module also incorporates Higher−Level Software Protocols, governing the functionality and interoperability with other modules.

****

**Figure 3.7:** **Bluetooth Architecture**

Host controller is made up of a digital signal processing section with link controller and central processor. Link controller composed of both hardware and software parts for the execution of base band processing and physical layer protocols. CPU core helps Bluetooth module to filter page request and to handle enquiries .Link manager is software which runs on the CPU and communicates to them with the help of link manager protocols.

# MULTIHOP NETWORK ASSEMBLY

Because Bluetooth is connection oriented, the BTnodes need to be assembled into a multihop network before any.

# 3.2 Design

We must take into account the following characteristics of our platform when designing a multihop network assembly procedure:

*Bluetooth Connection Establishment*. Bluetooth connections are established between a master and a slave. The assembly procedure must establish the role of each node with respect to a connection. Note that nodes cannot exchange information before they have established a connection. In addition, slaves cannot communication with other slaves or overhear the communication taking place on other connections. As a result, we cannot use protocols involving spontaneous communication among neighbor nodes.

*Dual Radio Approach*. There are three possible con- ﬁgurations for each dual-radio node: (i) a node can be connected as slave on its two radios, (ii) a node can be connected as slave on one radio and as master with up to seven connections on the other radio, or (iii) a node can be connected as master with up to seven connections on both its radios.

Device Discovery Protocol. In order for two devices to discover each other, they must be in two complementary states at the same time: Inquiry and inquiry scan. The inquiring device continuously sends out is anybody out their messages hoping that these messages (known as ID packets) will collide with a device performing an inquiry scan. To conserve power a device

wanting to be discovered usually enters inquiry scan periodically and only for a short time known as the inquiry window. During this period, the device listens for inquiry messages.

The main challenges for the assembly procedure are thus:

A second approach consists in conﬁguring each node a priori. Each node is conﬁgured with a radio operating as a master and the other operating as a slave. This obliterates the need for the discovery and information exchange phases from the ﬁrst approach. The second approach constitutes a baseline. We chose to implement it on top of our Tiny Bluetooth stack.

Our baseline solution, inspired by BlueTree (discussed below), is the following. When a node boots up, it enables one of its radios (the slave radio) and starts looking for an- other node to connect to. In this stage, the node will not be discoverable/visible for other nodes; it considers itself an orphan looking for a network. If it discovers other nodes, it tries to connect to one of them as a slave4. If the connection succeeds, it will consider itself member of the network, and turn on its other radio (the master radio), making it discoverable and ready to accept connections from nodes that are not currently members of the network.

If the connection as a slave fails, it is because the master has reached its limit on the number of connections it can accept (recall that a master can connect to seven slaves). The node then tries to connect to one of the other nodes it has found in its vicinity. If there is no such other nodes ready to accept a connection then the node tries to connect again to the ﬁrst node it contacted. If a master connected to seven slaves receives three repeated connection requests from the same node N, then it disconnects one of its slaves and accepts the connection from the node N. It has been shown that when a master is connected to more than ﬁve slaves, additional slaves are in connection range with at least one of the connected slaves. As a consequence, it is probable that the disconnected node will ﬁnd a node that it can connect to in its vicinity.

**CHAPTER 4**

# WORKING

Sohrabi et al. describe two assembly procedures for the Sensoria dual-radio nodes (equipped with proprietary frequency hopping radios). The ﬁrst procedure proceeds as follows: the two radios of a node are tuned to two ﬁxed channels. During network assembly, messages are exchanged on those channels to form clusters (in each cluster a node is elected as a master while the other nodes are slaves). The constraint that a node cannot be master on its two radios ensures that the clusters constructed for the two radios are diﬀerent. Such overlapping clusters cover the entire network with a high probability; but there is no control over the topology of the connection tree. The second procedure relies on a discovery phase during which one of the radios is tuned to a control channel in order to broadcast information. Once a node has received information about its neighbors it takes a decision on its own role: a master remains tuned to the control channel and exchange data to its slaves on its second radio while a slave exchanges data on both its radios. Both procedures rely on tuning one or both radios onto a ﬁxed channel so that nodes can broadcast information to their neighbors. This is not possible with Bluetooth.

Basagni et al. proposed a set of scatternet formation protocols for Bluetooth devices: Blue Tree, BlueStar, Blue Mesh. The basic protocol is Blue Tree that constructs a multihop network with a tree topology. Blue Tree proceeds in two phases. First, every node obtains information from its neighbors (nodes spend enough time inquiring and responding to discover their neighbors). Second, a designated node, the blue root, initiates the construction of the Blue Tree. This node is assigned the role of a master and its neighbors becomes its slaves. Recursively, each node that has become a slave is assigned the role of master with respect to its un- connected neighbors. The blue root is thus the root of the connection tree, all the intermediate nodes in the connection tree are both slaves and masters, while the leaves are just slaves. The constraint that one master cannot be connected to more than seven slaves is handled via tree reorganization. This work rely on the assumption that Bluetooth supports scatternets (intermediate nodes are master and slaves on a single radio). Our assembly procedure is an adaption

of the Blue Tree protocol for the dual-radio conﬁguration. The main diﬀerence is that our approach does not include a discovery phase, rather nodes can join the network at any time (an unconnected node has its slave radio turned on and is looking for a master).

# 4.1 IN-NETWORK QUERY PROCESSING :-/

Tiny DB developed at UC Berkeley and Intel, processes declarative queries over sensor data within the sensor network. The proposed query language allows user to collect, ﬁlter and aggregate data produced in the sensor net- work. In-network query processing proceeds in two phases. First, the query is disseminated from a gateway node to all appropriate sensor nodes inside the network. This ﬁrst phase results in the establishment of a routing tree rooted at the gateway and spanning all the sensor nodes producing data for the given query. Second, data is processed when transmitted up the routing tree.

Each sensor node runs an instance of TinyDB, which is responsible for (i) establishing and maintaining the rout- ing tree, (ii) processing query fragments over data received via the network or read on a local sensor and (iii) trans- mitting processed data up the routing tree. Aggregates are processed in a distributed manner: each node computes a partial state record based on the values it reads and obtains from its children. The partial state records are transmitted and the actual aggregate value is obtained at the root of the routing tree.

TinyDB is a TinyOS library. It relies on basic TinyOS building blocks essentially for sending and receiving data over the network, for setting the power saving mode, for setting timer events and for reading sensor data.

TinyDB has been designed for the Mica motes , assuming communication based on connection less broadcast on a shared channel radio. The choice of a Bluetooth radio challenges some of the assumptions that underly its design and implementation.

# 4.2 Topology Management :-

For each query, TinyDB constructs a routing tree rooted at the gateway on which the query is submitted. The rout- ing tree is constructed by having nodes pick one of their neighbors as their parent.

This procedure for routing tree construction is very much similar to the one we described in the previous section for network self-assembly. The construction of the BlueTinyDB routing tree consists in constructing a directed version of the connection tree, with the edges oriented towards the gateway on which the query is submitted. The oriented edges are maintained as a parent-child relationship (independent from the master-slave relationship maintained by the self- assembly component).

BlueTinyDB constructs the parent-child relationship while disseminating a query. Each node retrieves the list of neighbors from the self-assembly component. The node from which the query is received becomes the parent, the other neighbors become children, regardless of whether they are master or slaves. There are two issues to be considered:

There might be several gateways connected to a cluster of nodes and there might be several queries submitted to diﬀerent gateways at the same time. Those queries will share the same connections. It is thus desirable that the connection tree is as bushy as possible, in order to minimize the height of the routing trees, regardless of the gateway to which a query is submitted.

*•*

Madden et al. [12] have proposed Semantic Routing Trees to restrict the distribution of a query to those nodes who actually participate in a given query. The idea is to maintain meta-data on each node to decide whether the nodes accessed through a given child will participate in the answer to the query (in which case the query is distributed further to this child) or not (in which case the query is not distributed to this child). The fact that all routing trees share the same connection tree facilitates the collection and the maintenance of this meta-data.

*•*

4.3Separated Channels:

TinyDB relies on the TinyOS MAC layer to avoid collisions between nodes transmitting during the same interval. Those collisions occur not only between parents and children but between any two nodes who are in transmitting range of each other. If the density of nodes is high...then we can expect a high rate of collisions. As a consequence, TinyDB monitors channel contention and adaptively reduces the number of packets sent as contention rises.

The problem of channel contention is not acute in the context of a Bluetooth radio. Each connection constitutes a separated channel. There is fenly interference between pairs of nodes hopping on the sanie frequency at a given point in time. This is a marginal problem.

Note that separated channels are expected to boost the performance of flooding as collisions are avoided between parents and children as well as between branches of the rout ing tree. Note however that separated channels do not permit the snooping used in TinyDB to optimize the performance of certain aggregate operators (such as MAX) or to com pen sate for the loss of some messages.

**Calibration of the Binodes:-**

The Tiny Bluetooth stack allows us to run experiments with the BJ nodes, in order to calibrate them with respect to throughput as well as energy consumption.

**Code Footprint :-**

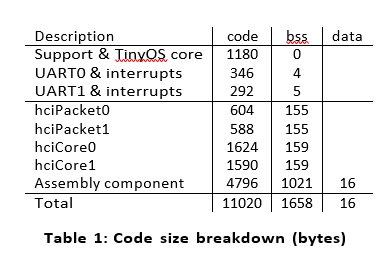
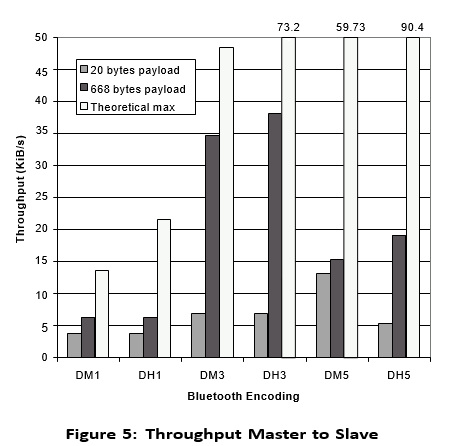
Our first challenge was to squeeze a tiny version of the Bluetooth stack within the Bloods. The code footprint for our Tiny Bluetooth stack is less than 3 KIB. It is thus comar cable to the native TinyOS stack (approximately 2 KIB [9]) and one order of magnitude less than the Smart-its Blue tooth stack (approximately 30 KiB ).

**4.4 Throughput :-**

Bluetooth specification promise high throughput. Can our Tiny Bluetooth stack take advantage of this potential on the Binodes?

We measured throughput on point-to-point connection between master and slave. Figure 5 shows the results we obtained for all possible Bluetooth encodings and two pay load sizes.

The distinction between initialized and non-initialized variables is interesting as non-initialized variables are not part .



**Figure 4.3.1 Figure 4.3.2**

Bluetooth deﬁnes 6 encoding that correspond to the com- bination of two levels of resistance to noise (due to diﬀerent levels of redundancy in the encoding – DM: high resistance and DH: low resistance) and three conﬁgurations of the TDM scheme (the node transmits for 1, 3 or 5 time slots). The two payload sizes we consider are 20 bytes, i.e., enough to transport a few integers which is the case for many sensor network applications, and 668 bytes which is the maximum HCI packet size, i.e., enough to transport high bandwidth traﬃc such as images or sounds.

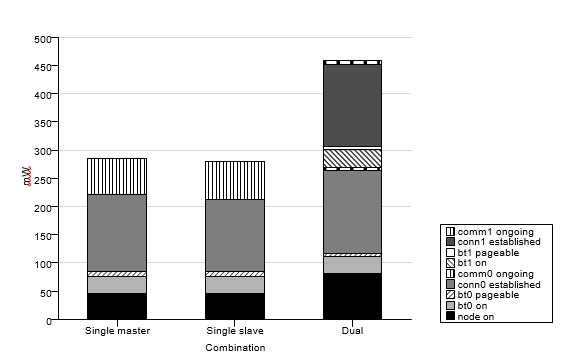
## 4.5 ENERGY CONSUMPTION

The throughput results are encouraging; however, energy consumption is the key metric for the calibration of the BTnodes. We thus measured both current draw and voltage9 for diﬀerent regimes of our experimentation platform. Figure 6 summarizes our results.

Our ﬁrst goal was to measure energy consumption for an idle BTnode (in black on the ﬁgure). According to the manufacturer, the 8 MHz MCU can use up to 12 mA at 5 V (60 mW) in idle sleep mode. With a slightly lower clock frequency, we observe a lower energy consumption (about 8 Note that we had to modify TinyOS to operate the UART at full speed. The UART relies on two registers for sending bytes: one, called the shift register, contains the byte currently being sent, the other contains the byte to send next. We ﬁxed the UART module so that it generates an event

each time the UART moves a byte to the shift register in- stead of generating an event each time the shift register is empty and the UART has no more data to send.

9We used an input voltage of approximately 5V. We observed that voltage varied slightly during the experiments. We thus decided not to assume constant voltage to compute the energy consumption.



# Figure 4.3

This is mostly due to the (unused but turned on) led on the BT tester, which consumes about 22 mW, as well as the voltage regulator and the serial voltage level converter. Turning a Bluetooth radio on consumes about 30 mW extra (in idle mode). Making the radio pageable, or in quarriable requires an additional 9 mW. A BTnode with a single radio waiting for a connection thus consumes 89 mW. In the dual radio case, turning on both radios and making the pageable and in quirable consumes about 155 mW. Nodes use about 136 mW to maintain connections. Additional experiments showed that putting a connection in sniﬀ mode saves a marginal 5 mW. This suggests that some optimizations might have been missed in the Bluetooth module design.

Once a connection is established sending or receiving data consumes an additional 65 mW when transferring at 6 KiB/- sec when using a single radio and 5 mW for each radio when transferring 10 packets per second.

The BTnodes consume ﬁve times more energy than the Mica motes doing nothing! This is due to the fact that the MCUs are placed in diﬀerent sleep modes. As we have seen with TinyDB in Section 5.1, the Mica motes favors applications that manage themselves the time they spend in sleep mode. This way, the MCU can be put to sleep in power save mode, where only the external clock can send wake-up signals, i.e., the motes do not get data from sensors or from other nodes while the MCU is in sleep mode.

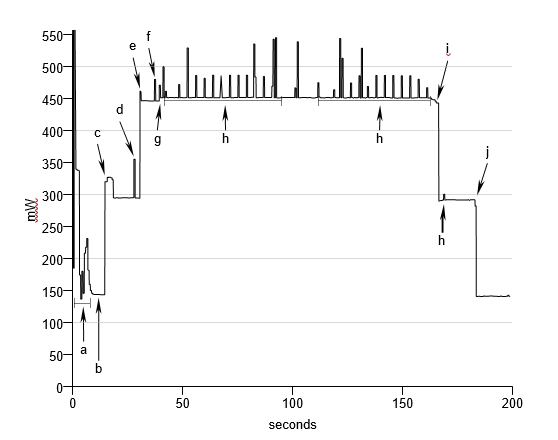
# 4.5 Network Self-Assembly :-

Because it is expensive to maintain a connection it is likely that the network will be assembled repeatedly when data needs be transmitted. Network assembly should thus be rapid and energy eﬃcient.

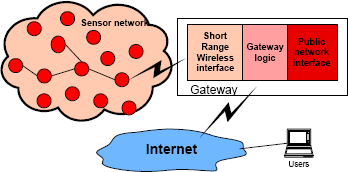
Figure 7 shows energy consumption as a function of time on a BTnode during network assembly. The ﬁgure is annotated with letters corresponding to the diﬀerent phases of the experiment. As the node is turned on (a), it initializes its *slave* radio. After a while, it discovers its parent-to-be, connects to it (b) and enables its *master* radio that becomes discoverable. The connection is established within 20 seconds.

For the sake of clarity, we turn on the children nodes one after the other. After 30 seconds, the ﬁrst child is turned on, it is detected and a connection is established on the node’s *master* radio (c). After 40 second, the second child is turned on and a second connection is established (d). Energy consumption corresponds to the calibration presented in the previous Section. Additional connections result in a very limited increase in energy consumption (about 3-4 mW).

Once those connections are established, the node routes data from both children to its parent: both children send packets with payload of 5 and 50 bytes every 10 seconds for a 100 seconds. Note that the children do not send in synch. The peaks of energy consumption we observe from 40 seconds until 160 seconds correspond primarily to the master .

**Figure 4.4**

**4.6 TOOH BASED SENSOR NETWORKS**

* Challenge: It is to ensure interoperability among various Bluetooth manufactures’ devices and to provide numerous applications.
* One such application is : WIRELESS SENSOR NETWORKS (WSN)
* Important features of WSN: Collaboration of network nodes during execution and Data Centric nature. Many smart sensor nodes scattered in the field collect data and send it to users via ‘gateway’ using multi−hop routes.
* WIRELESS SENSOR NETWORKS (WSN): WSN consists of number of small devices equipped with a sensing unit, microprocessors, wireless communication interface and power source*.*

## Figure4.5: Wireless Sensor Network

## 4.7 ALGORITHM FOR OPERATION OF NETWORKS

* Initialization of gateway and Bluetooth Inquiry Procedure.
* Discovery of Bluetooth device and Checking of major and minor devices.
* Setting of parameters and assigning type of devices and sensors.
* Description by Service−Class Field.
* Discarding of non−smart nodes.
* Else, service database of the discovered smart node is searched for sensor services.
* If no current sensor profile, then database is searched for serial port connection parameters.
* Lastly, Bluetooth link is established and data exchange with smart node starts.

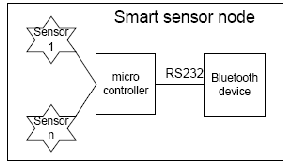
## 4.8 SENSOR NETWORK IMPLEMENTATION

OBJECTIVE: To build a Hardware plat form and generic Software Solutions to serve for research in WSN protocols.

Components of Sensor Network: Smart Sensor Nodesand Gateway

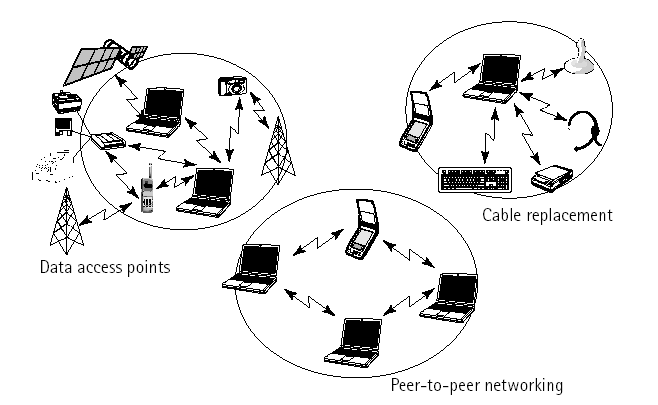
Gateway and Smart nodes are members of *piconets* and so, not more than 7 nodes can exist in the network.

Example: Pressure Sensor For implementation of Pressure Sensor as Bluetooth Node, following components are important: Bluetooth Device Sensors Microcontroller

TEDS − Transducer Electronic Data Sheet

**Figure 4.6 : SMART SENSOR NODE IMPLEMENTATION**

**4.9 Wireless connectivity over Bluetooth:**



**FIGURE 4.7**

The basic function is to provide a standard wireless technology to replace the multitude of propriety cables currently linking computer devices. Better than the image of the spaghetti-free computer system is the ability of the radio technology to the network when away from traditional networking structures such as business intranet. for, example imagine being on a business trip with a laptop and a phone. The Bluetooth technology allows interfacing the two. Then picture meeting a client and transferring files without cabling or worrying about protocols. That is what the Bluetooth will do.

4.10 A Wireless sensor network

**FIGURE 4.8**

Short range wireless interface

Gate way logic

Public network

interface

Internet

sensor network

Users

The main functions of a gateway are Communication with sensor Networks

* Shortage wireless communication is used.
* It provides functions like discovery of smart sensor nodes, generic methods of sending and receiving data to and from sensors, routing.
* Gateway logic
* It controls gateway interfaces and data flow to and from sensor network.
* It provides an abstraction level that describes the existing sensors and their characteristics.
* It provides functions for uniform access to sensors regardless of their type, location or N/W topology, inject queries and tasks and collect replies.
* Communication With Users

Gateway communications with users or other sensor networks over the Internet, WAN, Satellite or some shortage communication technology. From the user point of view, querying and tasking are two main services provided by wireless sensor networks. Queries are used when user requires only the current value of the observed phenomenon. Tasking is a more complex operation and is used when a phenomenon has to be observed over a large period of time. Both queries and tasks of time to the network by the gateway, which also collects, replies and forwards them to users.

**CHAPTER 5**

5.1 SENSOR NETWORK IMPLIMENTATION

The main goal of the implementation was to build a hardware platform and generic software solutions that can serve as the basis and a test bed for the research of wireless sensor network protocols.

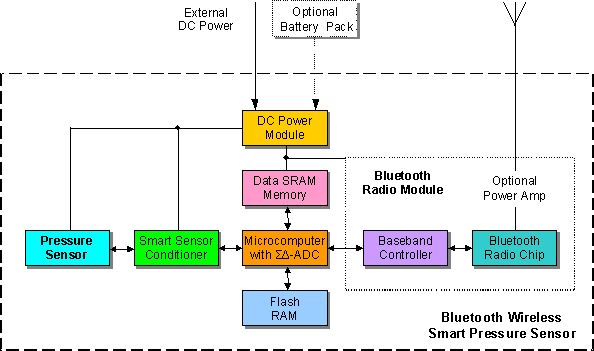
Implemented sensor network consists of several smart sensor nodes and a gateway. Each smart node can have several sensors and is equipped with a micro-controller and a Bluetooth radio module.

Gate way and smart nodes are members of the Piconet and hence maximum seven smart nodes can exist simultaneously in the network. For example, a pressure sensor is implemented, as Bluetooth node in a following way.

The sensor is connected to the Bluetooth node and consists of the pressure-sensing element, smart signal-conditioning circuitry including calibration and temperature compensation, and the Transducer Electronic Data Sheet (TEDS). These features are built directly into the sensor microcontroller used for node communication control plus memory for TEDS configuration information.

Smart Sensor Node Architecture

The architecture shown in figure can easily be developed for specific sensor configurations such as thermocouples, strain gauges, and other sensor technologies and can include sensor signal conditioning as well as communications functions.



**A Bluetooth wireless smart pressure sensor node**

**FIGURE 5.1**

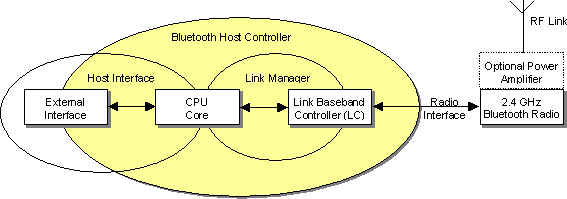
Conditioned along sensor signal is digitized and digital data is then processed using stored TEDS data. The pressure sensor node collects data from multiple sensors and transmits the data via Bluetooth wireless communications in the 2.4 GHZ base band to a network hub or other internet appliance such as a computer.

The node can supply excitation to each sensor, or external sensor power can be supplied. Up to eight channels are available on each node for analog inputs as well as digital output. The sensor signal is digitized with 16-bit A/D resolution for transmission along with the TEDS for each sensor. This allows each channel to identify itself to the host system. The node can operate from either an external power supply or an attached battery. The maximum transmission distance is 10 meters with an optional capability to 100 meters.

The IEEE 1451 family of standards are used for definition of functional boundaries and interfaces that are necessary to enable smart transducer to be easily connected to a variety of networks. The standards define the protocol and functions that give the transducer interchangeability in networked system, with this information a host microcomputer recognized a pressure sensor, a temperature sensor, or another sensor type along with the measurement range and scaling information based on the information contained in the TEDS data.

A blue tooth module consists primarily of three functional blocks – an analog 2.4 GHz., Blue tooth RF transceiver unit, and a support unit for link management and host controller interface functions.

The host controller has a hardware digital signal processing part- the Link Controller (LC), a CPU core, and it interfaces to the host environment. The link controller consists of hardware and software parts that perform blue tooth based band processing, and physical layer protocols. The link controller performs low-level digital-signal processing to establish connections, assemble or disassemble, packets, control frequency hopping, correct errors and encrypt data.



FUGURE 5.2

5.2 Bluetooth module Hardware Architecture :-

The CPU core allows the blue tooth module to handle inquiries and filter page request without involving the host device. The host controller can be programmed to answer certain page messages and authenticate remote links. The link manager(LM) software runs on the CPU core. The LM discovers other remote LMs and communicates with them via the link manager protocol (LMP) to perform its service provider role using the services of the underlying LC. The link manager is a software function that uses the services of the link controller to perform link setup, authentication, link configuration, and other protocols. Depending on the implementation, the link controller and link manager functions may not reside in the same processor.

Another function component is of course, the antenna, which may be integrated on the PCB or come as a standalone item. A fully implemented blue tooth module also incorporates higher-level software protocols, which govern the functionality and interoperability with other modules. Gate way plays the role of the Piconet’s master in the sensor network. It controls establishments of the network, gathers information about the existing smart sensor nodes and sensor attached to them and provides access to them.

5.3Discovery Of The Smart Sensor Nodes :-

Smart sensor node discovery is the first procedure that is executed upon the gateway installation. It goals to discover all sensor nodes in the area and to build a list of sensor’s characteristics and network topology. Afterwards, it is executed periodically to facilitate addition of new or removal of the existing sensors. The following algorithm is proposed.

When the gateway is initialized, it performs Bluetooth inquiry procedure. When the blue tooth device is discovered, the major and minor device classes are checked. These parameters are set by each smart node to define type of the device and type of the attached sensors. Service class field can be used to give some additional description of offered services. if discovered device is not smart node it is discarded. Otherwise, service database of the discovered smart node is searched for sensor services. As currently there is no specific sensor profile, then database is searched for the serial port profile connection parameters. Once connection strings is obtained from the device. Blue tooth link is established and data exchange with smart mode can start.

# 5.4 A WIRELESS SENSOR NETWORK:-

Wireless sensor network provides two important actions querying and tasking.

Querying: When there is need of current value of observed event, queries are used. It is more composite operation and is useful when an event needs to be noticed for a long time.

* These two services querying and tasking are allocated to the system through “gateway” which also forwards the collected responds to users.
* The main functions of a gateway are
* Interaction with sensor networks.
* Short wireless transmission is used.
* It contributes functions like finding of smart sensor nodes
* Provides techniques of sending and receiving data from sensors, routing.
* It controls gateway attachment and dataflow to and from sensor network.
* It provides standard of dealing with ideas that gives detail about the current participating sensors and their characteristics.
* It allocates function for consistent approach to sensors without being affected by their type, location or network topology, introduce queries and tasks and gather respond.
* Communication with Users.
* Gateway communication with user and another sensor networks through the internet, WAN, Satellite or other short communication technology.

# 5.5 ADVANTAGES :-

Within this article, we’ll walk through those implications and highlight a few scenarios where Bluetooth-based smart sensor networks work well (and when they do not). Bear in mind that these implications are based on the idea that you may build a mesh network to solve Bluetooth range issues.

**Implications Of Range Performance Issues**

**Battery life**: When nodes are in a mesh, each node has to act as both a sensor and a repeater. It must constantly listen for, relay, and route network traffic. Thus, you’ll likely experience decreased battery life.

**Reliability**: If one node in a key location drops out of your mesh, it could take down a good chunk of the network.

**Cost**: Bluetooth is relatively inexpensive, but this cost will vary drastically depending on network coverage in a mesh. Covering a small home, for example, may not be too expensive—but covering a large office building will be. Part of that is because you have to add repeater nodes where they aren’t necessarily needed just to ensure that your network reaches to the furthest points.

**Latency**: To get a message from point A to point B in a mesh network, the data will have to travel through points C, D, E, F, and G first. Thus, to get a message where it needs to go, you could introduce a few seconds of latency.

# 5.6 APPLICATIONS:-

1. Constant sensing
2. Health monitoring
3. Event detection & local control of actuators
4. Smart buildings
5. Smart grid and energy control system

# 5.7 CONCLUSION:-

Wireless sensor networks are fascinating research area with multi feasible applications and with many solutions. They are combination of various tiny devices having the ability of interacting and dealing with data. Bluetooth is an easy and suitable option for data communication in sensor networks. To plan routing and application-level procedures, we overlook multiple affairs related to MAC layer, physical layer, application layer and routing layer. For the automatic link up and information exchange, the Bluetooth devices need to bring within the range of another device.

Our experiments with the BTnodes suggest that Blue- tooth based sensor nodes could be appropriate for a niche of applications exchanging unpredictable bursts of data during a limited time period. It is diﬃcult to predict that Bluetooth will emerge as an alternative of choice for a larger class of sensor network ap- plications. First, as long as scatternets are not supported we will have to rely on a dual-radio approach which is ex- pensive and energy ineﬃcient.

Second, using Bluetooth for commercial purpose requires a certiﬁcation (to guarantee that heterogeneous devices can communicate) which is irrelevant and unrealistic in the context of sensor networks. Third, the encapsulation of the three lower layers on hard- 10A similar experiment without data transmission also exhibits energy peaks but with a more regular intensity.

11We observe that connections take most of the time 5 to 10 seconds with some outliers requiring 20 to 30 seconds. We have studied the characteristics of device discovery and connection establishment in previous work.

* 1. **Glossary:**
* Illict = not allowed by low.
* A piconet = an ad hoc network that links a wireless user group of devices using Bluetooth technology protocols.
* Sniff = every packet between a selected device and the device it is communicating with, even when the link is encrypted.
* TinyOS = embedded, component-based operating system and platform for low-power wireless devices.
* Salves = (in past times) a person who was owned by another person and had to work for him/her.
* Scatternet = A chain of piconets created by allowing one or more Bluetooth devices to each be a slave in one piconet and act as the master for another piconet simultaneously.
* Priori = using facts or principles to decide the probable effects or results of something.
* Assembly = Utilizes machines, equipment, and/or workers to assemble parts and materials in a pre-defined sequence until there is a finished product.
* Cabling = used to refer to electrical or electronic cables.
* Robust= strong.
* Redundant = having lost your job because your employer no longer needs you.
* A gateway = a network node used in telecommunications that connects two networks with different transmission protocols together.
* Sensor = a device that is used to record that something is present or that there are changes in something.
* IEEE = Institute of Electrical and Electronics Engineers.

**References**

1. Mark Weiser, “The Computer for the Twenty-First Century”, Scientific American, 1991.
2. Wireless World Research Forum, “Book of Visions”, [http://www.wireless-world-research.org](http://www.wireless-world-research.org/)
3. Scenarios for Ambient Intelligence, EU IST Advisory Group
4. H. Gharavi and K. Ban, “Video-based multihop ad-hoc sensor network design,” in roc. World Wireless Congress, San Francisco, CA, May 2002, pp. 469–474.
5. J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust", ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99),

Seattle, WA, August 17-19, 1999.

1. G.J. Pottie, W.J. Kaiser, “Wireless Integrated Network Sensors”, Communications of the ACM, May 2000, Vol. 43, No. 5
2. J.M. Rabaey, M.J. Ammer, J.L. da Silva Jr., D. Patel, S. Roundy, “PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking", IEEE Computer Magazine, July 2000
3. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, “System Architecture Directions for Networked Sensors”, Proceedings of the ASPLOS 2000