WIRELESS AD-HOC CONTROL NETWORKS

A thesis submitted in fulfillment of the

requirement for the award of the degree

MASTERS OF ENGINEERING - RESEARCH

from

UNIVERSITY OF WOLLONGONG

by

SHENGRONG BU

BACHELOR OF ENGINEERING

SCHOOL OF ELECTRICAL, COMPUTER AND

TELECOMMUNICATION ENGINEERING

(2005)

Certification

I, Shengrong Bu, declare that this thesis, submitted in fulfillment of the requirement for the award of masters of Engineering by Research, in the School of Electrical, Computer and Telecommunication Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualification at any other academic institution.

…………………

Shengrong Bu

October 2005

Contents

List of Figures

List of Tables

List of Abbreviations

ACL	Asynchronous Connectionless Link
A/D	Analog to Digital Converter
ADC	Analog to Digital Converter
BEB	Binary Exponential Backoff
CAN	Controller Area Network
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
D/A	Digital to Analog Converter
DAC	Digital to Analog Converter
DA	Directory Agents
DDC	Direct Digital Controller
GUI	Graphical User Interface
HCI	Host Controller Interface
HTTP	Hypertext Transport Protocol
I/O	Input/Output
IC	Integrated Circuit
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
JVM	Java Virtual Machine
L2CAP	Link Layer Control and Adaptation Layer Protocol
LAN	Local Area Network
LCD	Liquid Crystal Display

MAC	Media Access Control
MCU	Microcontroller
MEMS	Microelectronic-mechanical Systems
NCAP	Network Capable Application Processor
NCS	Networked Control System
PC	Personal Computer
PCB	Printed Circuit Board
PHY	Physical Layer
RAM	Random Access Memory
RF	Radio Frequency
SA	Service Agents
SCO	Synchronous Connection-Oriented
SDP	Service Discovery Protocol
SDP_DPU	Service Discovery Protocol Data Unit
SLM	Salutation Managers
SLP	Service Location Protocol
STIM	Smart Transducer Interface Module
TCP	Transmission Control Protocol
TEDS	Transducer Electronic Data Sheet
TII	Transducer Interface Module
TINI	Tiny InterNet Interface
TS	Transducer Status
UART	Universal Asynchronous Receiver Transmitter
UPnP	Universal Plug and Play

UA	User Agents
WASNets	Wireless Ad-hoc Sensor Networks
WACNets	Wireless Ad-hoc Control Networks

Abstract

Control systems have gone through major changes over the last decades, evolving from fully analogue systems to distributed digital controllers. The initial architecture of the computer control systems was central, as all the sensors and actuators were connected to a high performance computer. As the number of sensors and actuators deployed in the control systems increased significantly, this approach revealed many weaknesses such as high cost, poor reliability, poor maintainability, and limited extensibility.

With the emergence of low cost microprocessors, the computer controlled system adopted a more distributed architecture. Direct Digital Control (DDC) was the first distributed control system formally recognised in the literature and industry. Currently, control networks represent the latest development of the control systems architecture in industry. They combine localized intelligent digital controllers networked to a supervisory computer for data exchange and synchronization.

A new concept called Wireless ad-hoc control networks (WACNets), as the next stage in the evolution of control systems architecture is studied in this thesis. Such systems consist of a large number of nodes with sensing and/or actuation, local intelligence and control, and data processing and communication components. The size, number, density, capabilities and location-dependency of such nodes are determined by the specific application for which the nodes are employed. The protocols and algorithms that run on the nodes could also provide self-organizing and cooperative capabilities for random deployment of the nodes.

In addition to the development of a conceptual model for WACNets, a test-bed for validation of the concept based on IEEE 1451 compliant Smart Sensor and Bluetooth standard is designed and developed. In order to validate the test-bed and the concept, a monitor is also developed which provides interaction and communication with the nodes in the network from a host computer. The other contributions of the thesis include the design and development of a service discovery protocol based on Bluetooth Standard, a suite of software driving the test-bed, and validation of the test-bed. The results of the validation as presented in the thesis are quite encouraging and strongly indicate that the approach is feasible.

Acknowledgements

Many valued people have contributed to this project and this thesis, and helped me not only academically but also emotionally.

First of all, I sincerely thank my supervisors Professor Fazel Naghdy and Dr. Philip P. Ciufo for their continued guidance and support.

I would like to thank Stephen Davis for his immense help in regard to Java Programming and Tim Browne for his invaluable help.

Likewise, I want to thank the staff in the faculty workshop and all of my other friends for their assistance.

Finally, I leave my special thanks to my family, especial my parents, for their endless love, which gave me courage to overcome all my problems.

Chapter 1 Introduction

1.1 Background & Overview

The advances made in computer networking have had significant impact on industrial controllers. These control systems were traditionally central and hierarchical where large centrally located main frame or mini-computers were connected directly to dumb I/O racks through the I/O ports of the computer. The first major change took place when a distributed architecture was adopted. In this configuration, a central computer supervised distributed direct digital controllers (DDC) with some local intelligence in the I/O modules.

In the 1980's, with the development of computer networking, network connectivity was introduced into DDC systems. In a networked control system, information produced by different nodes is shared among the intelligent digital controllers. The benefits of networking in a distributed control system have resulted in the development of a number of network standards for control systems.

Compared with centralized control, control networks offer many advantages including less wiring, lower costs, easier setup and maintenance, and higher reliability. In light of these advantages, control networks have been used in a broad range of applications such as automated homes and offices, and automobiles etc. Figure 1.1 shows the evolution of control systems architecture from centralised networked systems.

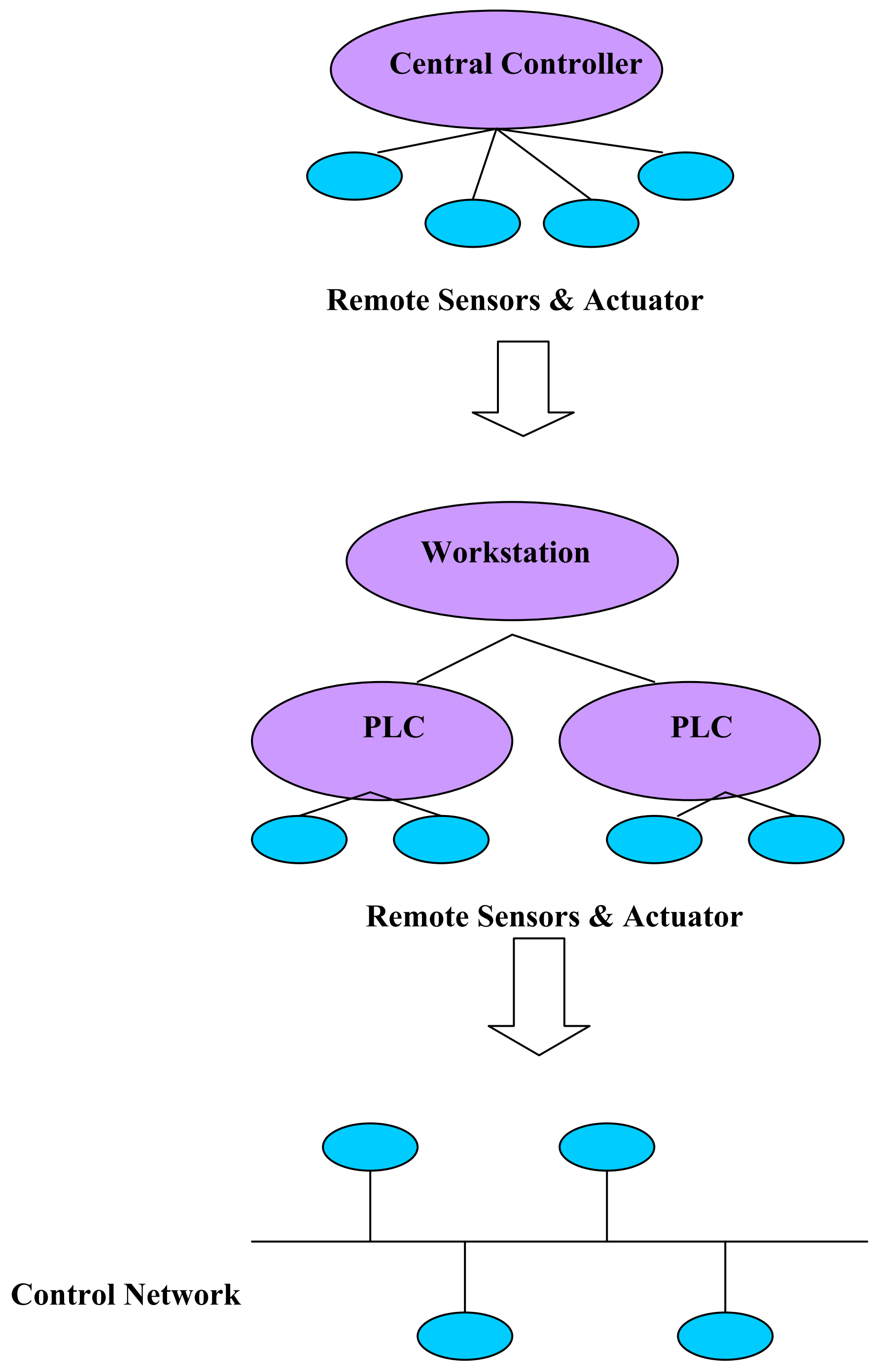


Figure 1.1 Evolution of Control Systems Architecture

With the advent of mobile computing and wireless communication, ad-hoc distributed systems are becoming feasible and attractive. Such systems represent peer to peer wireless communication among intelligent devices without any fixed infrastructure.

The peers form a network of diversified devices including sensors, actuators, appliances, and PDAs.

In this work, a new concept called WACNets, designed for distributed and remote monitoring and control is explored. Such systems represent the next stage in the evolution of distributed control and monitoring. Recent advances in mobile computing, wireless communications, MEMS-based sensor technology, low-powered analogue and digital electronics, and low-power RF design have created opportunities for the introduction of WACNets.

WACNet explores a framework for organic, evolutionary and scalable method of integrating a large number of intelligent and heterogeneous nodes. Each node consists of sensing and/or actuation, local intelligence and control, data processing and communication components. The size, number, density, capabilities and location-dependency of such nodes will be determined by the specific application for which the nodes are employed. Ideally they are expected to be low-cost, low-power, multi-functional and small in size.

1.2 Aim and Contributions of Thesis

As the primary aim, this thesis has studied the feasibility of WACNets and has developed a test-bed for its further research and development.

The work has made the following contributions:

(a) A comprehensive and critical review of the historically significant and recent projects related to control systems architecture has been conducted. The focus has specifically been on control networks and distributed control.

(b) A conceptual model of WACNets and its operation has been developed and proposed.

(c) A prototype WACNets node based on IEEE1451 and Bluetooth standards have been developed as a test-bed for the study of WACNets.

(d) A service discovery protocol based on Bluetooth standard has been developed and validated for WACNets.

(e) The performance of the test-bed and service discovery protocol has been validated through carefully designed experimental work.

(f) The thesis overall demonstrates that WACNets is a feasible concept and has potential for full implementation in control systems.

1.3 Structure of Thesis

The content of the thesis is organized in 8 chapters. An overview of the work and the contribution of the thesis are provided in Chapter 1. Chapter 2 provides a critical review of the literature on the control systems architecture and its evolution towards control networks. The conceptual model of WACNets is developed and presented in Chapter 3. The focus of Chapter 4 is on the design and construction of the hardware components of the first generation test-bed of WACNet.

The proposed WACNet Service Discovery Protocol (SDP) is presented in Chapter 5. The design of the software developed for different elements of the test-bed are described in Chapter 6. Chapter 7 provides the results of the experimental work carried out to validate the concept and the operation of the test-bed. Finally, the outcomes produced by the work are critically reviewed in Chapter 8, some conclusions are drawn, and possible future directions for the work are presented.

Chapter 2 Literature Review

2.1 Introduction

As the first step in research, a major literature survey on the work associated with WACNets was conducted. A critical summary of this study is provided in this chapter.

The chapter begins by reviewing the control systems and control networks widely deployed in the industry. It then continues by introducing WACNets. In the final part, the significance of the concept proposed in this research compared to the previous work will be described.

2.2 Evolution of Control Systems Architecture

Control systems have gone through major changes over the last decades. The initial systems were fully analogue. The signal measured by a sensor was processed in analogue form, the control signal was then calculated by an analogue signal based on the difference between the sensory signal and set point of the system. The control signal was used to drive the actuator. There were different processes used to carry out the signal processing in the controller including electronics, fluidic and hydraulic. A modular and structured collection of electronic components called analogue computers were also used as controllers.

With the introduction of computers in the 70's, a new approach to design and implementation of control systems emerged. The sensors were connected to the control computer via an analogue to digital converted. The computer processed the sensory signal and derived the control signal according to a particular control law. A digital to analogue converter prepared the signal in analogue form to drive the actuator [1].

The computer-based approach to control systems created new opportunities to implement complex control algorithms such as non-linear and heuristic control, rather impossible previously.

The initial architecture of the computer control systems was central, as all the sensors and actuators were all connected to a high performance computer. As the number of sensors and actuators deployed in the control systems increased significantly, this approached revealed many weakness such as high cost, poor reliability, poor maintainability, and limited extensibility [1].

As the cost of computers started to decrease and microprocessors introduced, the computer controlled system adopted a more distributed architecture. Direct Digital Control (DDC) was the first distributed control system which was formally recognised in the literature and industry. In this approach, a series of digital computers are supervised by a central computer. The sensors and actuators connected to the digital controllers did not still have much intelligence. The control loop for a particular process was implemented on the digital controller whereas the supervisory computer provided the set points and ensured the smooth and coordinated operation of the digital controllers connected to it [2]. This is illustrated in Figure 2.1.

Decentralised architecture of DDC offers increased reliability and reduced need for communication between local sensing and the central computer. The major control loops are performed locally and only critical data is transmitted centrally. This also makes DDC more efficient and reliable, with reduced maintenance overhead [2]. The central stations which are known as the front end computers are primarily used for user interface, monitoring, logging of data, and data management.

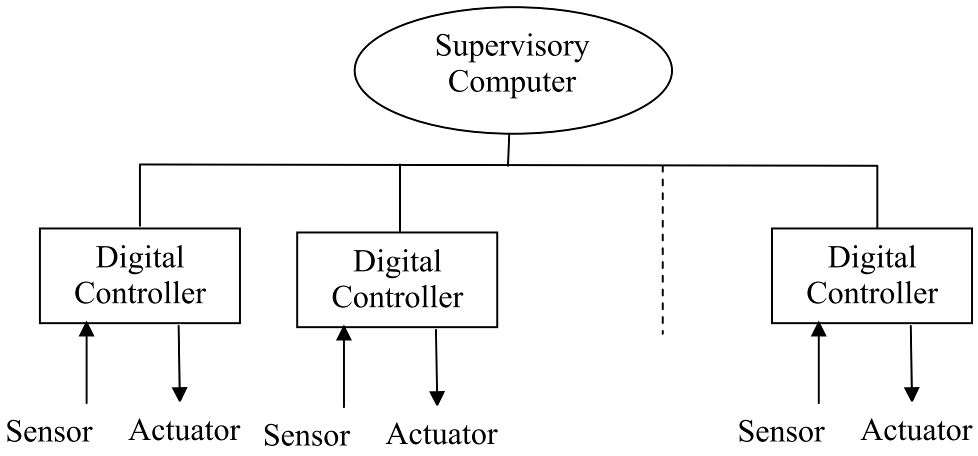


Figure 2.1 Controller

2.3 Control Networks

In the 1980's, with the development of computer networking, network connectivity was introduced into DDC systems. The network, as the medium connecting multiple intelligent devices, allows them to store data, communicate, and share information, as well as display and print information. In a networked control system, information produced by different nodes is shared among the intelligent digital controllers. The benefits of networking in a distributed control system have resulted in the development of a number of network standards for control systems. These standards are reviewed in this section.

Controller Area Network (CAN) [3] [4] was developed mainly for the automobile industry which is currently used in factory automation. DeviceNet (CAN Bus) is a deterministic, message-oriented protocol optimized for short messages. In a CAN-based network, each message is given a priority that determines network access in case of simultaneous transmission. Thus collisions do not destroy important messages as the message with the higher priority is delivered successfully [3] [4]. The major disadvantage of DeviceNet is the slow data rate of only 500 kb/s for up to 40 devices. CAN is not suitable for transmitting messages of large size [4].

ControlNet [5] is a highly deterministic network protocol in which the maximum waiting time before sending a message is characterized by the token rotation time. A token is passed around the network, and the node that has the token is allowed to transmit. Transmission continues until the nodes have finished transmitting or until the time they hold the token reaches the limit. At this point, the token is regenerated and passed on to the next logical node on the network. Since only one node can transmit at one time, message collision never occurs. The main disadvantage of ControlNet at low network loads is that a significant amount of time is spent passing the token around the logical ring [4]. In the case of an emergency, a node cannot gain access to the network until the token has completed its rotation around the ring.

Profibus [4] is a typical example of token-passing bus control networks. A significant amount of work has been carried out to improve the performance of this system. In order to satisfy real time constraints, Tovar [6], [7], and Vitturi [8] introduced the selection method of an important timer called Target Rotation Time (T_{TR}). The timer works by analysing the token cycle time in the data link layer of Profibus-FMS [9]. By varying (T_{TR}) for Profibus-FMS, Hong [10] demonstrated the characteristics of the communication delay. He [11] also proposed a scheme called bandwidth allocation to fulfil the real-time requirements. The scheme operates by transmitting periodic data within time limits in the Profibus-FMS. Lee [12] developed some methods to deal with the performance management of Profibus-FMS. In spite of such research, it is still challenging to choose suitable protocol parameters for token passing protocol under realistic situations.

Ethernet [4] [13] is becoming a prime network control candidate due to its low cost, availability, flexibility, and high communication rates. Ethernet uses the carrier sense multiple access with collision detection (CSMA/CD) mechanism for resolving contention in the communication medium in case of simultaneous data transmission. Ethernet has not been originally designed for real time applications and hence it is not a suitable protocol for control networks.

The major limitation is the nondeterministic nature of communication. The standard Binary Exponential Backoff BEB has also created a series of issues such as unfairness and substantial performance degradation, enabling Ethernet to transmit a large message size with a small amount of data [13]. Moreover in Ethernet, the end-to-end communication is not guaranteed as a message may be discarded after a series of collisions. In addition, Ethernet does not support message prioritization. Hence, the protocol does not guarantee that a specific message is not eventually lost due to collision. Message collision can significantly affect the network performance.

Several solutions have been proposed for better utilization of Ethernet in control applications.

Lee et al. [14] evaluated the performance of the switched Ethernet for networked control systems. In the switched Ethernet, the hub is an active device that repeats whatever received from the input port to the destination port. Also the connection between a station and the hub is a full-duplex link. The characteristics mentioned above make the switched Ethernet free of frame collision. They also showed that a maximum communication delay of Ethernet was about 418.8 msec, while that of switched Ethernet is several hundred microseconds. Therefore, the communication delay of the switched Ethernet is sufficiently small for real-time industrial networks. However, the cost is too high for deployment in industry [15].

Venkatramani et al. has proposed the implementation of a virtual token-passing scheme over Ethernet in order to avoid packet collisions [16]. Token management is executed by a higher-layer protocol rather than the MAC and, thus, their approach requires only software modification rather than hardware. However, this approach virtually adds another full-scale MAC layer on top of the Ethernet MAC layer. Thus, it potentially has high implementation complexity and may degrade the OS kernel performance because of many complicated mechanisms such as token management, which is also known to be the main disadvantage of FDDI [15].

In a network control system, reducing network traffic is the most effective way to minimize the effect of time delays on its performance. Yook et al [20] have predicted the status of a system on the network via estimators. The performance benefits gained in this method by reducing the network traffic outweighs the increased computational cost of the estimator and has a lower impact on system performance delays. When the difference between actual and estimated values of the status is more than a specific threshold, the transmission of state updates occurs.

LonWorks [1] technology developed by ECHELON Co. in USA has been widely used in industrial and home automation. LonWorks is a control network technology which uses a control network protocol called LonTalk. This is also known as the ANSI/ EIA 709.1 control networking standard. LonTALK is media independent, which means that it can be run over any physical media [1]. Networks can range in size from a small network embedded in a machine to large networks of thousands of devices controlling an entire factory or plant. LonTalk is a layered, packet-based, peer-to-peer protocol providing a comprehensive set of services for sending and receiving messages across the network without needing to know the topology of the network, or other device names or addresses. LonTalk is different from a TCP/IP network. Accordingly, LONworks does not support IP networking. Shahnasser and Wang managed to establish some interoperability between LONTalk and TCP/IP and have demonstrated that it is possible to control various devices over TCP/IP and LONTalk [19].

With the advent of mobile computing and wireless communication, ad-hoc distributed systems are becoming feasible and attractive. Such systems represent peer to peer wireless communication among intelligent devices without any fixed infrastructure. The peers form a network of various diversified devices including sensors, actuators, appliances, PDAs, laptops, etc.

2.4 WACNets

In this thesis a new architecture called WACNets for control systems is introduced. In WACNet, a synergy between ad-hoc wireless networks and intelligent control node creates a plug and play, and highly distributed systems.

A WACNet consists of a large number of geographically distributed intelligent and heterogeneous nodes with sensing and/or actuation, local intelligence and control, data processing and communication components. The processing unit can perform signal processing and control depending on the services required from the node, and the type and the number of sensors/actuators attached to it. The size, number, density, capabilities and location-dependency of such nodes will be determined by the specific application for which the nodes are employed. Ideally they are expected to be low-cost, low-power, multi-functional and small in size.

WACNet represents a radical shift from conventional control systems. It addresses more realistic applications and represents a natural progression from conventional control systems currently used in various industries including process control and building services. The proposed system offers new capabilities not present in the currently available industrial control systems. These new capabilities include:

(a) True ad-hoc structure, which simplifies the design, maintenance, extensibility, and scalability of the system.

(b) True peer-to-peer communication with no central supervisor, which introduces unnecessary overhead and complexity.

(c) Cost effective scalability.

(d) Cost effective wireless communications, which will significantly reduce the cost of commissioning and control of a control system.

(e) Provision of an evolutionary system through its ability to reconfigure in an autonomous fashion and provide an optimal distribution of resources.

(f) Robustness with respect to both disturbances and uncertainties.

(g) Interoperability in heterogeneous system environments.

(h) Verifiability of the system features at various levels of computational complexity.

Chapter 3 Overview of WACNet

3.1 Introduction

A new concept called WACNets, designed for distributed and remote monitoring and control is explored in this project. In this work, WACNets are considered as the next stage in the development of distributed control and monitoring. Recent advances in mobile computing, wireless communications, MEMS-based sensor technology, low-powered analogue and digital electronics, and low-power RF design have created opportunities for the introduction of WACNets.

In this chapter the overall concept is introduced and its major proposed characteristics are highlighted.

3.2 Structure of WACNet

A WACNet consists of a large number of geographically distributed intelligent and heterogeneous nodes with sensing and/or actuation, local intelligence and control, data processing and communication components. The processing unit can perform signal processing and control depending on the services required from the node, and the type and the number of sensors/actuators mounted on it. The size, number, density, capabilities and location-dependency of such nodes will be determined by the specific application for which the nodes are employed. Ideally they are expected to be low-cost, low-power, multi-functional and small in size.

Each node contains sufficient intelligence to make local decisions based on global/regional system state, and also has the ability to change its behavior based on external/ internal stimuli. In a WACNet, a collection of nodes dedicated to achieve a particular task are known as a cluster. A set of clusters working together towards a common objective forms a network. These nodes are not mobile, however, the membership of the clusters and the configuration of the network can change dynamically as the overall system evolves or some nodes fail. WACNet is classified as ad-hoc as there is no provision in the architecture for network infrastructure or central administration.

3.3 Evolution of WACNet

A WACNet goes through different stages of operation during its lifetime:

(a) **Boot up**: In this phase, the nodes are initialized and when ready, are placed in a listening mode. In this mode, the node listens for other devices and builds an identity list of devices within range. Once the node has identified and listed all the nodes within its communication range, it moves to the next phase.

(b) **Service Discovery**: During this period, the nodes advertise the services they can offer including the sensing and actuation capabilities. They will also provide information about their information processing capabilities and control strategies that they may offer. This phase could also be referred to as a "capability exchange" procedure.

(c) **Cluster formation**: Neighboring nodes with common or complimentary functions form clusters of nodes and start to coordinate and synchronize their activities to achieve their common goal. Within a cluster, one node is designated as the master of the cluster and the rest as slaves. It also keeps a registry of nodes associated with that cluster. The service list offered by each node includes the identification of the master node.

The task of cluster information can be accelerated if the functions expected from the network are broadcast to all the nodes. This information assists the nodes to match their services with the network requirements. This stage represents the first step in self-organisation and cooperation. Any node is capable of being a master and will have sufficient resources to perform this role. The information registry describing a cluster is held by all the nodes and not just the master. This adds redundancy to the cluster identity in case the master node drops out for some reason.

(d) **Network operation**: In this phase all the clusters in the network operate concurrently. The status of each cluster is hopped through the network to a monitoring station. The transfer of data is performed through communication of clusters in different clusters. Information is moved across the network in a typical ad-hoc approach.

(e) **Network Evolution**: As the operation of the network evolves, it might be required to get involved in new tasks. This will result in further self-organization as described in stage (c). Further evolution of the network can take place with the addition of new nodes due to:

I. failure of a node.

II. new requirements of the network, as it enters into new a stage of operation.

III. addition of new devices to the environment requiring control and monitoring.

3.4 Application of WACNet

WACNets have potential for employment in a variety of applications including environmental control of large building complexes, smart home control for security, identification and personalization, robot control and guidance in automatic manufacturing systems, and interactive toys. In addition, WACNet can be employed in distributed control, information dissemination, in-network processing and other distributed computations such as sensor fusion, classification, and collaborative target tracking [20]. They can have direct applications in commercial, industrial, agriculture, health and defense sectors. There are also possible applications in personal and institutional security, radiology, and medicine. A typical WACNet for building services is illustrated in Figure3.1.

The realization of WACNet introduces a number of constraints on the network nodes and the network itself. The network should be robust with respect to communication disturbances and delay, be able to reconfigure and to redistribute the available resources optimally with respect to the changing environment and be energy-efficient. In WACNet, limited bandwidth and energy constraints may cause problems because the throughput of the network may not be large enough to transmit all the necessary measurements and control signals. Moreover, excessive consumption of energy reserves of the nodes through frequent transmissions may cause the nodes to quickly fail [21].

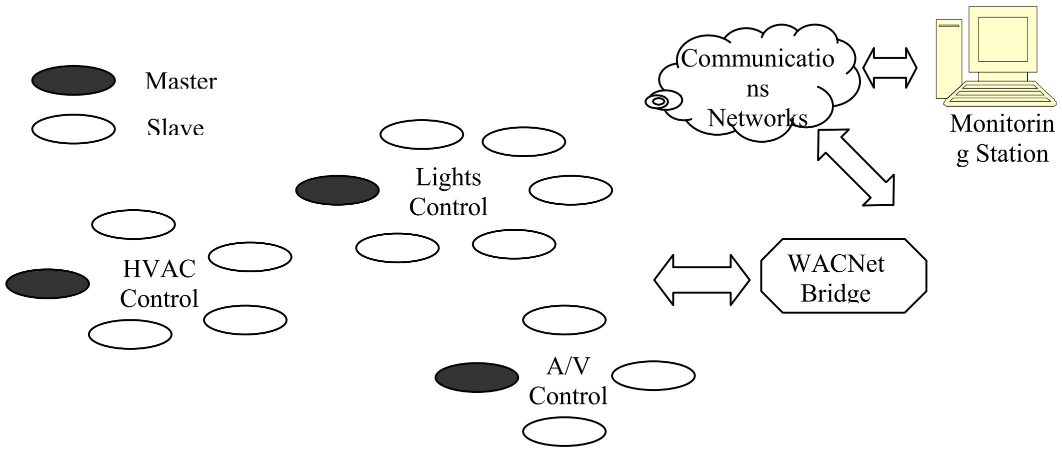


Figure 3.1 a Typical WACNet for Building Services

3.5 WACNet and Sensor Networks

While the WACNet in concept, approach and application is novel and original, it has some roots in other recent developments including ad-hoc wireless networks and ad-hoc wireless sensor networks. A comparative study of the WACNet with such technologies can highlight further the unique characteristics of the WACNet and the services it can offer.

Such systems are essentially meant to sense the environment and inform the users [22, 23] of their findings. The sensory information obtained by each sensor provides one perspective of the environment. The ultimate goal is that the infusion of this information creates an overall perception of the environment.

WASNets are formed through integration of a large number of spatially distributed sensors. They are utilitised for sensing and monitoring of the parameters of a particular environment. Each sensor is expected to be physically small and low cost. The main components of an intelligent sensor are a radio transceiver, a microcontroller and the power supply which is usually a battery.

The sensor network design is influenced by many factors, which include fault tolerance; scalability; production costs; operating environment; sensor network topology hardware constraints; transmission media; and power consumption [24]. These factors serve as a guideline for the design and development of various protocols intended for sensor networks. In addition, these influencing factors can be used to compare different schemes. These factors have been addressed by many researchers. However, none of these studies has a full integrated view of all factors that are driving the design of sensor networks and sensor nodes.

The WACNets and Sensor Networks share the following characteristics in contrast to other wireless ad hoc networks [24]:

- There is much larger number of nodes compared to ad hoc networks.
- The nodes are deployed densely.
- The nodes have a high probability of failures.
- The configuration of the networks changes quite often.
- The main communication model used by the nodes is broadcast in contrast to most ad hoc networks which are point-to-point.
- There are limitations in the nodes as far as power, CPU capacity and memory is concerned.
- There is no global identification allocated to the nodes due to overhead introduced by it.

In spite of such similarities, WACNets are also different from sensor networks in both the scope of their application and the mode of their operation. In WACNet

(a) The number of nodes can be significantly higher.

(b) The nodes can have a higher density.

(c) The nodes are more prone to failure due to higher complexity.

(d) The topology and configuration of the nodes can change more frequently.

(e) The nodes can have limited energy, processing power and memory.

(f) The nodes may not have global identification due to the amount of overhead they can introduce.

Similar to conventional networks, WACNet protocol stack consists of different layers. The design and architecture of protocols governing each layer should address the specific issues associated with it.

WACNet goes beyond information processing and perception of the environment. Each node, after sensing the local status and awareness of the regional and possibly global status, interacts with the environment, controls and regulates a series of processes. The information communicated between the nodes is more than the data obtained from the environment. It will include status of the local processes, synchronization signals between the nodes, the required set points and overall objective of the operation at different levels, and integrity of the overall system.

True self-organization of the nodes and collaboration between them is a necessity in WACNet. This is determined based on the task expected to be carried out by a cluster of nodes and the interaction they should perform with their environment and other nodes to achieve that task.

In more recent publications on sensor networks references are made to the inclusion of actuators in such sensors. There is, however, no reference to the role and function of such actuators in the proposed architecture of the networks. In fact, the strategy offered for the networks is mainly focused on sensors and with no provisions for actuators. Such differences make the WACNet a unique concept which not only embraces ad-hoc sensor networks, but offers a novel approach for introducing ad-hoc control networks in both conventional and new applications.

3.6 Implementation of WACNet

In WACNet, a smart transducer interface standard is needed to isolate the choice of transducers from the choice of the network. In this study, the architecture offered by IEEE1451 is employed to realize the WACNet concept.

Up until a decade ago, there was no common digital communication interface standard defined for transducers and the network capable application processors (NCAPs). Each transducer manufacturer built its own interface. Consequently, transducer manufacturers could not afford to support all the control networks for which their products would be suitable.

The recent development of the IEEE 1451 [25] standard has had a profound influence on the construction and implementation of distributed control networks. The objective of the 1451 family of standards is to make it easy to create solutions using existing networking technologies, standardized connections to smart devices, and a common software architecture. At the process connection level, 1451 provides standard ways of creating totally self-describing measurement and control devices. This allows the selection of the best-in-class products from the preferred vendors-true plug & play [26, 27]**.** It is also possible to choose the most appropriate network for an application. IEEE 1451 standard is deployed in WACNet to take advantage of its characteristics. The architecture of an IEEE1451 smart sensor unit in its network environment is illustrated in Figure 3.2.

In earlier versions of IEEE 1451 standard, a STIM could only connect to one NCAP via a 10-wired TII. This had a negative effect on hardware overhead. To overcome this problem, wireless technology has been deployed in IEEE1451 standard to implement a new interface standard. The wireless interface standard offers many advantages including the absence of costly, time consuming and tedious cabling and verification, lower infrastructure costs associated with STIM installation, the ease of repositioning or removing of STIMs, and the potential for deploying wireless STIMs in extremely remote and hazardous locations.

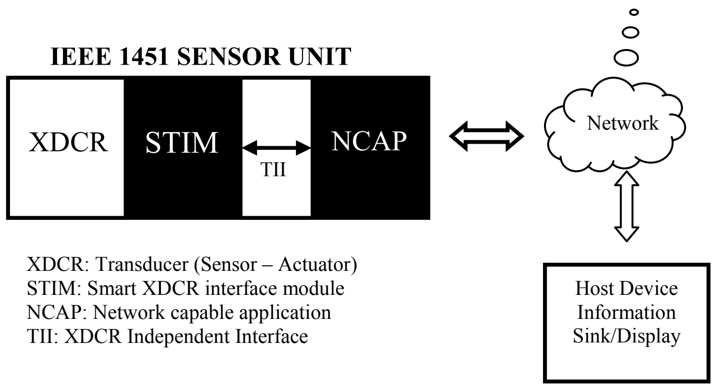

IEEE 1451 SENSOR UNIT
XDCR
STIM
TII
NCAP
Network
Host Device Information Sink/Display
XDCR: Transducer (Sensor – Actuator)
STIM: Smart XDCR interface module
NCAP: Network capable application
TII: XDCR Independent Interface

Figure 3.2 Conceptual model of a complete IEEE1451 Smart Sensor unit

Considering the design and functional requirements of WACNet, a combination of IEEE1451 compliant Smart Sensor and Bluetooth standards are employed to develop the first generation test-bed for the study of WACNet. The communication module of STIM, NCAP and Network can be implemented in several different ways using BT communication. In one configuration, STIM communicates with an NCAP module over TII wired link, while NCAP module communicates the data from STIM via a Bluetooth module as illustrated in Figure 3.3. This model limits the usage of an NCAP to one STIM.

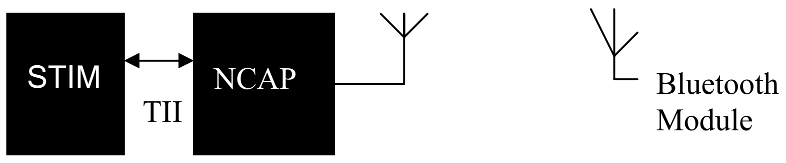


Figure 3.3 Communication Module

In the proposed system, STIM uses a Bluetooth to communicate with an NCAP module, which is connected to the network via a wired module. In this kind of network, the monitoring station can exist far away from the Plant (or at any point within the Internet.) The structure of wireless implementation of IEEE1451 in WACNet is shown in Figure 3.4.

In order to provide a gateway function for one or more wireless STIM units, the NCAP module has Bluetooth and Ethernet interfaces. The TII protocol, which defines a form of serial communication between the STIM and NCAP, is implemented above the Bluetooth link layer.

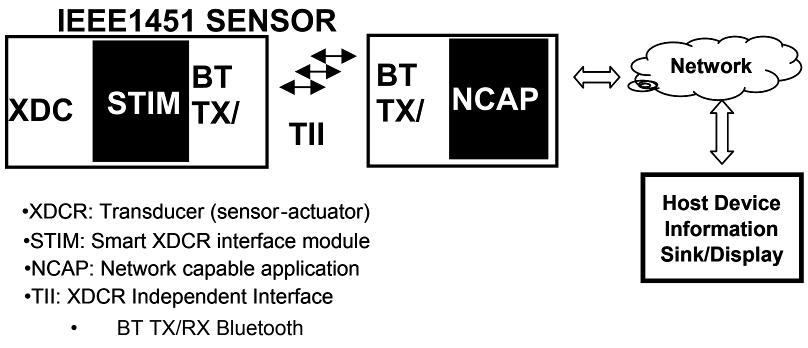

IEEE1451 SENSOR
XDC
STIM
BT TX/
TII
BT TX/
NCAP
Network
Host Device Information Sink/Display
•XDCR: Transducer (sensor-actuator)
•STIM: Smart XDCR interface module
•NCAP: Network capable application
•TII: XDCR Independent Interface
• BT TX/RX Bluetooth

Figure 3. 4 Wireless Model of WACNet nodes

The NCAP module receives data from STIM units via a Bluetooth receiver, transforms these Bluetooth packets to Ethernet packets, and forwards them to a monitoring station, referred to in this thesis as Monitor. An operator can monitor and/or control the whole system in real time through the local monitor system on demand.

In order to achieve wide coverage for wireless STIM units over the space of a factory area, NCAP units could be distributed across an Ethernet backbone, as shown in Figure 3.5.

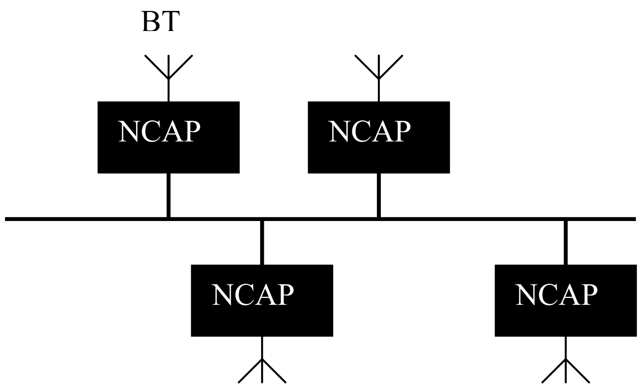


Figure 3.5 NCAP network

3.7 Clustering of the Nodes

The configuration and type of WACNets varies based on the number of STIMs and NCAPs present in the system. Accordingly, the structure of WACNets can be classified into three different categories of one-layer cluster model, two-layer cluster model and multi-layer cluster model.

The one-layer cluster model is the direct communication model as illustrated by an example in Figure 3.6. In this model, the nodes organize themselves into local clusters. Each cluster which consists of up to seven STIM slave nodes and one NCAP master node is called a Piconet. NCAP receives data from STIMs in the cluster, performs data fusion, and transmits the aggregated data to its own cluster or other neighboring cluster replacing the data.

In the example shown, Piconet 1 can perform a localized control function. In the piconet, the NCAP reports the local sensing and process information and control status to a remote Monitor when it is required. In Piconet 2, the NCAP receives sensory data from STIM nodes, and then condenses them before further transmission. Based on the broadcast data received from Piconet 2, NCAP in Piconet 3 activate the actuator in the piconet.

The second type of WACNet is illustrated by an example in Figure 3.7. This type can be employed in a situation where the number of STIM nodes are much larger that the NCAP. In this mode, the STIMs organize themselves into local clusters and communication with NCAPs via the STIM master nodes.

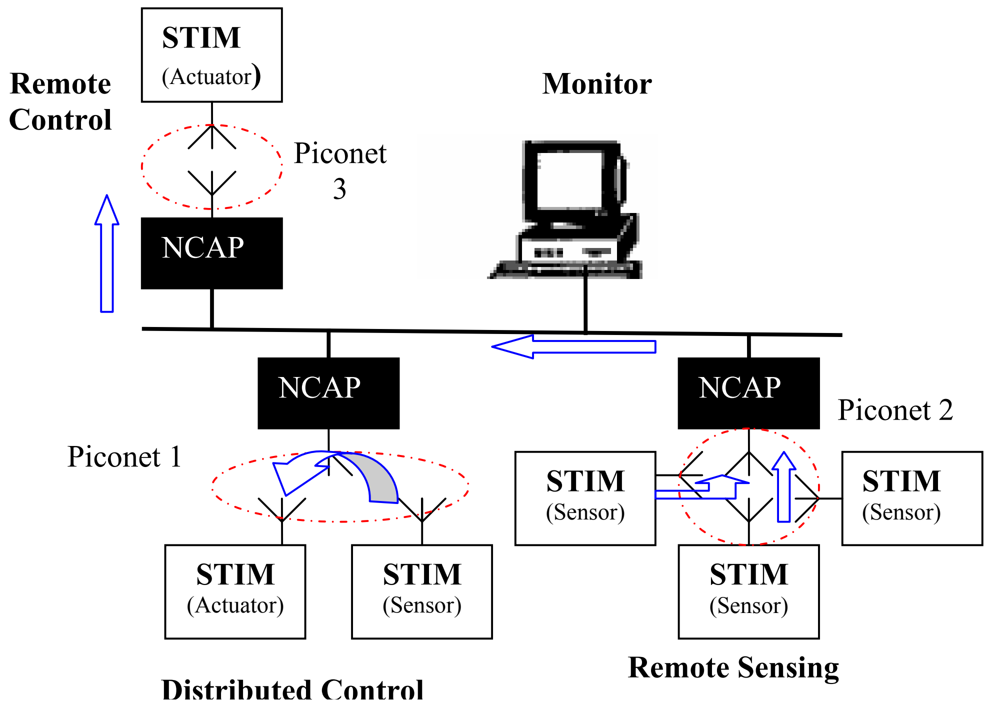


Figure 3.6 One-Layer Cluster of WACNets

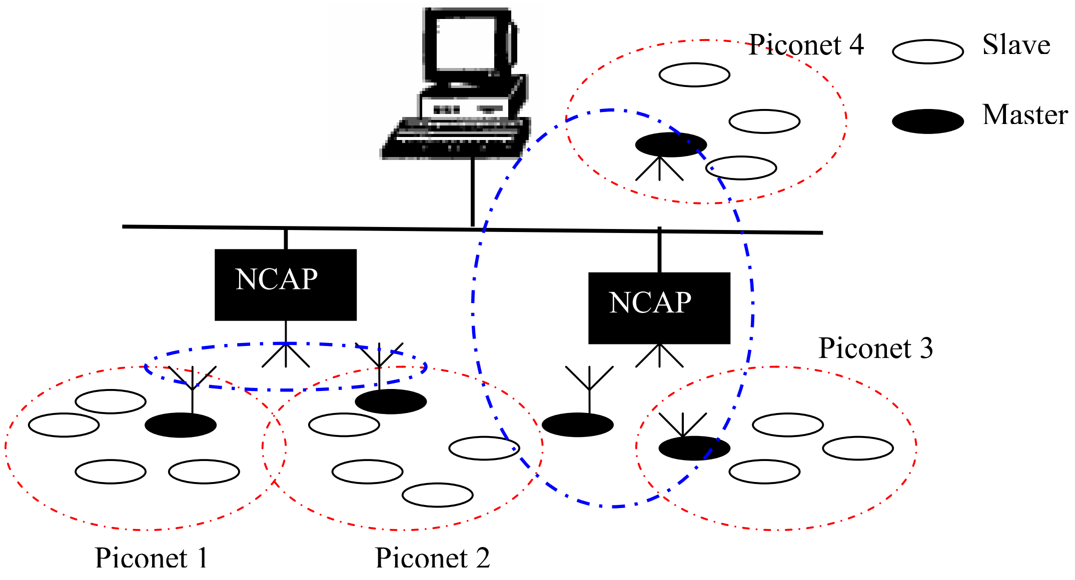


Figure 3.7 Two-Layer Cluster WACNet

In WACNets, STIM nodes might be distributed widely over a large area, and therefore some nodes may not fall in the communication range of NCAPs as illustrated in Figure 3.8. In this example, Piconet 8 cannot directly communicate with the NCAPs.

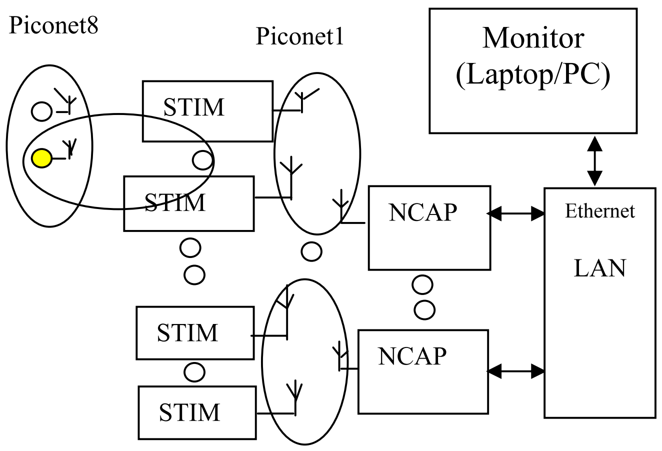


Figure 3.8 Piconet 8 has no direct access to NCAPs

3.8 Energy Cost of Nodes in WACNets

In a wireless communication system, transmission (E_{Tx}) and receiving energy costs (E_{Rx}) in each node are calculated as follows [21]:

$$E_{Tx}(k,d) = E_{elec}k + \varepsilon_{amp}kd^{\lambda} \qquad (3.1)$$

$$E_{Rx}(k) = E_{elec}k \qquad (3.2)$$

where *k* is the length of the message in bits, *d* the distance between transmitter and receiver node and λ the path-loss exponent ($\lambda \geq 2$), a factor that depends on the RF environment, and is generally between 2 and 4 for indoor environments, E_{elec} is the energy being dissipated to run the transmitter or receiver circuitry and ε_{amp} is the energy dissipation of the transmission amplifier.

According to equation 3.1, transmission of data through several short intermediate hops of data across different nodes is more energy efficient than using a long hop. Because of these advantages, the clustering and multi-hop routing algorithm employed in this work employ data aggregation methods to greatly reduce energy dissipation.

3.9 Clustering Connections

In the third model, there are more a large number of STIMs in the WACNet. Neighboring nodes with common or complimentary functions form clusters. The clusters satisfy different requirements. For example, in a building, light sensors need to update information back to monitor every several minutes. However, the temperature information does not change as fast. Hence, the real time constraint on light sensors is tighter than temperature sensors.

Another important factor in the operation of WACNet and the connectivity of the nodes is the amount of power available in the node for its communication and operation. Based on these factors, an index called Device Grade is defined which determines the quality of the connectivity of the nodes, clusters and NCAPs in WACNet. This index is defined by

$$DG = w_p * DeviceLeftPower + w_t * Class\,\mathrm{Re}\,altime - w_r * Dis\,\tan\,ceToNCAP \qquad (3.3)$$

In this relationship, $w_r, w_p,$ *and* w_t represent the weights for distance between the STIM master node and NCAP node, Power left in the master device, and class of real-time requirement for the cluster, respectively. The class of real-time is associated with the priority of the cluster in WACNet. In this equation, $w_r < w_p < w_t$, which indicates that the class of real-time requirement of tasks in the node is the most important parameter compared to the power left in the device, and distance to the NCAP.

For the multiple clusters in the vicinity of an NCAP, the cluster with higher *DG* has higher priority to connect to the NCAP than others. Other clusters might be forced to connect to the NCAP via other clusters. Hence, the whole network system forms a hierarchical structure of the *DG*s.

Chapter 4 WACNet Testbed

4.1 Introduction

This chapter provides a review of the prototype model designed and developed as the first generation testbed of WACnet. It identifies the major characteristics of the design, describes the specific problems encountered in the process and the solutions developed.

In this project, the required hardware has been broken down into three main categories, according to the concept of WACNet introduced in Chapter 3, including:

(a) Monitor hardware

(b) NCAP hardware

(c) STIM hardware

4.2 The Prototype Model

The diagram in Figure 4.1 illustrates the overall structure of the WACNet testbed developed in this work. Each STIM can communicate with other STIMs and NCAPs via Bluetooth modules. Computers connected to the network are able to remotely access the TINI, and therefore monitor the WACNet via a web browser. In this chapter different elements of this system will be described.

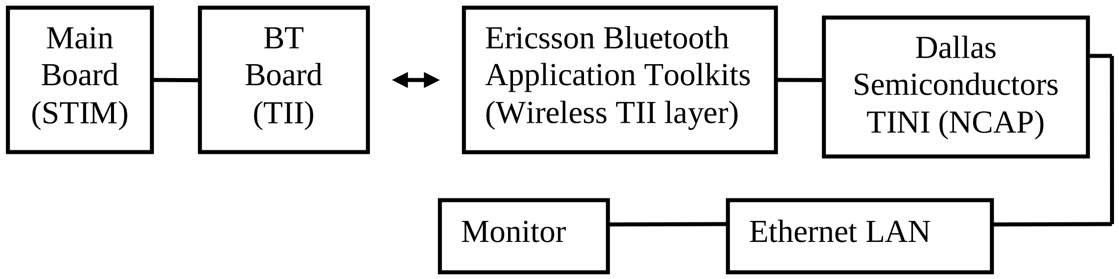

Main Board (STIM)
BT Board (TII)
Ericsson Bluetooth Application Toolkits (Wireless TII layer)
Dallas Semiconductors TINI (NCAP)
Monitor
Ethernet LAN

Figure 4.1 – Block Diagram of the Developed WACNet Testbed

4.3 Monitor Hardware

In the first generation testbed of WACNet, a computer with a Network Interface Card (NIC) serves as a monitoring device observing the operation of WACNet through a web-based management console. The system can be also controlled by the other computers on the Internet with Internet Explorer 6.0, which are illustrated in Figure 4.2.

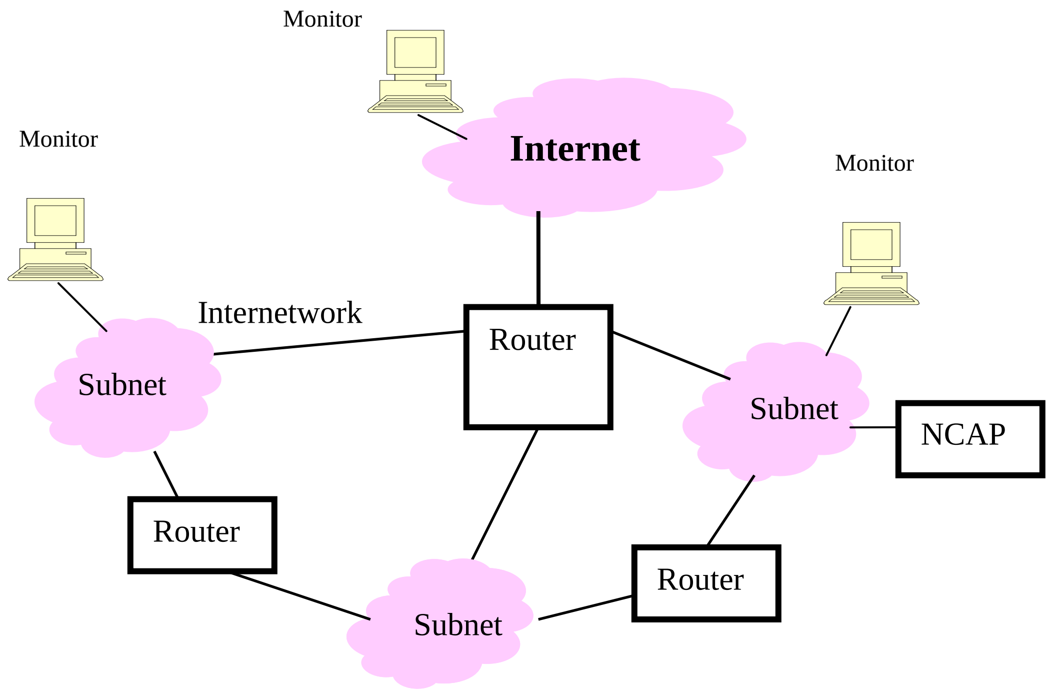


Figure 4.2 the Distribution Diagram of Monitors

The above diagram clearly describes that the Monitors and NCAPs can be located in the same subnet, different subnets, and even different networks.

4.4 NCAP hardware

In this work, the role of the NCAP module is performed by Tiny Internet Interface (TINI) manufactured by Dallas Semiconductors. TINI is chosen because of its low cost and the functions it provides. The major advantage of the TINI board is that it can be accessed from almost anywhere in the world, if the network is connected to the Internet and properly configured. In this system, the TINI board takes on the role of a Hypertext Transfer Protocol (HTTP) server, enabling the nodes associated with it to become network enabled. Therefore, any computer on the network can access the hardware without any special need for extra drivers.

4.4.1 TINI board

The TINI is a Unix/Linux based embedded system with a range of I/O channels including a 10BaseT network adaptor for LAN connection and dual RS232 serial ports. These ports provide the necessary I/O for this project.

A connection between the NCAP and Monitor is made using a CAT5 crossover cable, while TINI and Ericsson Bluetooth Toolkit connecting each other using a 9 way female-to-female (DB9 to IDC) connector. A USB A-B cable is also deployed to supply the Bluetooth Application Toolkit power.

4.4.2 Wireless TII

An Ericsson Bluetooth Application Toolkit is deployed as the most favorable way to incorporate wireless TII in the first testbed of WACNet. The Toolkit is a ready-made single board solution, comprising of a Bluetooth radio module, 2.4GHz antenna and an RS232 serial command interface.

4.5 STIM hardware

STIM can perform different functions including local sensing, actuation and control depending on the services required from the nodes, and the type and the number of sensors/actuators attached to it. The functional block diagram of a typical STIM is illustrated in Figure 4.3.

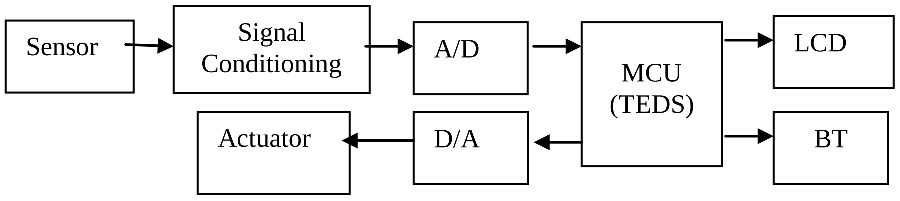


Figure 4.3 Architecture of STIM

In this system, the measured signal from the sensor is amplified, filtered and digitized via an A/D converter. The digitized data is then sent to a microcontroller unit (MCU) for processing. The digital signal produced in the node is converted into analogue before it is sent to drive the actuator. Sensors and actuators can be in different nodes. With the deployment of a Bluetooth wireless device, all data collected in the node can be transmitted wireless to other nodes.

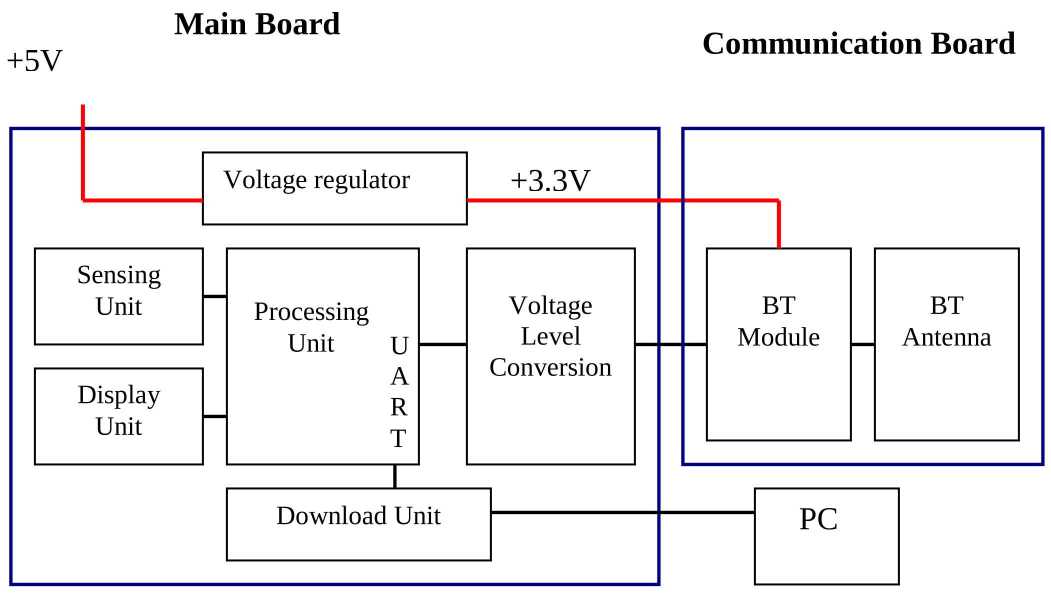


Figure 4.4 the Block Diagram of STIM

In the first generation testbed of WACNet, the STIM is broken down into two parts including the main board and the communication board for easy testing. The block diagram of the hardware device attached to the STIM is shown in Figure 4.4. Each part is described as follows.

4.5.1 Main Board

1) Processing Unit

On the main board, a microcontroller is needed to communicate with the sensing unit and the Bluetooth module, and connect with LCD. Hence, the microcontroller chosen should have at least one Universal Asynchronous Receiver Transmitter (UART) to communicate with the Bluetooth unit and enough I/O ports to provide extra functionality such as connection of LCD and sensing unit, or even actuator.

The microcontroller chosen for this project is the PIC16F877 which meets the basic requirements mentioned above. This keeps the size of the circuit small and the cost low. Furthermore, the PIC16F877 supports in-circuit programming, allowing user code to be downloaded to the PIC while it remains on the board. The main features of PIC16F877 are [28]:

- It is capable of operating up to 20 MHz.
- It is equipped with five I/O ports.
- It has sufficient amount of both program and user memory including

  8k x 14 FLASH;

  368 x 8 bytes of RAM;

  256x 8 bytes EEPROM;
- One UART;

A suitable crystal oscillator should be chosen for the microcontroller. In this project, the crystal is selected based on the error probability of USART (Universal Synchronous Asynchronous Receiver Transmitter) in the chip at a particular crystal frequency. The following equation 4.1 [29] defines the *SPBRG*.

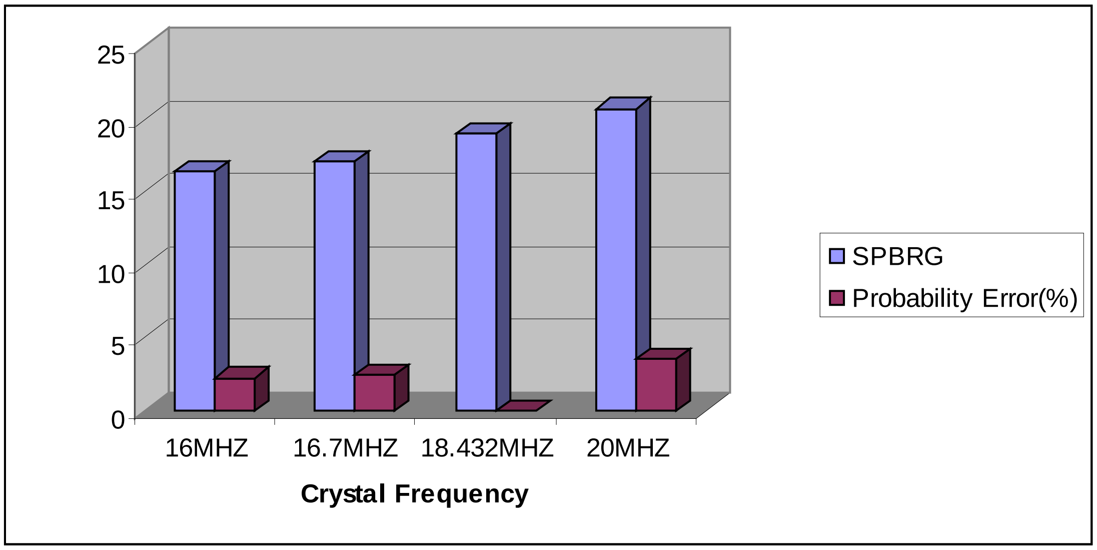
$$SPBRG = \frac{FOSC}{16 * BAUD} - 1 \qquad (4.1)$$

In the equation above, *FOSC* and *SPBRG* represent the crystal speed and the value which should be placed in the *SPBRG* register of the microcontroller to achieve a desired baud rate *BAUD*, respectively. The baud rate *BAUD* is set to a constant value of 57600 baud, which is the default baud rate of the ROK007 Bluetooth modules. The *SPBRG* should be an integer between zero and 255 [29]. The error probability P_e of USART is calculated by:

$$P_e = \frac{|SP_1 - SP_2|}{SP_1} * 100\% \qquad (4.2)$$

Where SP_2 and SP_1 represent the value of SPBRG register as calculated by (4.1) and the nearest integer value of SPBRG to SP_2 respectively.

The error probabilities of several different kinds of crystals often used for microcontrollers are compared, and results are shown in Figure 4.5



:

Figure 4.5 Relations between Crystal Frequency and SPBRG and PE

From the above diagram, it can be clearly seen that an 18.432MHz crystal produces the smallest USART transmission error. This is the reason for choosing 18.432MHz as the crystal frequency in this project.

2) Download Unit

The 16F877 supports in-circuit programming, allowing user code to be downloaded to the PIC via UART while it remains on the board. UART signals operate with a high/low voltage range of 0-5V (TLL/CMOS logic levels), while RS232 signals typically operate at high/low levels of $\pm 10V$. Hence, voltage level conversion is needed to communicate between each other. MAX232, line-driver IC, is deployed in this project to convert RS232 serial signals to UART serial format.

3) Sensing Unit

There are many sensing variables, such as position, velocity and acceleration used in a control system. In the first generation test-bed, a temperature sensor is employed in the construction of the prototype STIM. Temperature is relatively a straightforward metric measure which can be easily varied, estimated and quantified compared to other variables. Different kinds of sensor will be deployed in the future test-beds.

The DS 1620, capable of measuring ambient temperature to within ± 0.5 degrees Celsius [30], is chosen as the temperature sensor for this project based on its digital nature, low cost and high level of accuracy.

The DS1620 is a “one-wire device”, which transmits and receives all data over a single bidirectional pin. However, the PIC 16F877 does not support bi-directional I/O ports. To communicate with each other, a 74HC 126AP IC is employed to control a developed bus scheduling arrangement.

4) Display Unit

An LCD is deployed for the board for viewing and debugging program operation in real-time. The minimum power supply suggested by the LCD datasheet is 4.5V. Hence a +5V power supply is chosen for the board.

An 8-bit data port is required to send data to the LCD module according to the datasheet. There are three control lines for the LCD, including Read/^Write. Since in this work no read is performed on this display device, the Read/^Write control line is connected to the ground, which keeps LCD enabled to write.

5) MAX604 Voltage regulator

The main prototype source is approximately +5V, while ROK 101 007 requires a typical source and reference voltage of 3.3V (3.6 Max) [31]. This means that a voltage regulator is necessary to supply power to the ROK 101 007 module. After some investigation, MAX604 was identified as the most appropriate voltage regulator for the project. This device has an 8 pin DIP voltage regulator, with 3.3V regulation and a current supply of up to 500mA.

6) Design Results

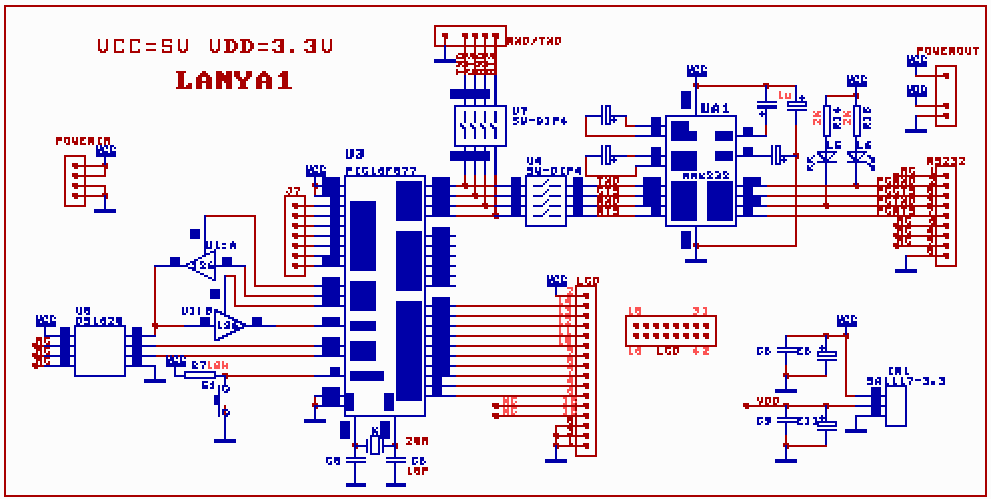


Figure 4.6 Circuit Diagram of Main Board

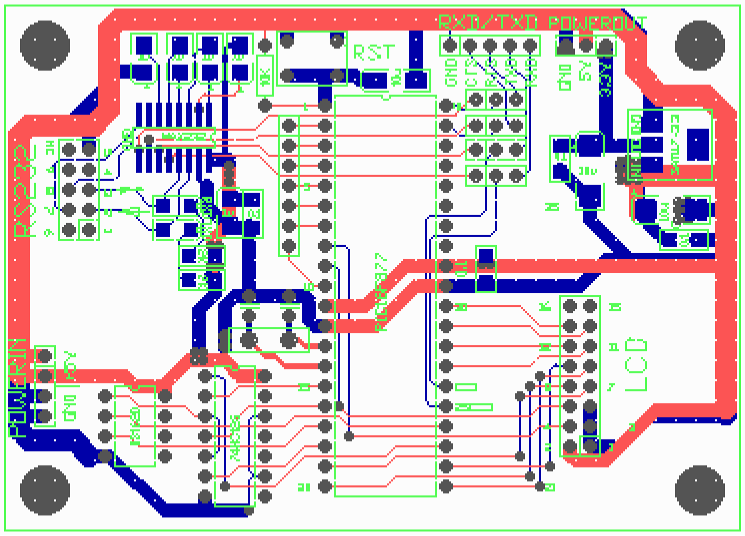


Figure 4.7 Printed Circuit Board (PCB) of the Main Board

The circuit diagram and printed circuits Board (PCB) of the main board developed for this project are shown in Figures 4.6 and Figure 4.7 respectively.

4.5.2 Communication Board

1) Ericsson ROK007 Bluetooth modules

Based on the research undertaken, ROK 101 007 module made by Ericsson was identified as the most suitable Bluetooth module for this project. The module is very compact with a dimension of 3.3 x 1.7 x 0.365 cm. The golden colored metal covering provides excellent RF shielding. It also has a 50Ω antenna interface for the best signal strength performance [32].

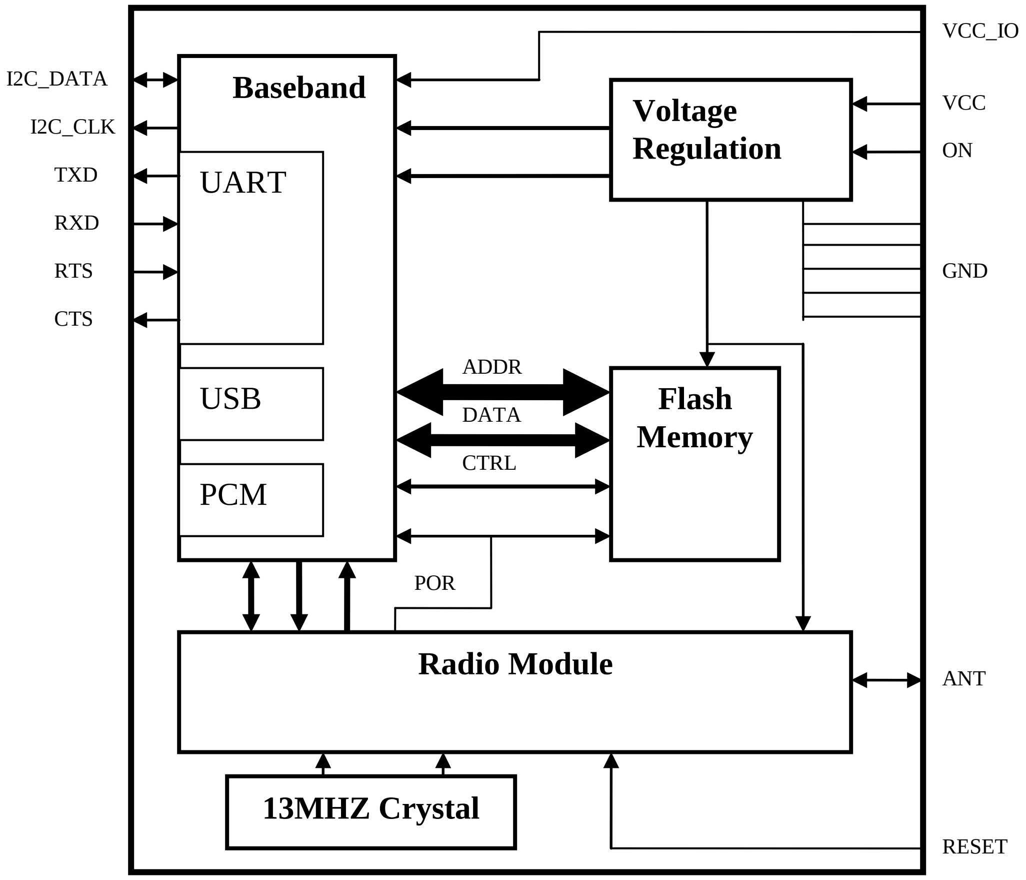


Figure 4.8 Schematic Diagram of Bluetooth Module

There are three major sections in the ROK 101 007 module: a baseband controller, a flash memory and a radio that operates at the globally available 2.4-2.5 GHz ISM band with spread spectrum technology. The baseband controller includes firmware for Host Controller Interface (HCI), which handles all communication with an external host such as a microcontroller. The external host and the baseband controller communicate with each other through a UART or a USB interface. The diagram of the Bluetooth module is illustrated in Figure 4.8 [32]:

2) Suyin Bluetooth carriers/sockets

Bluetooth modules, like any other small-scale microelectronics hardware, are sensitive to high temperatures [33]. It is also difficult to do any soldering on this module as the leads of the circuits are located beneath the module. In order to overcome such challenges, a socket system developed by Suyin [34] is used in this project. The socket itself is hardwired directly to the circuit board, and the Bluetooth module can be inserted and removed, via the release of a latch.

3) Rangestar 100902 Bluetooth antenna

It is important for Bluetooth model to choose an antenna. For ROK007 chipset, the $50\,\Omega$, 2.4GHz 100902 antenna unit suitable for Bluetooth design is used [35]. The antenna is mounted on the entire top of the PCB. Care should be taken to prevent any ground or power plane line to go underneath the antenna.

4) Results

The circuit diagram and printed circuits Board (PCB) of the communication board developed for this project are shown in Figure 4.9 and Figure 4.10 respectively. The goal of keeping the circuit small and compact has been achieved with the PCB being only slightly larger than the Bluetooth Unit.

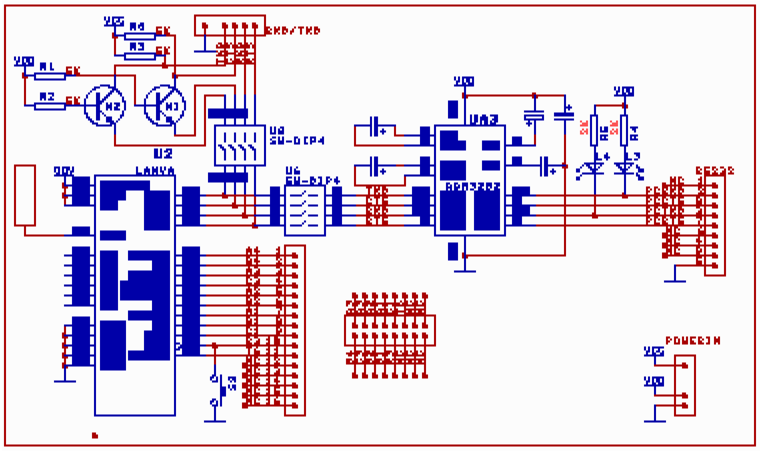


Figure 4.9 the Circuit diagram of Communication Board

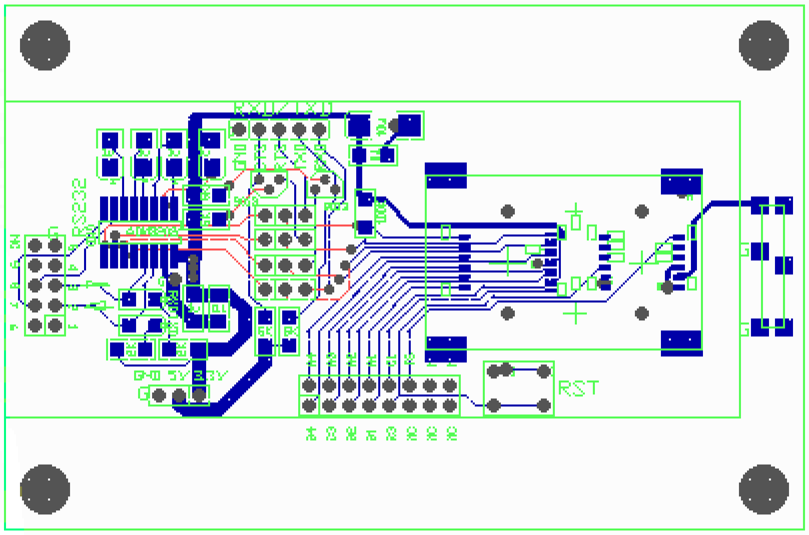


Figure4.10 Printed Circuits Board (PCB) of Communication Board

4.5.3 Connection between main board and transmission board

1) Communication function between Microcontroller and BT module

The communication of Bluetooth with other devices is implemented within the Bluetooth hardware/ firmware module as shown in Figure 4.11 [36]. In this system, the Bluetooth device is composed of two components including one external host (microcontroller) and a Bluetooth Module. The external host (microcontroller) and baseband controller in Bluetooth model communicate with each other through a UART or a USB interface.

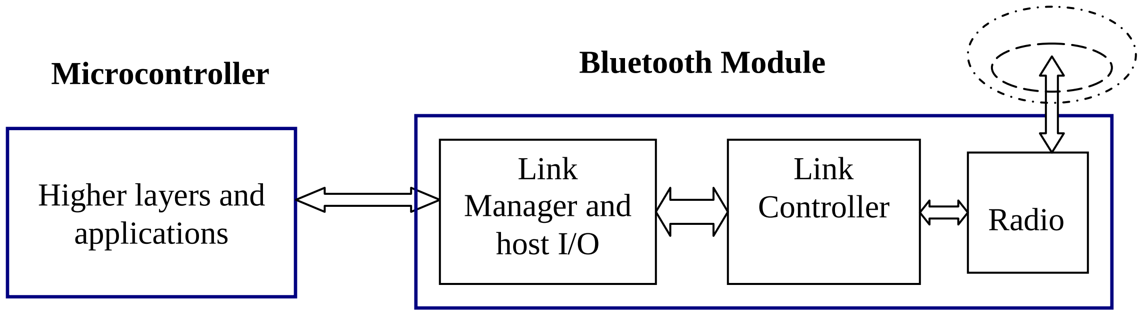


Figure 4.11 the Bluetooth Device

Microcontroller PIC 16F877 is deployed as an external host in this project. Since PIC 16F877 does not have any USB interface, the ubiquitous serial communication interface, UART is used to communicate between the host and ROK 101 007. The microcontroller's UART is programmed in this work to send data at the default baud rate for ROK 101 007, 57.6 kbps, which is not fast enough to reach the maximum baud rate of 460.8kbps.

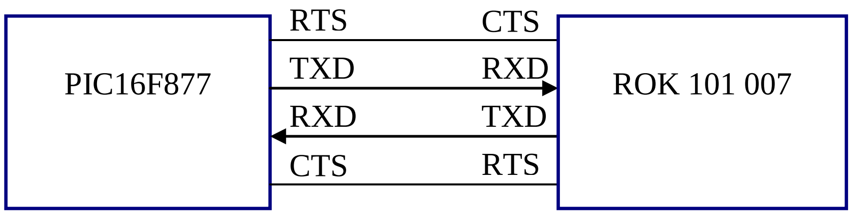


Figure 4.12 UART communication between the PIC16F877 and BT module

As illustrated in Figure 4.12 [36], the ROK 101 007 employs RTS/CTS flow control mechanism, in which two separate signal lines (RTS and CTS) between the microcontroller and the Bluetooth chip, control the flow of information and TxD and RxD pins provide the data flow. Microcontroller first raises the RTS control line to signal Bluetooth that it has some information for transmission. In response, the Bluetooth raises its CTS line to indicate that it is ready to transmit the data [36].

2) Voltage Level Conversion

There are four parameters defined for the logic levels in a digital logic family including VIL, VIH, VOL and VOH [36]:

- VIL represents the maximum voltage level, interpreted as a ‘0’ by a digital input.
- VIH represents the minimum voltage level, interpreted as a ‘1’ by a digital input.
- VOL represents the guaranteed maximum voltage level seen on a digital output as ‘0’.
- VOH represents the guaranteed minimum voltage level seen on a digital output z ‘1’.

A voltage difference between VIL and VIH produces an undefined logic state. Application of an undefined voltage results in excessive current to be drawn by CMOS inputs. In a functional circuit, the VIL of the input should be higher than the VOL of the output when a digital output is connected to a digital input. Similarly, the VIH of the input should be lower than the VOH of the output.

In the PCB design, discrete and passive voltage divider voltage level conversion methods are deployed as they are inexpensive and simple to use.

A) Discrete

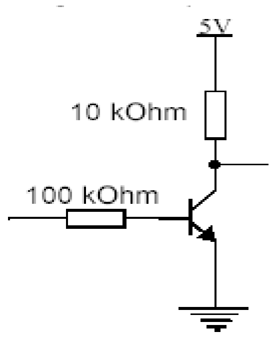


Figure 4.13 Discrete voltage level conversion

The circuit shown above converts a voltage swing of 0-3.3V to a voltage swing of 0-5V. The resistor values may have to be modified depending on the switching speed required [37].

B) Passive Voltage divider

The resistor voltage divider is used to divide down the 5V signal to the 3.3V, concerned with avoiding violation of the absolute maximum ratings of the Bluetooth Module [37]. With an appropriate choice of resistors, this will work fine, but it will draw static current all the time. This is illustrated in Figure 4.14.

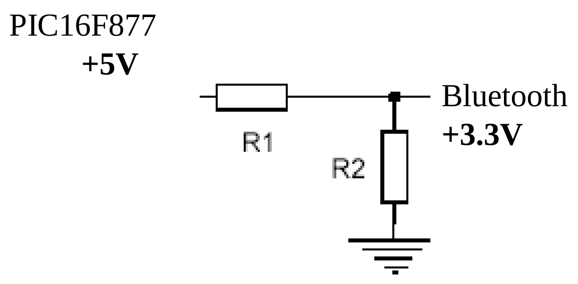


Figure 4.14 Passive Voltage Divider Voltage Level Conversion

The final version of STIM is shown in Figure 4.15.

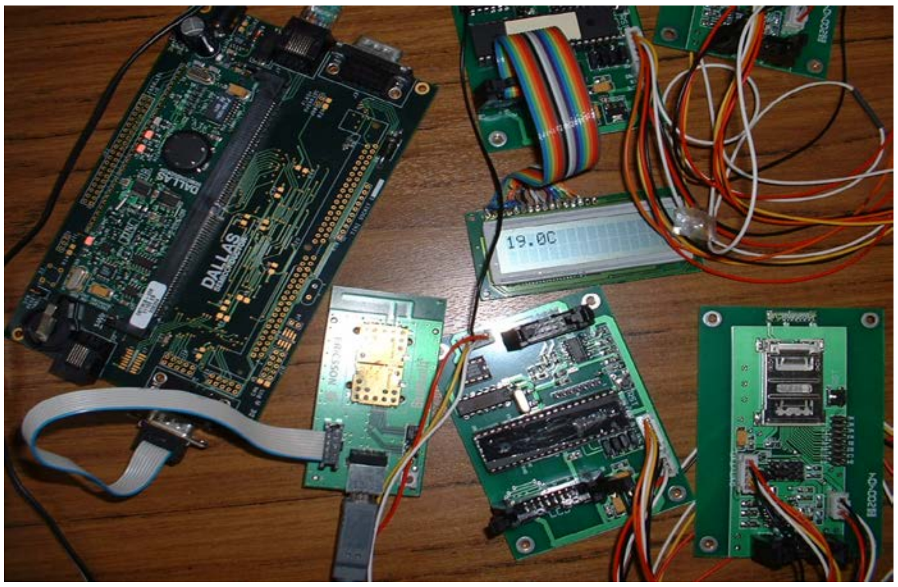


Figure 4.15 STIM Board

Chapter 5 Service Discovery

5.1 Introduction

Service discovery refers to the ability of applications, network devices, and services to advertise their capabilities to and to find other applications, network devices, and services to complete specified tasks [38]. Service discovery has been found essential for WACNets in implementing automatic discovery of networked devices and remote control of one device by another. In this work, a Service Discovery Protocol (SDP) is developed and validated for WACNet based on the first generation test-bed developed in this project.

This chapter provides a review of the existing Service Discovery Protocols (SDPs) and then describes the details of WANet SDP.

5.2 State of the Art

Typically, service discovery involves clients, lookup or directory servers and service providers. Service registration and lookup are very import components for most Service Discovery Protocols (SDPs). Currently, the most popular SDPs are SLP (Service Location Protocol), Jini, Salutation, UPnP (Universal Plug and Play), and Bluetooth SDP [39].

5.2.1 Service Location Protocol (SLP) [39]

SLP, developed by the IETF SvrLoc working group, aims to be a vendor-independent standard. Designed for TCP/IP networks, SLP is scalable up to large enterprise networks. The SLP architecture consists of three main components: User Agents (UA), Service Agents (SA) and Directory Agents (DA), which is not mandatory. SLP has therefore two operational modes, depending on whether a DA is present or not. If a DA exists on the network, it will collect all service information advertised by SAs. When a user needs a certain service, UAs will send their Service Requests to the DA and receive the desired service information. If there is no DA in the network, UAs repeatedly send out their service Request to the SLP multicast address (239.255.255.253), and a SA matching the service types requests will unicast a reply.

In SLP, the failure of the DA will lead to information loss and breakdown in service discovery. Furthermore, it adds an additional component to be administered. The implementer has to deploy a separate method in order to access a remote service, since SLP only provides its contact and location information.

5.2.2 Jini [39]

Jini, developed by Sun Microsystem, is an extension of the programming language Java. It mainly addresses two critical issues as follows:

- How do devices connect with each other in order to form a simple ad-hoc network?
- How do the devices provide services to others in the network?

A process called Discovery and Join is deployed by devices and applications in order to register with Jini network. Namely, an application or device places itself into the

Lookup Table on a lookup server, when it intends to join a Jini network. In Jini, the Lookup Table can store pointers to services and Java-based program code for these services. When a client requests a service, the Jini object code provides it a direct access to the service deploying an interface known to it.

The main drawback of Jini is that it depends on the programming environment and also consumes a great deal of memory and processing power to run a JVM. This can be difficult for some embedded systems.

5.2.3 Salutation [39]

The Salutation architecture is developed by the Salutation Consortium. Service discovery in Salutation is defined at a higher layer, and the transport layer is not specified. Salutation is also independent of the network technology and the programming language, and may run over multiple infrastructures.

The Salutation architecture consists of Salutation Managers (SLMs), serving as service brokers, where services register their capabilities. When one client needs a service, it sends a request message to the SLM. When discovering a desired service, the client is capable of requesting the SLM to utilize the service.

5.2.4 Universal Plug and Play (UPnP) [39]

Universal Plug and Play (UPnP) is developed by an industry consortium, founded and lead by Microsoft. Its usage is proposed for home or small office computer networks, where different devices are able to automatically discover and utilize each other.

There are three major components in UPnP: control device, device and services. When entering a network, devices broadcast a short message containing its type, unique identifier, and a pointer to more information. The control device searches by means of multicast for interesting devices whose services it requires to use.

The main drawback of UPnP is that it depends on a specific network technology (TCP/UDP and IP). Also, an UPnP entity needs heavy resources to allow support for GENA and SOAP web servers, and XML parsing [39]. Its purely peer-to-peer architecture and has potential to increase the network traffic due to extensive use of multicast messaging.

5.2.5 Bluetooth Service Discovery Protocol [40]

The Bluetooth protocol stack contains the Bluetooth Service Discovery Protocol (Bluetooth SDP). Based on the Piano platform by Motorola, Bluetooth SDP has been modified to suit the dynamic nature of an ad-hoc network. SDP is used to locate services provided by or available via a Bluetooth device.

The SDP server of Bluetooth Service Discovery Protocol (Bluetooth SDP) runs on any device which can provide such service. A set of service records is maintained on the server. There is a service record available for each service. A Bluetooth device intending to use a service is called an SDP client. During a service discovery, a request message including the list of the service devices requires is sent by SDP client. The SDP server checks for a match and responds by a service handle providing the attributes of the services found. The Bluetooth service discovery process is illustrated in Figure 5.1 [40].

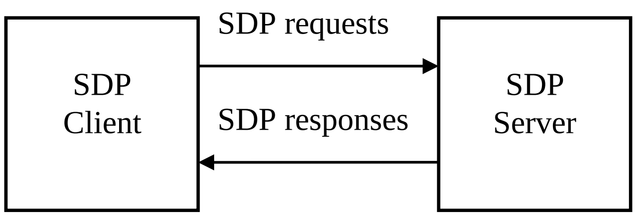


Figure 5.1 SDP Client-Server Interactions

These Service Discovery Protocols like SLP, Jini and UPnP are designed with traditional wired network, which have quite different characteristics from WACNet. The Bluetooth Service Discovery Protocol is based on connecting to a specific node and querying about its available services. When the network becomes larger, the protocol of querying every single node in the network about the possibly unavailable service wastes both time and resources. A protocol to achieve service discovery should therefore be developed for WACNet.

Service Definition and Service Discovery Protocol for WACNet are described in the following sections.

5.3 Service Definitions of WACNet Nodes

IEEE 1451.2 standard defines a smart transducer interface model (STIM), a transducer electronic data sheet (TEDS), and a digital interface to access the data [41]. The schematic structure of the STIM model in IEEE 1451.2 standard is illustrated in Figure 5.2 [35].

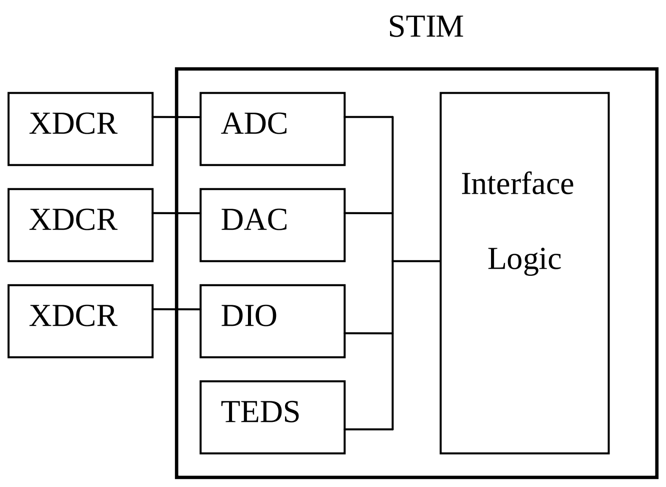


Figure 5.2 Schematic Structure of STIM model

In the proposed system, TEDS uses an object-oriented approach to describe the properties and service abilities of the total STIM unit and transducer channels it controls. The parameters of TEDS have been redefined to meet the requirement of the WACNets and they are stored in a non-volatile memory with each STIM.

In each STIM node, TEDS consists of one Meta-TEDS that includes common information for all the transducers and several channel-TEDSs with information about each transducer. The parameters of Meta TEDS and Channel-TEDS are provided in Tables 5.1 and 5.2 respectively.

Table 5.1 Parameter Definition of Meta-TEDS

Meta TEDS			
Field #	Description	Field Length	Field Type
1	Meta TEDS Length	1 Byte	U8
2	STIM Name	4 Byte	C32
3	STIM ID	6 Byte	U48
4	Maximum Data Rate	6 Byte	U48
5	Number of Implemented Channels	1 Byte	U8

The parameters of these tables are defined as follows:

- STIM Name: The name of the STIM as defined by the manufacturer.
- STIM ID: Identification field associated with the STIM, whose value is globally unique. In the proposed system, the MAC address of Bluetooth model attached to the STIM can be employed as the STIM ID.
- Maximum Data Rate: Describes the maximum data rate, which is supported by the STIM interface, in bits per second.
- Number of Implemented Channels: The number of transducers attached to the STIM, with a maximum of 255. A Channel number of ‘0’ is called CHANNEL_ZERO and addresses all the transducer channels within a STIM.

Table 5.2 Parameter definition of Channel-TEDS

Channel TEDS			
Field #	Description	Field Length	Field Type
1	Channel TEDS Length	1 Byte	U8
2	Transducer Name	8 Byte	C64
3	Device Manufacturer	8 Byte	C64
4	Transducer Status	1 Byte	U8
5	Transducer Priority	1 Byte	U8
6	Group ID	1 Byte	U8
7	Channel Type Key	1 Byte	U8
8	Technical Parameters	*N* Bytes	-

- Transducer Name: The name of the transducer defined by the manufacturer.

- Device Manufacturer: Text string identifying the manufacturer of the transducer unit.
- Transducer Status (TS): Transducer can be read (TS =1) or written (TS=2) or both (TS=3).
- Transducer Priority: The value of priority is decided by the real-time requirement of the transducer. For example, the real time constraint on a light sensor is tighter than temperature sensors.
- Group ID: STIM nodes with the channels of the same Group ID can work together to implement one particular task.
- Channel Type Key: It specifies the channel transducer type as illustrated in Figure 5.3. 00H is defined as general sensor and 80H as general actuator.

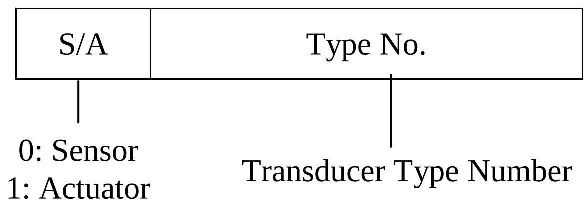


Figure 5.3 the Definition of Channel Type Key

- Technical Parameters: The number of parameters and field length *N* bytes vary with the transducer type. The technical parameters for the temperature sensor are shown in Table 5.3.

Table 5.3 Definition of technical parameter for a temperature sensor

Field #	Description	Field Length	Field Type
8	Technical Parameters	11 Byte	-
Measuring Range			
	Lower Limit	4 Byte	F32
	Upper Limit	4 Byte	F32
Measuring Accuracy			
	Accuracy	3 Byte	F24

Field type is defined in Table 5.4, which consists of all field types and their length description.

Table 5.4 Definition of Field Type

Field Type	Field Description
C32, C48, C64	String
U8	Unsigned Int
F24	1 Byte for integer, 2 byte for decimals
F32	2 bytes for integer, 2 bytes for decimals

5.4 The Architecture of WACNet Service Discovery

The architecture of WACNet service discovery is a hybrid of peer-to-peer and client-server architecture. In WACNet, master nodes are considered as server-like devices that reduce communication cost as service advertisement and search messages are addressed to master nodes instead of broadcast to the entire WACNet.

In the proposed system, master nodes and bridge nodes are aware of their piconet members. The slave node first sends a request to the master node to discover service in the same piconet. If the requested service is available in the piconet, the master node will send back a response to the slave node. Otherwise, the master node will forward the inquiry to the next bridge node.

Figure 5.4 [40] illustrates the implementation of service discovery between a slave and a master in the WACNet. In the proposed system, there are service record database for each master node to store the service information produced by the slave nodes provide in the same piconet.

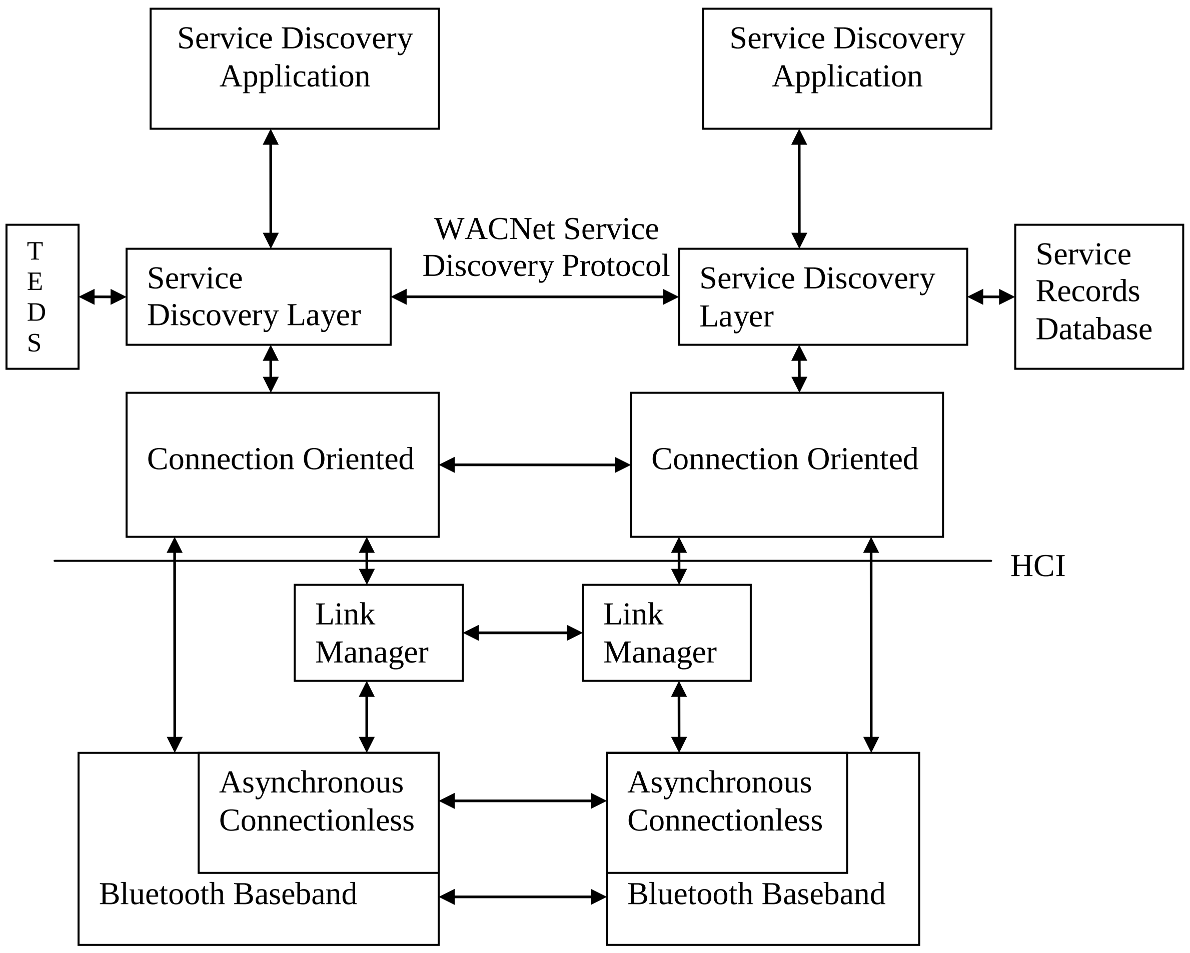


Figure 5.4 Service Discovery between one master and one slave

A message from a slave is sent to the master node via the physical layer. This will require the transfer of the message from the higher layers to the physical layer in the salve and vice-versa in the master, as illustrated in Figure 5.5. The “lower layers” referred to in this diagram are the ones described in Figure 5.4.

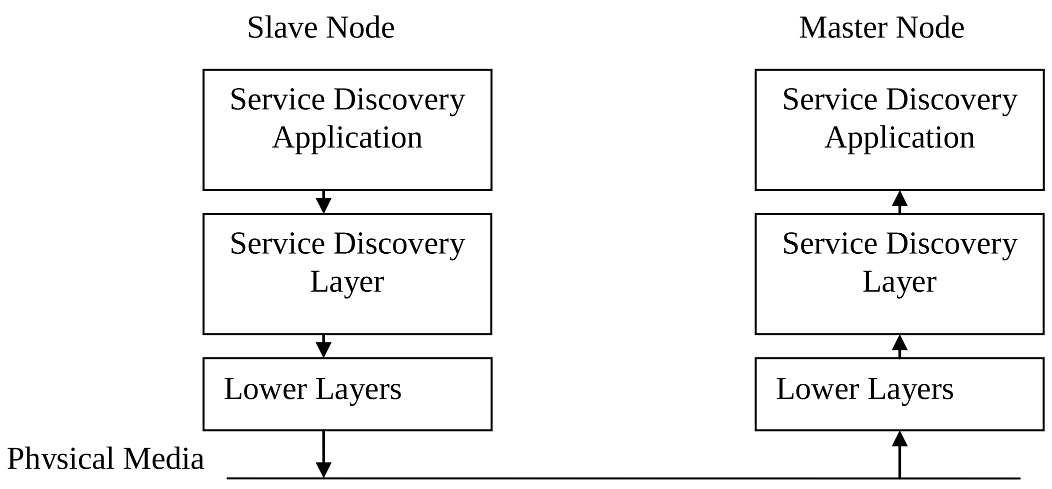


Figure 5.5 Protocol Model of WACNets

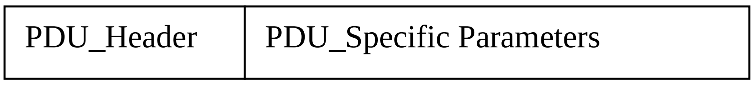
5.5 Implementation of WACNet SDP

The service discovery is implemented through a number of service operations as described below. This section first introduces the data unit format deployed in WACNet SDP, and then describes the implementation of the WACNet SDP.

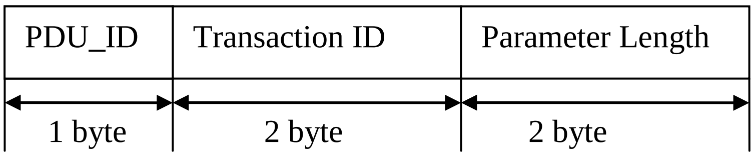
5.5.1 WACNet SDP Data Unit Format

Each Service Discovery Protocol Data Unit (SDP_PDU), generated by the service discovery layer consists of a PDU header followed by PDU_specific parameters. The header contains two fields: a PDU_ID and a Parameter Length, as illustrated in figure 5.6. PDU_specific parameter varies according to PDU_ID, and will be described in the corresponding sections.

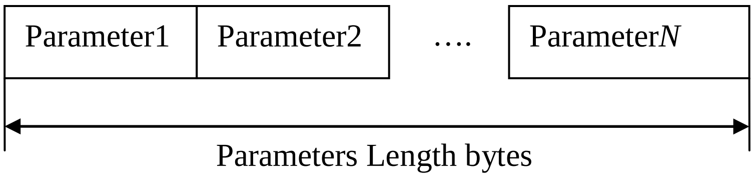
SDP_PDU:



PDU Header:



PDU_Specific Parameters:



PDU_ID: field identifies the type of PDU, i.e. its meaning and the specific parameters.

PDU_ID: Size: 1 Byte

Value	Description
N	The PDU_ID field identifies the type of PDU.
0X01	SDP_JoinClusterRequest
0X02	SDP_ServiceRegisterRequest
0X03	SDP_ServiceRegisterResponse
0X04	SDP_ServiceSearchRequest
0X05	SDP_SearchSearchReply
0X06	SDP_ServiceUseRequest
0X07	SDP_ServiceUseReply
0X08	SDP_ReadTEDSRequest
0X09	SDP_ReadTEDSRespond
0X10	SDP_WriteTEDS

Transaction ID: Size: 2 Bytes

Value	Parameter Description
N	Transaction ID field uniquely identifies request PDUs and is used to match response PDUs to request PDUs.

Parameter Length: Size: 2 Bytes

Value	Parameter Description
N	The parameter length field specifies the length (in byte) of all parameters contained in the PDU.

Figure 5.6 SDP_PDU Data Unite Format

5.5.2 WACNet SDP

The WACNet SDP includes two important parts: service registration and service search, which are described in the following subsections.

A. Service Registration

Service registration is an important component of WACNets SDP. It enables the nodes or users to look for a node with a particular service, and for the WACNet to offer plug and play capability for minimum data transmission. This subsection specifically describes the implementation of service registration of WACNets, which is designed based on the service definitions of the nodes provided in section 5.3.

Initially, each node in the WACNet decides whether to become a master and each node has a probability p ($0<p<1$) of becoming a master. Therefore, the possibility of n masters in the network with N nodes is determined by equation 5.1.

$$P_p(n \mid N) = \frac{N!}{n!(N-n)!} p^n (1-p)^{N-n} \qquad (5.1)$$

The probability P_p reaches its maximum when $\frac{dP_p}{dp} = 0$, namely, which results in $p = \frac{n}{N}$.

In a WACNet with N nodes, n is chosen as $\frac{N}{8}$ to minimize the number of the piconets (In each piconet, there is one master and up to 7 active slave nodes.). Hence, the probability p of each node to become a master is $\frac{1}{8}$. This decision is made by each node choosing a random integral number between 0 and 7. If the number is 0, the node becomes a master, otherwise, a slave node.

A node which decides to be a slave carries out an Inquiry process to find the nearby master nodes. It then tries to connect to one of the found nodes in the identified order, by sending a ***Create_Connection*** packet to the node. If the node eventually responds with a ***Connection_Complete*** Event packet (with the status indicating successful establishment of connection.), the connection is implemented.

As defined in Section 3.3, only neighboring nodes with common or complimentary functions can form clusters of nodes. Hence, the slave node sends a ***JoinClusterRequest*** message with all Group IDs it has in the Channel_TEDSs to the connected master, requesting permission to join the cluster. The master receiving this message will compare the received Group IDs with all the Group IDs it owns. If there is no equal Group ID, the master will disconnect this STIM node. Following that, the STIM node will connect to the next neighboring master, and will send another ***JoinClusterRequest*** message to it. The process continues until it connects to one of the masters with the same Group ID. Following that, the master will reply a ***ServiceRegistrationRequest*** message to the slave node, which then sends a ***serviceRequestReply*** message containing the service information the slave can provide back to the master. If the slave node cannot find a master with the same Group ID, it will finally become a master node. The whole process, as illustrated in Figure 5.7, assists in the implementation of the Plug and Play feature.

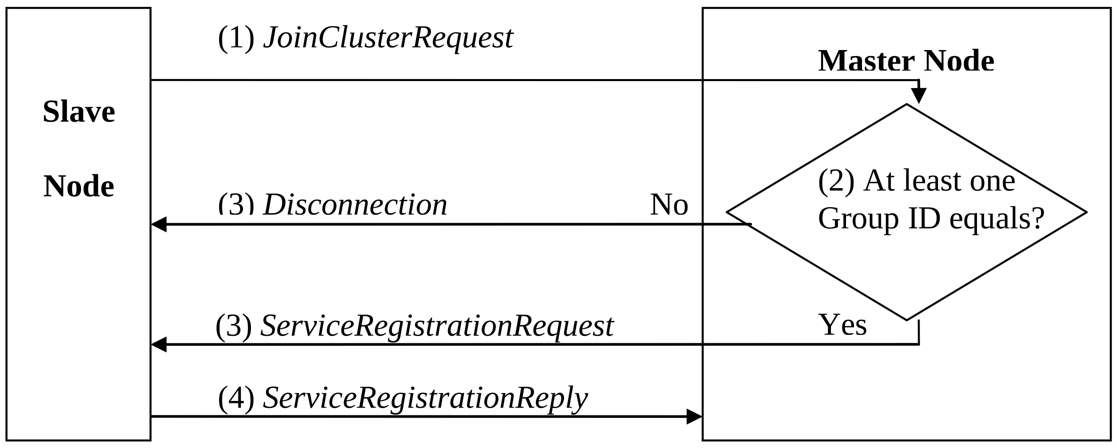

Slave Node
Master Node
(1) JoinClusterRequest
(2) At least one Group ID equals?
No
(3) Disconnection
Yes
(3) ServiceRegistrationRequest
(4) ServiceRegistrationReply

Figure 5.7 the Service Operations to Join a Cluster

The numbers in the brackets shown in the diagram above denote the order in which the packet is transferred. The PDU specific parameters for the SDP_PDUs of Figure 5.7 are defined in Table 5.5.

Table 5.5 the definition of the PDU specific parameters for the messages in the figure 5.7

PDU_ID Description	PDU specific parameters
SDP_JoinClusterRequest	Connection_Handle, Group IDs
SDP_ServiceRegistrationRequest	Connection_Handle
SDP_ServiceRegistrationReply	Connection Handle, Number of Implemented Channel, Each channel characteristics (Channel No., Transducer Priority, Group ID, Channel Type Key)

After the cluster formation stage, each master node has a table (Table 5.6) which lists the connection information of all the slave nodes in the cluster.

Table 5.6 Connection Information of all the slave nodes in the cluster

MAC Address	Piconet ID	Connection Handle	Role of the Slave
6 Bytes	3 Bits	12 Bits	2 bits

The parameters of this table are defined as follows:

- MAC Address: This table lists the MAC (medium access control sub-layer) address of all of the slave nodes in the cluster.
- Piconet ID: When a slave node connects to the master, the master will provide it with a Piconet which is stored in the table. The first connecting device is allotted a piconet ID of 1, the second one is allotted a piconet ID of 2, and so on.
- Connection Handle: Each node will also have a corresponding Bluetooth Connection Handle, which is used for Bluetooth communication between the master and the slave.
- Role of the Slave Node: The role of slave node is assigned by the master as
  0: Normal slave; 1: Backup slave; and 2: Bridge slave. Bridge slave is deployed in connection with different piconets and holds the master's MAC addresses of these piconets.

After the process of service registration, every master node forms a service record for all slaves in the cluster, as illustrated in Table 5.7. The bridge nodes acting as a relay between piconets also are supplied with this service record table, so the efficiency of inter-piconet communication can be significantly improved.

Table 5.7 Service Record for one slave node in the cluster

Piconet ID	Number of Implemented Channels(*N*)	characteristics of Channel 1	…	characteristics of Channel *N*
3 Bits	1 Byte	4 Bytes	…	4 Bytes

- Piconet ID: Please refer to Table 5.6.
- Number of Implemented Channels: Please refer to Table 5.1.
- Each Channel Characteristics: it describes the main characteristics of each channel, as defined in Table 5.8, with the parameters provided in Table 5.2.

Table 5.8 Definition of Each Channel Characteristics

Characteristics of Channel			
Channel No.	Transducer Priority	Group ID	Channel Type Key
1 Byte	1 Byte	1 Byte	1 Byte

In each cluster, the service record in the master node will also be backed up in the backup node. The master will then send updates to the Backup devices whenever a change occurs in the service record. This process is illustrated in Figure 5.8. The numbers in the brackets shown in the figure denote the order in which the packet is transferred.

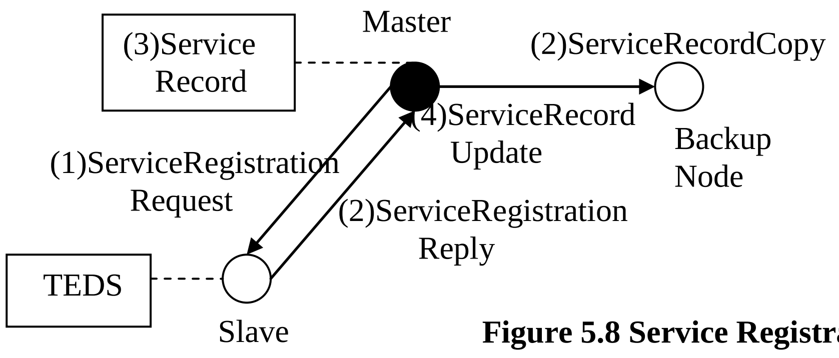


Figure 5.8 Service Registration

In the WACNets, all the clusters with common or complimentary functions are considered as one group, in which all the nodes have at least the same group ID in the Channel_TEDSs, which can represent the whole group. Each group connects with one NCAP node.

In WACNets, after a cluster is formatted, the master node then sends the connected NCAP node a ***ServiceRegistrationPermissionRequest*** message to require a permission of the NCAP node to register the services of all the nodes in the cluster into it. If NCAP finds the information of this cluster in its service table, it will send a ***ServiceRegistrationPermissionDeny*** message back to the master. Otherwise, the NCAP node will reply with a ***ServiceRegistrationRequest*** message to the master. After the master has received this message, it will send a ***ServiceRegistrationReplyforNCAP*** to the master node. The whole process is illustrated in Figure 5.9.

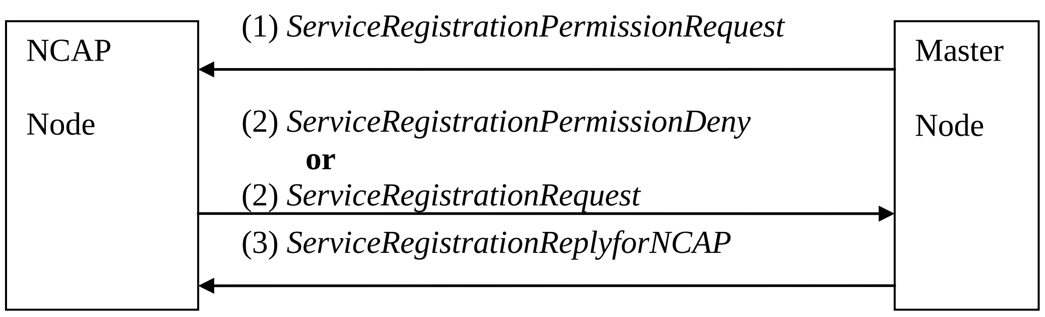

NCAP Node
Master Node
(1) ServiceRegistrationPermissionRequest
(2) ServiceRegistrationPermissionDeny
or
(2) ServiceRegistrationRequest
(3) ServiceRegistrationReplyforNCAP

Figure 5.9 Service Registration between Master nodes and NCAP nodes

The PDU specific parameters for the SDP_PDUs of Figure 5.9 are defined in Table 5.9.

Table 5.9 definition of the PDU specific parameters for the messages of Figure 5.9

PDU_ID Description	PDU specific parameters
SDP_ServiceRegistrationPermissionRequest	Connection_Handle, MAC_Address
SDP_ ServiceRegistrationRequest	Connection_Handle
SDP_ ServiceRegistrationReplyforNCAP	Connection Handle, Transducer Priorities and Channel Type Keys

Each NCAP node has a service record, which lists the characteristics of all the clusters connected with this NCAP node. In the service record, each cluster is described as in the Table 5.10.

Table 5.10 Description of Each Cluster in NCAP

Description of a Cluster		
Field #	Description	Field Length
General Description of Each Cluster		
1	Cluster No.	5 bits
2	The Number of the Slaves	3 bits
Service Abilities for the Cluster		
3	Channel Type Key 1	1 Byte
	Transducer Priority 1	1 Byte
	….	
	Channel Type Key N	1 Byte
	Transducer Priority N	1 Byte

B. Service Search

When a user looks for a node with particular service ability, the monitoring station will send a ***ServiceQuest*** message with Channel Type Key to the corresponding NCAP node, and the NCAP will check its service record table. If a match occurs, the NCAP node will send a ***ServiceSearchRequest*** message to the corresponding master node first searched. The master node will send a ***ReadTEDSRequest*** to the corresponding slave node. After receiving this message, the slave node replies with a ***ReadTEDSReply*** message to the master node. On receipt of this message, the master node sends a ***ServiceSearchReply*** message to NCAP nodes. This is eventually sent to the Monitor, via which, the user will get the required information. The process is shown in the Figure 5.10.

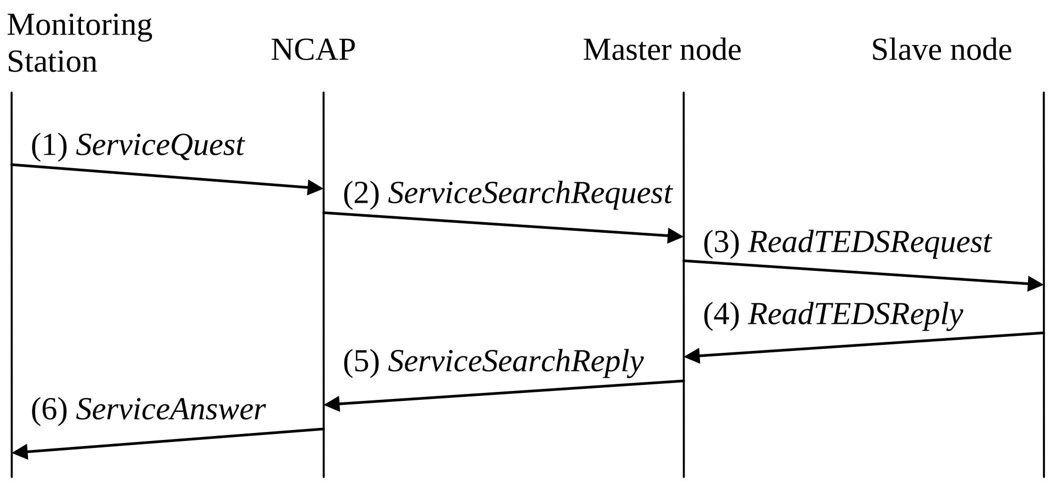


Figure 5.10 Service Search from a User

The PDU specific parameters for the SDP_PDUs in the Figure 5.10 are defined in Table 5.11.

To discover a node with particular service ability, the slave node will first send a ***ServiceRequest*** message to the corresponding master node, and the master node will check its service record table. If a match occurs, the master node will send a ***ServiceReply*** message back to the slave node. If the requested service is not available in the piconet, this ***ServiceRequest*** message will be forwarded to the bridge node in the same piconet to check whether the requested service is available in the neighboring piconet (within the scatternet) or not.

Table 5.11 PDU specific parameters for the messages in Figure 5.10

PDU_ID Application	PDU specific parameters
SDP_ServiceSearchRequest	Connection_Handle, MAC_Address, Channel Type Key
SDP_SearchSearchReply	Connection_Handle, piconet ID,
SDP_ReadTEDSRequest	Connection_Handle, piconet ID,
SDP_ReadTEDSRespond	Connection_Handle, Cluster ID, piconet ID, The Techanical Parameters of

In the proposed system, the bridge node can respond with information about services of all the piconets connected to it, without questioning the master. This would significantly improve the efficiency of the inter-piconet communication in the network.

Chapter 6 Software Design of WACNets

6.1 Introduction

During the project, many important software libraries have been developed to drive the hardware components of WACNet testbed and to verify the feasibility of the WACNet Model and its service discovery protocol. This chapter outlines the general software structure, and then specifically describes the related software design and its functionality of each part respectively.

6.2 Overall Software Structure

In the proposed system, the whole software libraries can be grouped into three categories in terms of the design of the hardware platform:

(a) Software running on the Monitor

(b) Software running on the NCAP

(c) Software running on the STIM

The overall software stack implementation for WACNet is illustrated in figure 6.1, which visually describes how nodes interrelate. Monitor can communicate with a NCAP via Ethernet physical layer and STIM can communicate with other STIM nodes or NCAP nodes via Bluetooth Module. As illustrated in the diagram, each category consists of various modules, which will be described in the following sections.

6.3 Monitor Software

In the WACNet, users can monitor and control the network via the management console executed on the Monitor. The management console is an applet embedded web page, and its main function is to provide a graphical monitoring and control interface to the WACNet. In the proposed system, the software programmed for Monitor can be separated into two parts:

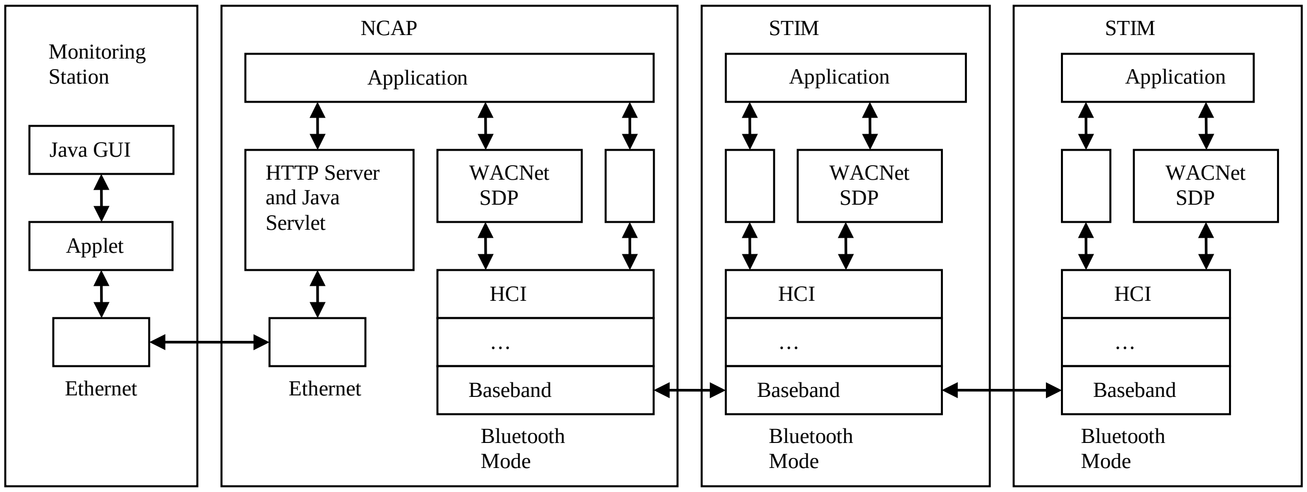


Figure 6.1 Overall Software Stacks of WACNet

(1) HTML software

(2) An applet, which is embedded into the HTML software.

6.3.1 The Design Participle of Monitor Software

Monitor performs network control and management via the NCAP module. The complimentary relationship between applets on the Monitor and servlet running on the

NCAP forms the basis of the network communications between the Monitor and NCAP, which is illustrated in the figure 6.2.

In order to send command or data to a servlet process, the applet must ‘call’ the servlet on the NCAP by the following HTTP request:

GET /servlet/BluetoothServlet?function=parameter_value HTTP/1.0

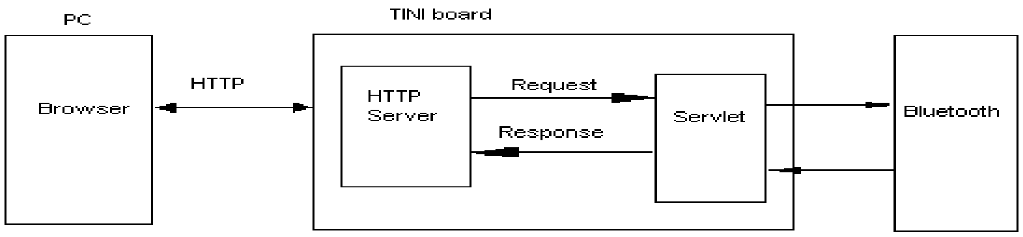


Figure 6.2 Request and Response of Servlets Applets and Bluetooth Module

It will pass the function it wishes to execute as a parameter, whose value varies with the command or the data sent to the servlet.

The process of sending data from the TINI to the applet on the Monitor is more straightforward. Simple, writable data streams are established as the sole method used to transfer asynchronous data from NCAP to the Monitor.

The following section describes the implementation of Monitor software, following the design principles mentioned above.

6.3.2 The Implementation of Monitor Software

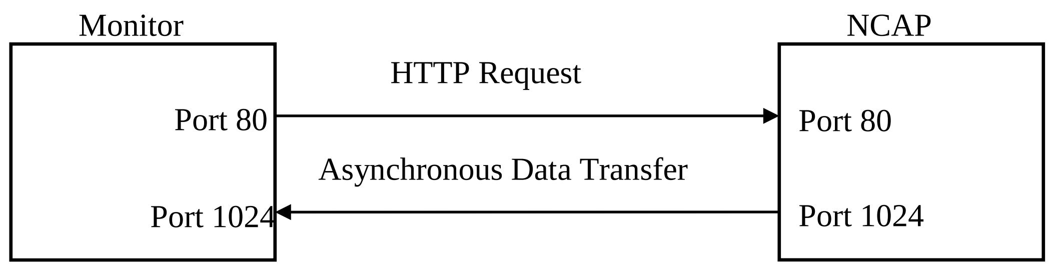


Figure 6.3 Communication Channel between the management console and NCAP

In the proposed system, a bi-directional communication channel between the management console and NCAP is established, which is illustrated in figure 6.3. As illustrated in the diagram, port 80, which is reserved for common process HTTP, and an unused port 1024 are deployed to send HTTP request to the NCAP and transfer asynchronous data from the NCAP to the Monitor respectively.

When the applet program is started, the server program on the NCAP has already been running and listening to the port 80, waiting for a client request. When it first appears on the screen, the applet sends a “Start” HTTP request to servlet on the NCAP via port 80. The applet then opens another socket, which is connected to the server running on the NCAP, to receive data directly from the servlet.

The applet and servlet processes can exchange data via the communication channel provided by this socket-to-socket connection.

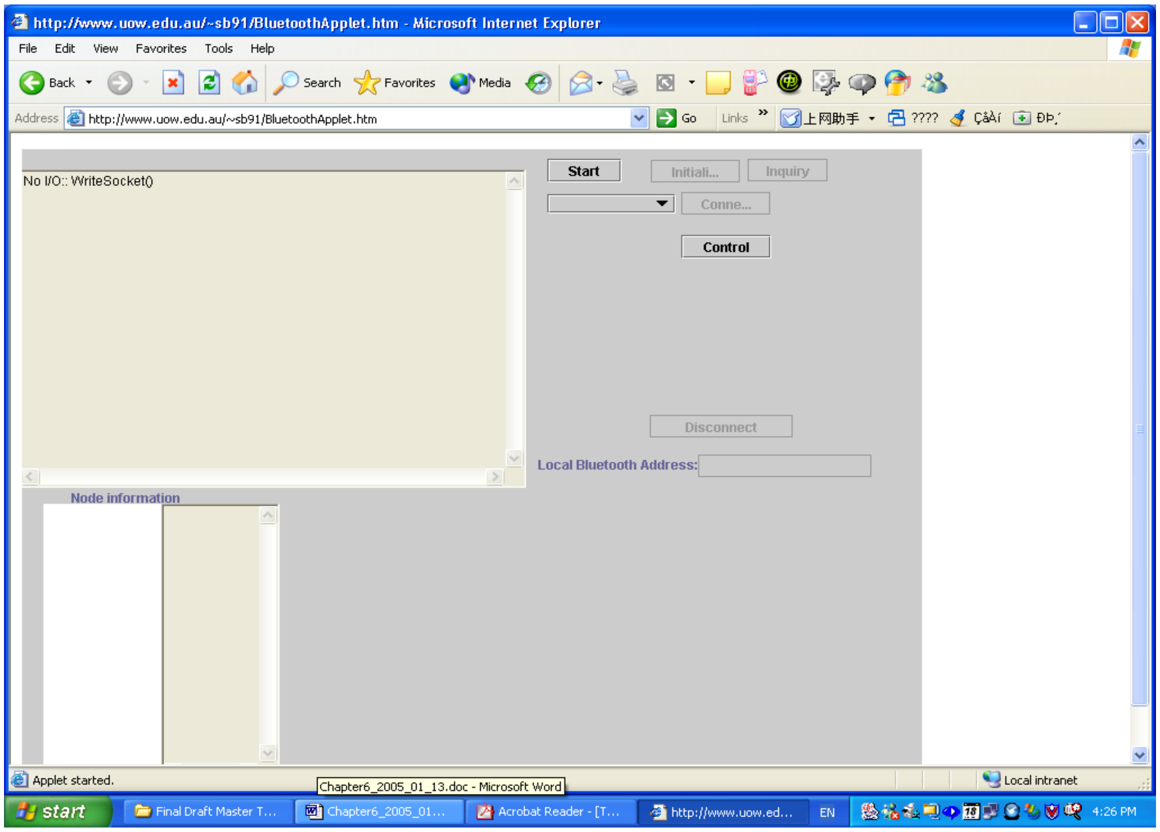
If a user wants to operate WACNet, he presses the corresponding button on the applet. When a function button on the management console is pressed, a corresponding HTTP request is sent to the servlet. The HTTP request is then decoded on the servlet side based on the parameter value, and appropriate actions are taken.

When an applet receives data from servlet on the NCAP, the raw data is passed to determine the type of the received data through a decoding process, and then directed towards the proper display field on the applet.

6.3.3 The Screen-shot of the Applet for WACNet

The applet coded and designed in Java is user friendly, and easy to operate. The applet describes the general functions of nodes as provided by WACNet, and offers users with further inquiry tools. A screen-shot of the applet can be seen in figure 6.4, which contains the following elements:

- Command Log: Used to show the commands previously sent and their status.
- Control Buttons: Commands are sent to either the NCAP or a STIM (via the NCAP) by pressing these control buttons.
- Information fields: Information fields show the requested information of the WACNets.


http://www.uow.edu.au/~sb91/BluetoothApplet.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.uow.edu.au/~sb91/BluetoothApplet.htm Go Links
Start
Initiali...
Inquiry
Conne...
Control
Disconnect
Local Bluetooth Address:
Node information
Applet started.
Local intranet
start
Final Draft Master T...
Chapter6_2005_01...
Acrobat Reader - [T...
http://www.uow.ed...
EN
4:26 PM
Chapter6_2005_01_13.doc - Microsoft Word

```
No I/O:: WriteSocket()
```

Figure 6.4 the Applet Screen-shot

6.4 NCAP software

The NCAP software consists of three major sections: HTTP server, Bluetooth servlet and Bluetooth software, which will be described in this section.

6.4.1 HTTP Server software

A HTTP web server installed on the TINI provides the prototype NCAP the required functions. A web server program called the TINI HTTP Server developed by Smart Software Consulting [36] is used. When the server is started, it listens to TCP port 80 for content requests from the Monitor. Upon request from the station, the Bluetooth Servlet is called.

6.4.2 Bluetooth Servlet Software

At first, the servlet waits for HTTP request. When it receives a “Start” HTTP request from the applet, the servlet will open an unused TCP port 1024 on the TINI. Following that, upon the receipts of a HTTP request, appropriate actions are taken.

6.4.3 Bluetooth Software

The Bluetooth device communicates through the Host Controller Interface (HCI) protocol, as illustrated in Figure 6.5 [36].

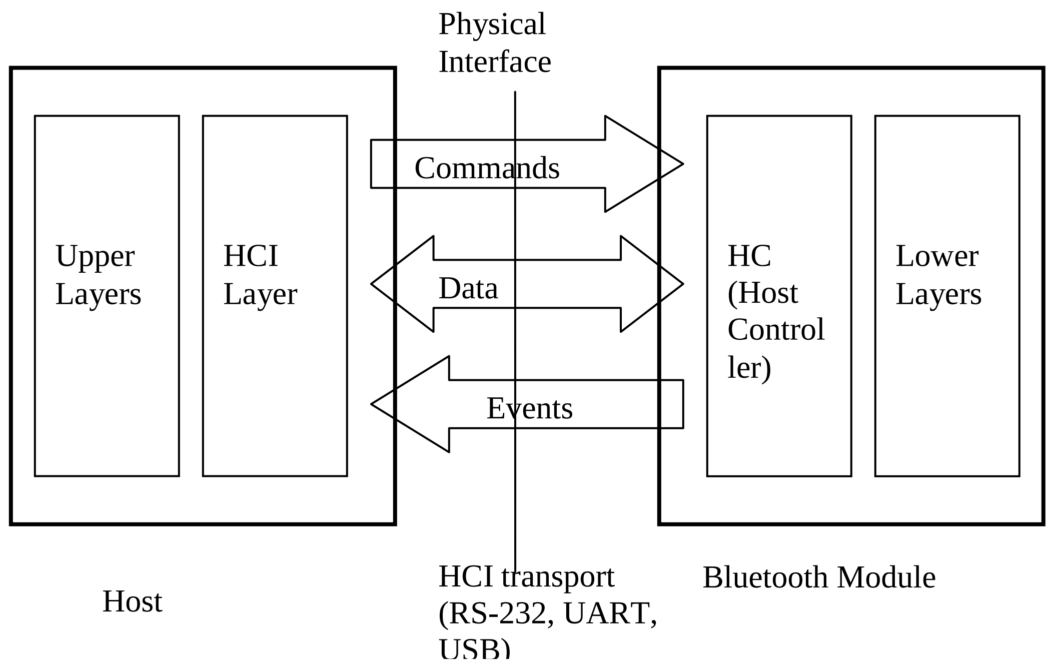


Figure 6.5 Communications between Host and Bluetooth Module

Three classes of HCI_PDUs, which are deployed to exchange information between the host controller in the Bluetooth module and the HCI layer in the host, are command HCI_PDUs, event HCI_PDUs and data HCI_PDUs. In this diagram, commands denote the class of command HCI_PDUs, which carries control and management information sent from the HCI layer to the host controller. The class of event HCI_PDUs carries management and control information from the host controller to the HCI layer. Finally, the class of data HCI_PDUs carries fragments of L2CAP_PDUs Asynchronous Connectionless Link (ACL) data. Each packet format is defined in the Bluetooth specifications PDF file [43].

In the NCAP, the Bluetooth software has seven modules:

- SDP(SDP.java) – implements the service discovery functions;
- Serial Module (SerialEvent.java) – provides serial routines for communication between the microchip and Bluetooth Module;
- HCI Module (HCI.java) – performs various HCI functions for Bluetooth Module;
- Bluetooth List Module (FoundBts.java) – contains a list of Bluetooth devices found an inquiry process;
- Bluetooth Device Module (bts.java) – contains specific information about the Bluetooth devices themselves; and
- Application Module (bt.java) - This is the main program loop for NCAP.

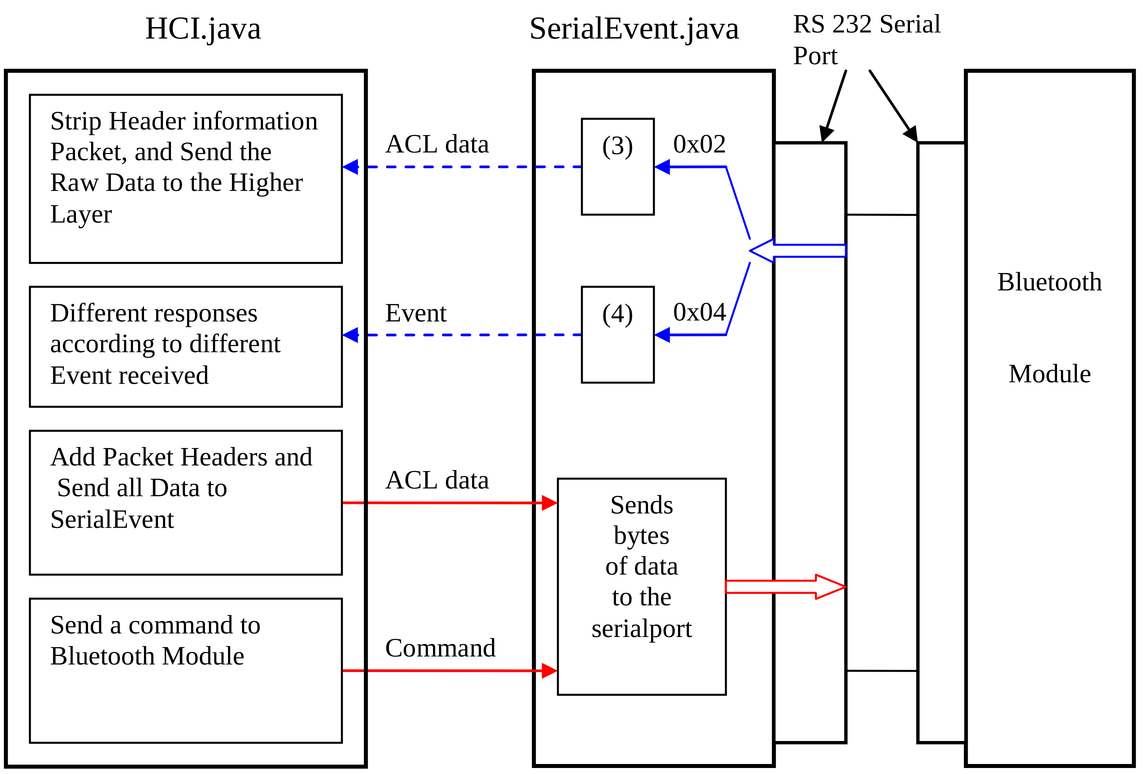

HCI.java
SerialEvent.java
RS 232 Serial Port
Strip Header information Packet, and Send the Raw Data to the Higher Layer
Different responses according to different Event received
Add Packet Headers and Send all Data to SerialEvent
Send a command to Bluetooth Module
ACL data
Event
ACL data
Command
(3)
0x02
(4)
0x04
Sends bytes of data to the serialport
Bluetooth Module

Figure 6.6 Relationships among HCI.java, SerialEvent.java and Bluetooth Module

Figure 6.6 describes the relationship among HCI.java, SerialEvent.java and Bluetooth Module. As illustrated, the NCAP is connected to Bluetooth Module via RS232 serial port.

SerialEvent.java provides a SerialPortEventListener to listen to the serial port. If serial port receives data from external, SerialEvent.java then reads the whole packet, and saves it temporarily. If the first byte of packet Header is 0x02, it means that ACL Data packet is received, and it will be sent to HCI.java, after stripping some corresponding header information. Otherwise, if the first byte of packet header is 0x04, it indicates that Event packet received. After striping corresponding header information, the Event packet is also sent to HCI.java. This process is illustrated in Figure 6.6 with dotted line. If

SerialEvent.java receives any packet either ACL Data packet or command packet from HCI.java, it will simply send them to the serial port as data bytes.

6.5 STIM software

The STIM software, running on the PIC16F877 programmed by C language, consists of four segments:

(1) Bluetooth Codes

(2) LCD Drivers,

(3) Transducer Drivers (The drivers for the transducers, which are connected with the STIM node)

(4) Main System Routines

Bluetooth code is written to initialize the Bluetooth module, and complement other functions. Please refer to the related section for NCAP. In this section, other three segments will be described in detail.

6.5.1 LCD Driver Software

LCD Driver Software is developed to initiate the LCD (including LCD function set) according to LCD user manual, and to illustrate various information on the LCD. Before the information is sent to LCD for display, LCD Driver Software will notify is the LCD of the location that information should be displayed.

6.5.2 Transducer Driver Software

One kind of transducer deployed in the proposed system is temperature sensor DS 1620. In order to get the real-time information, a significant amount of code has been written to control DS 1620, and convert the numerical temperature values to string representations.

Chapter 7 Validation and Results

7.1 Introduction

This chapter reports on the results of experiments conducted to evaluate the performance of the WACNet test-bed, and to validate the algorithms and protocols developed for it. The experimental work on the test-bed is focused on validating the correct operation of its different components including the Monitor, STIM nodes and the NCAP nodes. In addition, the performance of the overall system and the feasibility of WACNet Service Discovery Protocol are also verified. The details of these experimentations and the results obtained are provided in this chapter.

7.2 Validation of WACNet Testbed

In the test-bed, there are two STIM nodes, one NCAP node and one PC deployed as the Monitor. Some experiments are undertaken initially to ensure the correct operation of the system in its most basic configuration. The process is described in the following subsections.

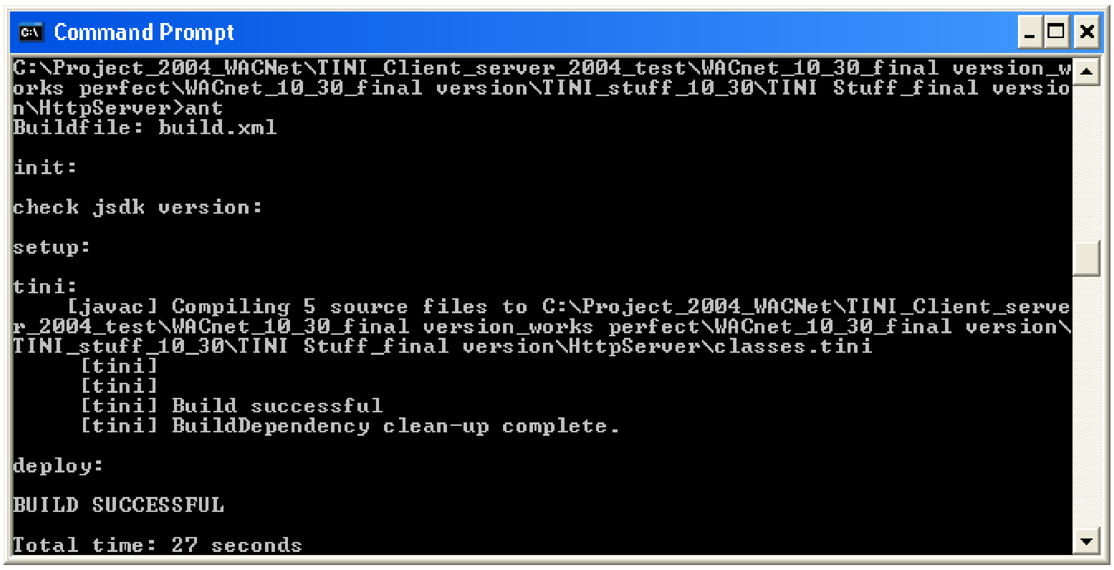
7.2.1 Test of NCAP Node

In the current testbed, each NCAP node consists of a TINI board and an Ericsson Bluetooth Application Toolkit. As described in Chapter 4, TINI board is deployed to enable the proposed system to become network enabled. It takes on the role of a HTTP server and therefore any computer on the network is enabled to access the proposed system without any need for extra drivers.

During the setup process for the NCAP node, a CAT5 crossover cable is first connected to the 10BaseT network adaptor on the TINI board, while TINI and Ericsson Bluetooth Toolkit are connected with each other using a 9 way female-to-female connector. A USB A-B cable is also deployed to supply the Bluetooth Application Toolkit with power.

The Java source code for TINI board is first written and tested on a PC, before it is uploaded into the TINI board. The serial port in the code should be changed from COM1 to serial 0 to ensure its operation on the TINI.

A program called ANT is used in this project to convert all the java class files into a single TINI file, and then upload the TINI file into the TINI board automatically through the crossover cable. The screenshot for the process is illustrated in Figure 7.1.

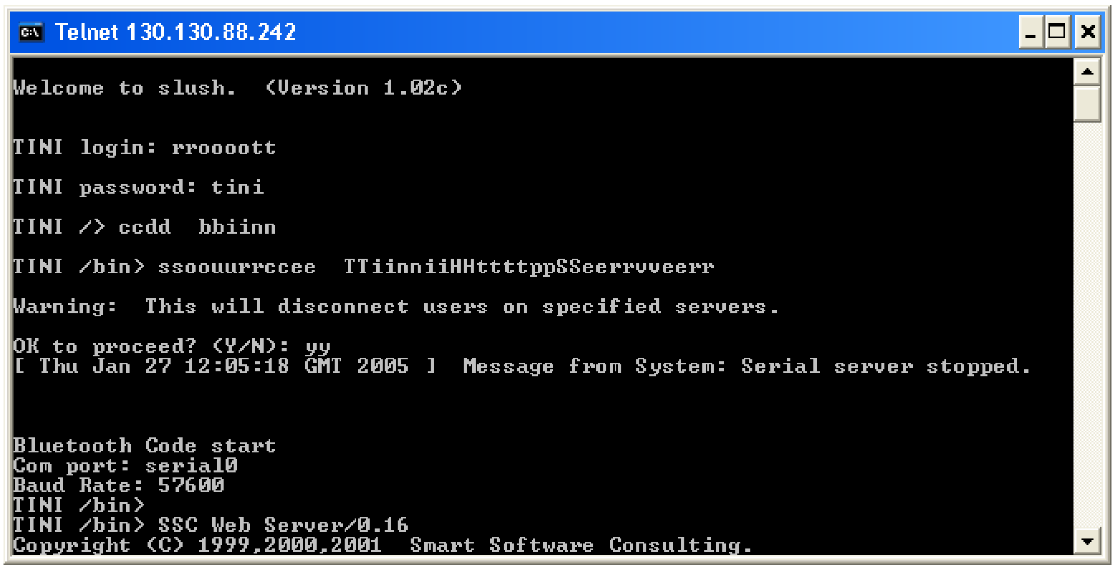


```
C:\Project_2004_WACNet\TINI_Client_server_2004_test\WACnet_10_30_final version_w
orks perfect\WACnet_10_30_final version\TINI_stuff_10_30\TINI Stuff_final versio
n\HttpServer>ant
Buildfile: build.xml

init:

check jsdk version:

setup:

tini:
    [javac] Compiling 5 source files to C:\Project_2004_WACNet\TINI_Client_serve
r_2004_test\WACnet_10_30_final version_works perfect\WACnet_10_30_final version\
TINI_stuff_10_30\TINI Stuff_final version\HttpServer\classes.tini
     [tini]
     [tini]
     [tini] Build successful
     [tini] BuildDependency clean-up complete.

deploy:

BUILD SUCCESSFUL

Total time: 27 seconds
```

Figure 7.1 Upload of the Source Code into TINI board

The build successful message is shown on the screen, which means that the java class files have been successfully converted into a TINI file and then uploaded into the TINI board. The total process takes 27 seconds. In case of failure of build, a message is displayed and the cause of failure is mentioned.

When the program is uploaded into the TINI board, it can be executed via TELNET. When the TINI board is logged into TELNET, several commands are deployed to start the server in the TINI board, illustrated in Figure 7.2. The screenshot below shows that the server in the TINI has been started successfully and the serial port 0 and the Baud Rate of 57600 are designated for the data transmission between the TINI board and the Bluetooth Toolkit.



```
Welcome to slush.  (Version 1.02c)

TINI login: rroooott

TINI password: tini

TINI /> ccdd  bbiinn

TINI /bin> ssoouurrccee  TTiinniiHHttttppSSeerrvveerr

Warning:  This will disconnect users on specified servers.

OK to proceed? (Y/N): yy
[ Thu Jan 27 12:05:18 GMT 2005 ]  Message from System: Serial server stopped.


Bluetooth Code start
Com port: serial0
Baud Rate: 57600
TINI /bin>
TINI /bin> SSC Web Server/0.16
Copyright (C) 1999,2000,2001  Smart Software Consulting.
```

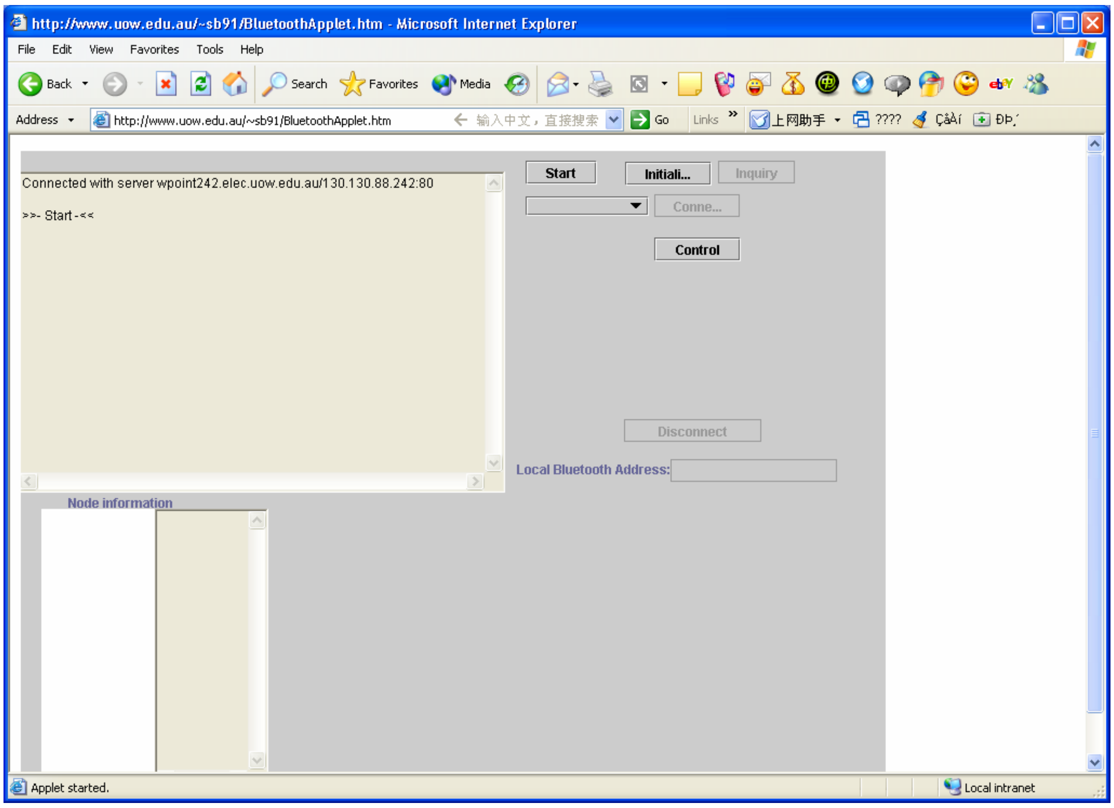
Figure 7.2 Start of the Server in the TINI board

7.2.2 Test of the Monitor

In the current testbed, a desktop computer with a Network Interface Card (NIC) serves as the Monitor. The applet used on the Monitor is placed on the web page http://www.uow.edu.au/~sb91/BluetoothApplet.htm. The Java Runtime Environment and ability to use sockets are required to show the applet on the Monitor. Java Runtime Environment can be downloaded from the WACNet.html web page. To get permission for the socket, the following source code needs to be saved into c:\Documents and Settings\your username\ named as .java.policy (policy file).

```
grant {
permission java.net.SocketPermission "130.130.88.242", "accept, connect, listen,
resolve";
};
```

When the requirements mentioned above are met, applet can be successfully shown on the Monitor via the web page BluetoothApplet.htm. The screenshot of the applet is illustrated in Figure 7.3. The commands sent from the applet and their statuses are shown in the command log on the left top side of the applet. The nodes found by Monitor are listed in an information field called node information on the left low side of the applet.



http://www.uow.edu.au/~sb91/BluetoothApplet.htm - Microsoft Internet Explorer

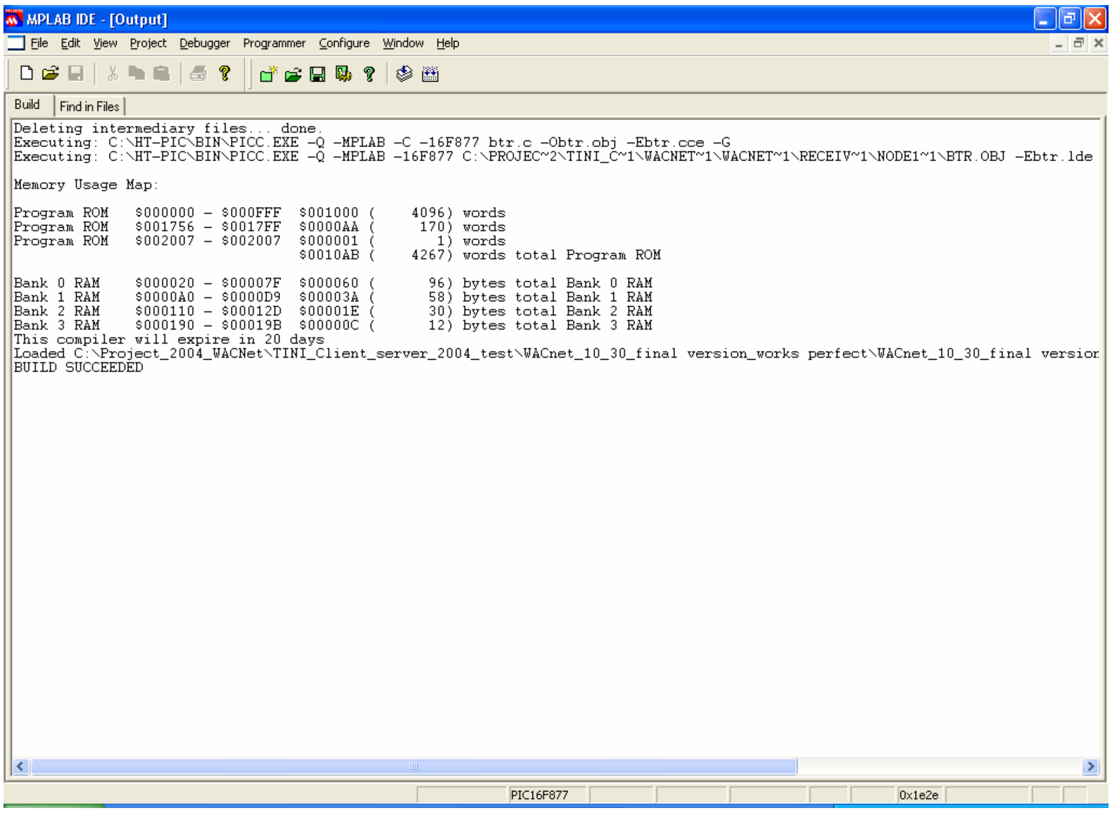
```
Connected with server wpoint242.elec.uow.edu.au/130.130.88.242:80

>>- Start -<<
```

Applet started.

Figure 7.3 Screenshot of Applet

7.2.3 Test of STIM Node

In the current test-bed, each STIM node consists of a main board and a communication board. The source code for STIM node is written in C programming language on the host computer before it is downloaded into the microcontroller PIC16F877 on the STIM via the serial port. A program called MPLAB IDE is used in this project to convert the C files into a single hex file. The screenshot of the result is illustrated in Figure 7.4.

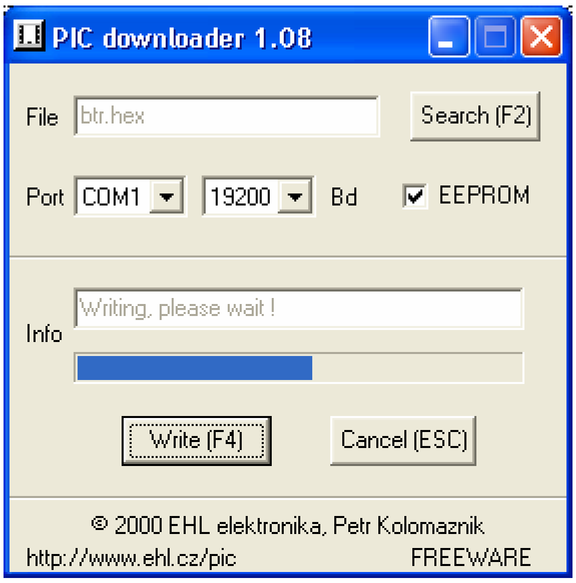

MPLAB IDE - [Output]
File Edit View Project Debugger Programmer Configure Window Help
Build
Find in Files
PIC16F877
0x1e2e

```
Deleting intermediary files... done.
Executing: C:\HT-PIC\BIN\PICC.EXE -Q -MPLAB -C -16F877 btr.c -Obtr.obj -Ebtr.cce -G
Executing: C:\HT-PIC\BIN\PICC.EXE -Q -MPLAB -16F877 C:\PROJEC~2\TINI_C~1\WACNET~1\WACNET~1\RECEIV~1\NODE1~1\BTR.OBJ -Ebtr.lde

Memory Usage Map:

Program ROM   $000000 - $000FFF  $001000 (    4096) words
Program ROM   $001756 - $0017FF  $0000AA (     170) words
Program ROM   $002007 - $002007  $000001 (       1) words
                                 $0010AB (    4267) words total Program ROM

Bank 0 RAM    $000020 - $00007F  $000060 (      96) bytes total Bank 0 RAM
Bank 1 RAM    $0000A0 - $0000D9  $00003A (      58) bytes total Bank 1 RAM
Bank 2 RAM    $000110 - $00012D  $00001E (      30) bytes total Bank 2 RAM
Bank 3 RAM    $000190 - $00019B  $00000C (      12) bytes total Bank 3 RAM
This compiler will expire in 20 days
Loaded C:\Project_2004_WACNet\TINI_Client_server_2004_test\WACnet_10_30_final version_works perfect\WACnet_10_30_final versior
BUILD SUCCEEDED
```

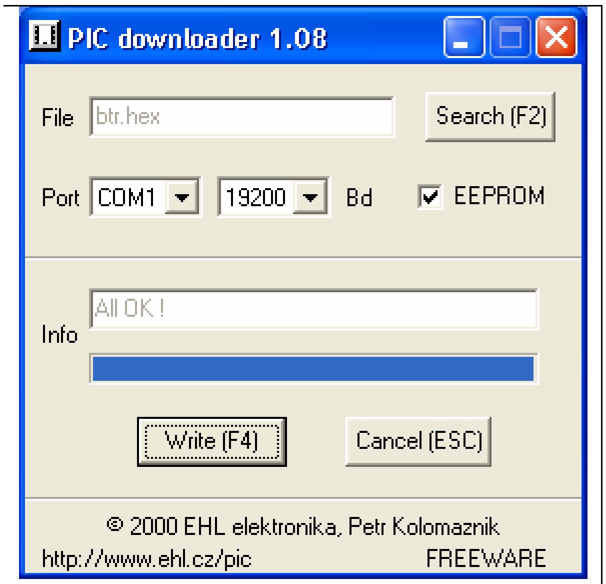
Figure 7.4 - Conversion from C Files to Hex File via MPLAB IDE Program

As seen on the screenshot, the build has been successful. This implies that all the C files have been successfully converted into one hex file. The screenshot also shows how the memory usage map is assigned in the microcontroller PIC 16F877, when the hex file is download into the PIC16F877.

A program called PICdownloader is used in this project to download the hex file into the PIC 16F877 from the computer through the serial port. The process is illustrated in Figure 7.5. The right screenshot shows that the hex file has been successfully downloaded into the PIC16F877 through port COM1 at the baud rate of 19200. When successfully downloaded, the program starts to run automatically. For diagnostic purposes, an LCD is deployed for each STIM node to view and debug the program operation in real-time.



File: btr.hex
Port: COM1, 19200 Bd, ✓ EEPROM
Info: Writing, please wait !



File: btr.hex
Port: COM1, 19200 Bd, ✓ EEPROM
Info: All OK !

Figure 7.5 Download of the Hex File into PIC 16F877 through PIC Downloader

Various messages are shown on the LCD during the test period of the STIM node, including:

(1) Start to initialize

(2) Initialization successful

(3) Show the local sensing information. The local temperature, e.g., 22.5 °C, is shown on the LCD when the temperature sensor is used in the STIM node.

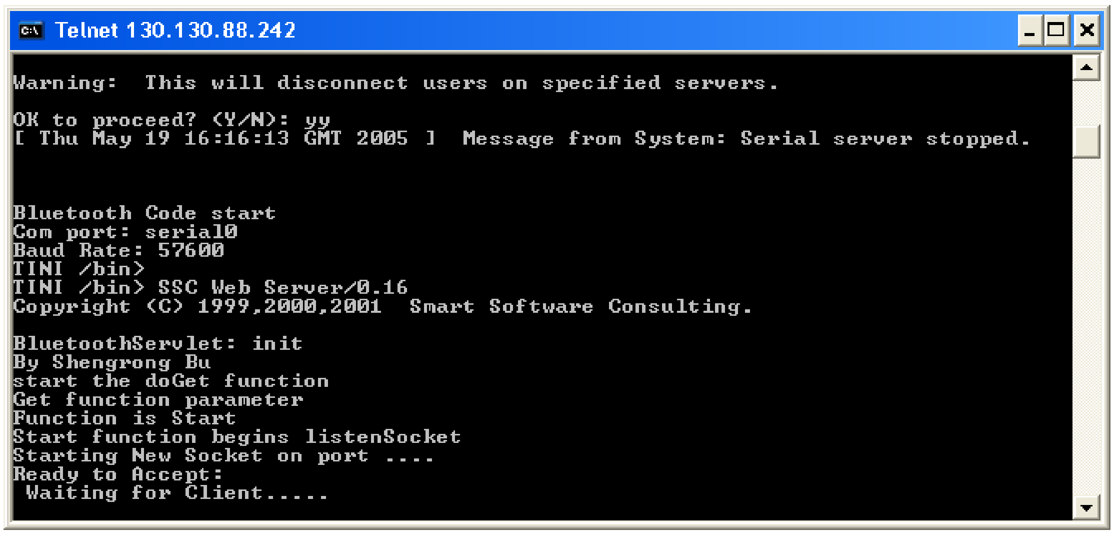
The above test information shows that the STIM node works properly. If the operation fails, an error message is shown on the LCD.

7.2.4 Validation of the Communication Links

A. Communication between Monitor and NCAP Node

Prior to the communication test between the Monitor and NCAP, both a HTTP server and Java servlet process need to start on the NCAP. After that, the HTTP server listens for the requests from the Monitor. The Monitor then downloads the management console from the web page BluetoothApplet.htm, and initializes it on the Monitor.

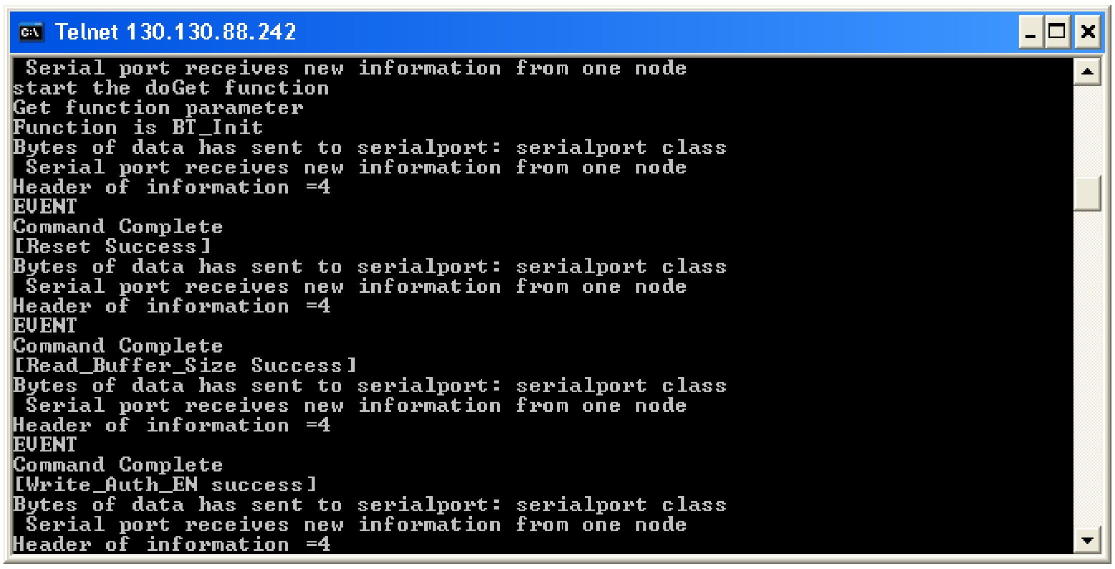
The result on the TINI board is illustrated in Figure 7.6. It shows that the server on the TINI board is ready to accept and wait for the request from the Monitor. On the Monitor side, the applet shows that Monitor is successfully connected to the server 130.130.88.242, which is the IP address of the TINI board.



```
Warning:  This will disconnect users on specified servers.

OK to proceed? (Y/N): yy
[ Thu May 19 16:16:13 GMT 2005 ]  Message from System: Serial server stopped.


Bluetooth Code start
Com port: serial0
Baud Rate: 57600
TINI /bin>
TINI /bin> SSC Web Server/0.16
Copyright (C) 1999,2000,2001  Smart Software Consulting.

BluetoothServlet: init
By Shengrong Bu
start the doGet function
Get function parameter
Function is Start
Start function begins listenSocket
Starting New Socket on port ....
Ready to Accept:
 Waiting for Client.....
```

Figure 7.6 Server on the TINI Board Waits for the Request from the Monitor

When a command button on the applet is pressed, the corresponding HTTP request is sent to the servlet in the NCAP and appropriate action is taken. Figure 7.7 shows the process in the NCAP when the applet button called 'initialize' is pressed. The diagram shows that NCAP has received a HTTP request with the function of BT_Init, and appropriate actions for initialization are taken successfully.



```
 Serial port receives new information from one node
start the doGet function
Get function parameter
Function is BT_Init
Bytes of data has sent to serialport: serialport class
 Serial port receives new information from one node
Header of information =4
EVENT
Command Complete
[Reset Success]
Bytes of data has sent to serialport: serialport class
 Serial port receives new information from one node
Header of information =4
EVENT
Command Complete
[Read_Buffer_Size Success]
Bytes of data has sent to serialport: serialport class
 Serial port receives new information from one node
Header of information =4
EVENT
Command Complete
[Write_Auth_EN success]
Bytes of data has sent to serialport: serialport class
 Serial port receives new information from one node
Header of information =4
```

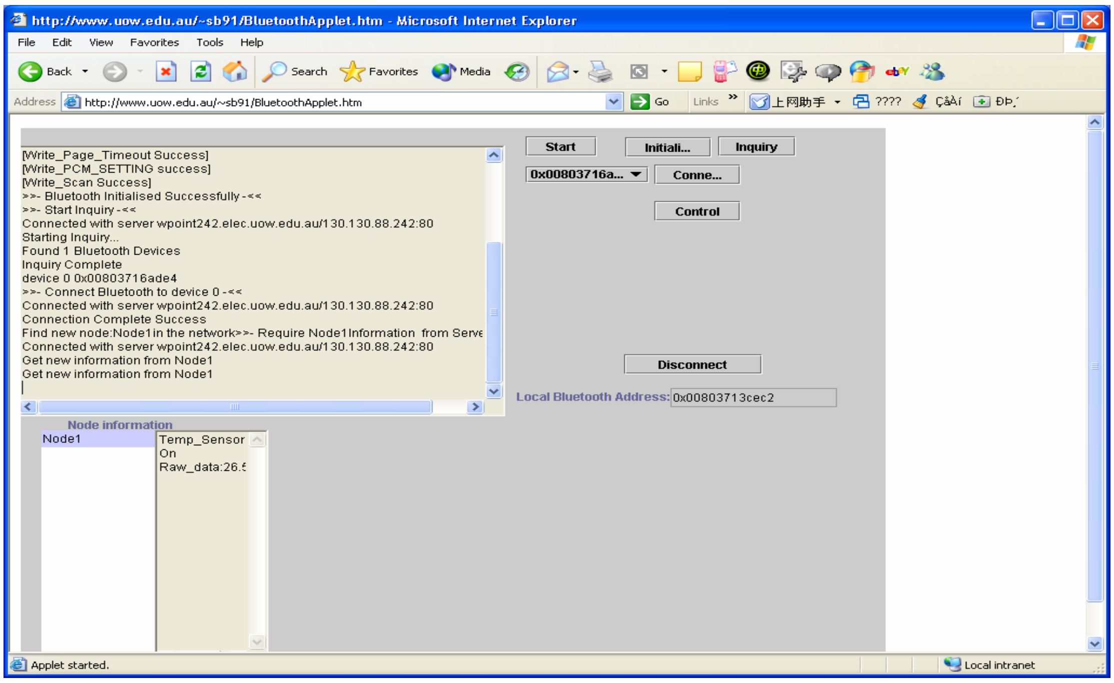
Figure 7.7 Typical Communication Test between Monitor and NCAP

B. Communication between STIM Nodes

This section describes the tests undertaken to ensure that correct communication takes place between STIM nodes. When the power supply of the STIM nodes is switched on, the programs in the PIC16F877 starts to run automatically. After the initialization process, the STIM nodes randomly start the inquiry process and inquiry scan process respectively. The result of the inquiry is then shown on the LCD of each STIM. In the test carried out, one node is designated into inquiry process and another into inquiry scan process. Following that, the STIM nodes go into the connection process. When the connection between STIM nodes is successfully established, the data from one STIM node is transmitted to another, which is then shown on the LCD. After observing the other STIM node, the same text appears without errors. This demonstrates that STIM nodes could communicate with each other without errors.

C. Communication between Monitor, NCAP and STIM

In this communication test, when the NCAP node starts the inquiry process, the inquiry result is shown on the Monitor. Following that, the NCAP node goes into the connection process. When the Bluetooth connections are successfully established between the NCAP and STIM nodes, the data from STIM is successfully received by NCAP, and displayed on the applet of the Monitor, as illustrated in Figure 7.8.


http://www.uow.edu.au/~sb91/BluetoothApplet.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.uow.edu.au/~sb91/BluetoothApplet.htm
Go Links
Start
Initiali...
Inquiry
0x00803716a...
Conne...
Control
Disconnect
Applet started.
Local intranet

```
[Write_Page_Timeout Success]
[Write_PCM_SETTING success]
[Write_Scan Success]
>>- Bluetooth Initialised Successfully -<<
>>- Start Inquiry -<<
Connected with server wpoint242.elec.uow.edu.au/130.130.88.242:80
Starting Inquiry...
Found 1 Bluetooth Devices
Inquiry Complete
device 0 0x00803716ade4
>>- Connect Bluetooth to device 0 -<<
Connected with server wpoint242.elec.uow.edu.au/130.130.88.242:80
Connection Complete Success
Find new node:Node1in the network>>- Require Node1Information  from Serve
Connected with server wpoint242.elec.uow.edu.au/130.130.88.242:80
Get new information from Node1
Get new information from Node1
```

Local Bluetooth Address: 0x00803713cec2

Node information

```
Node1        Temp_Sensor
             On
             Raw_data:26.5
```

Figure 7.8 Result of Communication Test between Monitoring, NCAP and STIM

7.3 Performance of the Overall System

In this section, several tests are undertaken to evaluate the performance of the overall system. In two of the experiments, the transmission rate and the delay in the nodes are examined as the nodes need to operate in real time in WACNet.

Initially, two tests are undertaken to evaluate the performance of the overall system in different types of environments. WACNet might be deployed in industrial environment, where there are different kinds of interference. In the experiment, the WACNet test-bed was first deployed near a microwave oven generating a series of short, sharp noise bursts. Following that, the WACNet test-bed was deployed in a mobile environment, where the STIM nodes were moved around. The results of the experiments are illustrated in Figure 7.9.

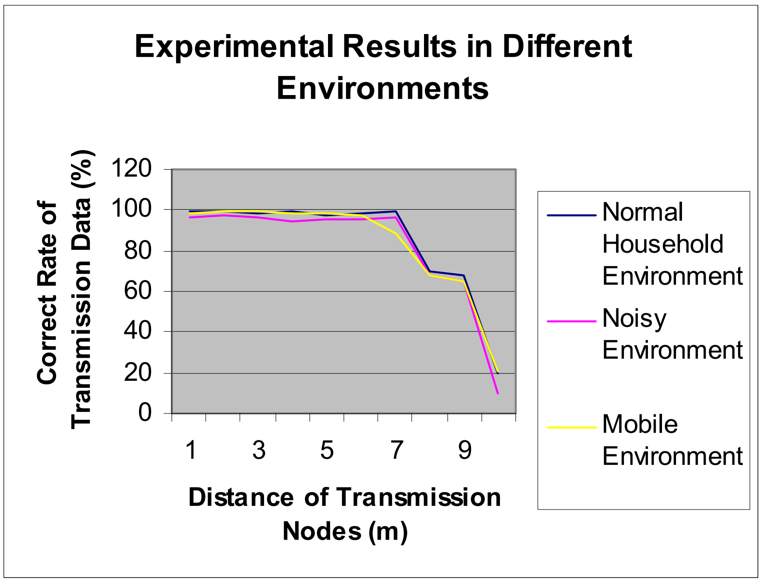


Figure 7.9 Experimental Results in Different Environments

In this diagram, the relationship between the distance of transmission nodes and their correct rate of transmission data in different environments are shown. According to these results, the practical transmission distance of nodes in the test-bed is less than 7 meters compared to the theoretical distance of 10 meters defined for Bluetooth. Within around 7 meters, an increase of distance between the nodes does not cause any data loss. In a noisy environment, the performance of the overall transmission is quite solid with few data loss. The experimental results also show that the movement of the nodes does not have any adverse affect on the overall communication of the system.

Several tests were then undertaken to investigate the delay of the system. The first test shows that it takes around three seconds to completely load the applet from the Internet. The aim of the second experience was to investigate how long it would take for the Monitor to initialize the Bluetooth Module attached to NCAP node, make inquiry to STIM node and connect to it, and then both get ready to exchange data. The execution time of each processor was recorded and displayed by the applet in the Monitor. The results are illustrated in the Figure 7.10.

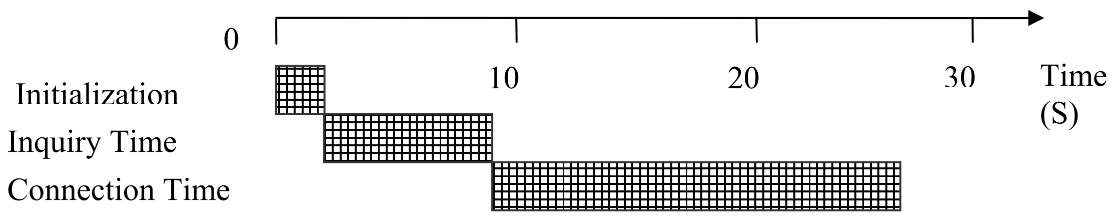


Figure 7.10 Timing of different operations

The results are averaged for 15 experiments. According to these results, the time required for the initialization BT, making inquiry to STIM node, and connect to it are 2.105 sec, 7.076sec and 18.363sec respectively. The connection time is almost doubled when the number of STIM nodes in the test-bed is increased by two fold, while the inquiry time is kept constant. The above results also show that the overall system meets the real-time requirements. The system can be further improved in the future. For example, the RS232 communication between the microcontroller and Bluetooth module on the STIM node can be replaced by a USB port.

7.4 WACNet SDP Experiment

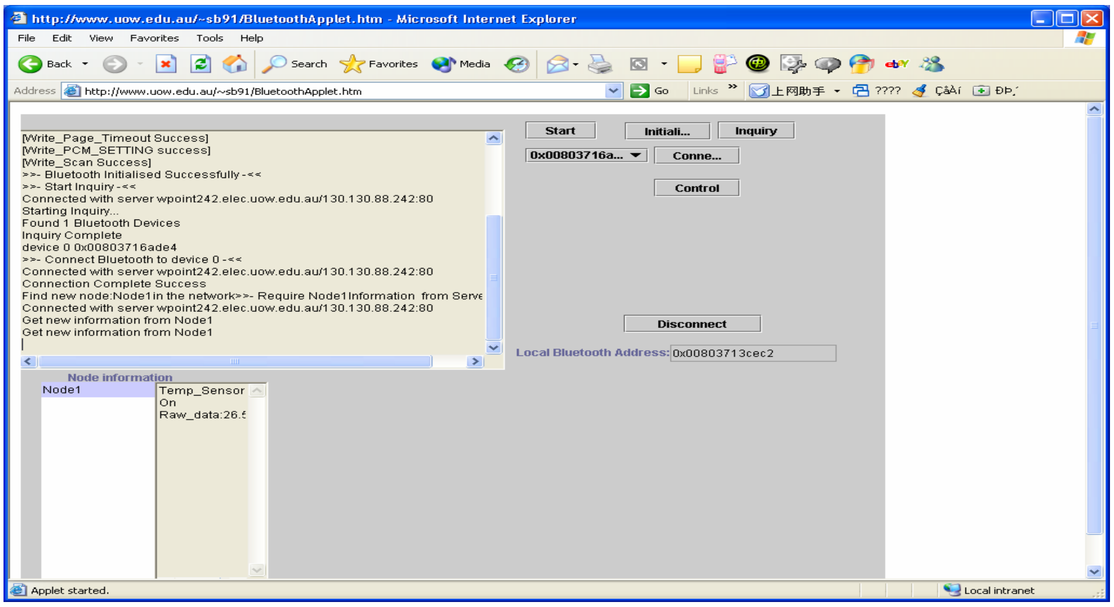
The experiments described in this section were undertaken to verify the WACNet Service Discover Protocol (SDP) based on the first generation test-bed. Nodes in the WACNet should be able to find other complementary network devices, applications and services that are needed to properly complete specified tasks via the use of this SDP.

The first experiment was undertaken for the first part of WACNet SDP, service registration described in section 5.5.2 using the test-bed, where two STIM nodes were set as master nodes in the experiment. After the process of service registration, the service table shown in Table 7.1 was formed in the NCAP node. This table lists the characteristics of these two STIM nodes.

Table 7.1 the Result of Service Record in NCAP Node

Cluster No.	The Number of the Slaves	Channel Type Key	Transducer Priority
00001	000	00000001	00000010
00002	000	00000001	00000010

The result shows that there are only two masters without slaves connected to this NCAP node. The channel Type Key 01 and Transducer Priority 02 mean that transducer is a temperature sensor and real-time requirement is medium according to the definition in section 5.3. This table shows that two master nodes with temperature sensor have registered their services successfully. The corresponding information is also transmitted and shown on the applet in the Monitor.


http://www.uow.edu.au/~sb91/BluetoothApplet.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address http://www.uow.edu.au/~sb91/BluetoothApplet.htm
Go
Links
Start
Initiali...
Inquiry
0x00803716a...
Conne...
Control
Disconnect
Node information
Applet started.
Local intranet

```
[Write_Page_Timeout Success]
[Write_PCM_SETTING success]
[Write_Scan Success]
>>- Bluetooth Initialised Successfully -<<
>>- Start Inquiry -<<
Connected with server wpoint242.elec.uow.edu.au/130.130.88.242:80
Starting Inquiry...
Found 1 Bluetooth Devices
Inquiry Complete
device 0 0x00803716ade4
>>- Connect Bluetooth to device 0 -<<
Connected with server wpoint242.elec.uow.edu.au/130.130.88.242:80
Connection Complete Success
Find new node:Node1in the network>>- Require Node1Information  from Serve
Connected with server wpoint242.elec.uow.edu.au/130.130.88.242:80
Get new information from Node1
Get new information from Node1
```

Local Bluetooth Address: 0x00803713cec2

Node1

```
Temp_Sensor
On
Raw_data:26.5
```

Figure 7.11 Test Result of Service Search

The aim for the second experiment was to search for a node with a particular service using the WACNet SDP. The result for when the user searches for a node with the service ability of either general sensor or temperature sensor via the Monitor is shown in Figure 7.11. The applet screenshot shows the information of the first node responding to the search command of the Monitor applet. The screenshot shows that the node is a temperature sensor and the temperature is 22.5 °C. The related service information of this STIM node has been saved into the text file named *NodeInformationText*.

Chapter 8 Conclusions and Further Research

8.1 Introduction

The WACNet is an innovative concept that has potential to change the way control networks are designed and applied. The focus of the thesis was on developing a conceptual model for the approach and validating the concept based on a proposed test-bed.

The work conducted in the thesis and the results produced are encouraging and indicate that the concept pursued in the research is feasible, though challenging and requiring a significant research and development before it can become commercially viable.

In this chapter, the strengths and the weaknesses of the work in this thesis are identified and the outcomes produced are reviewed. Some conclusions are drawn and the possible future directions of the research will be highlighted.

8.2 Feasibility of the Concept

A comprehensive study of the evolution of control systems architecture was carried out in the thesis. It was illustrated that with the emergence of affordable digital technology, computer networking and smart sensors and actuators, the control systems are evolving towards highly distributed systems. The main motivations are lower cost, more efficient design and implementation, higher reliability, and easier maintainability and extensibility.

The study also clearly indicates that WACNet concept is a logical extension of the current distributed networked control systems towards ad-hoc architecture. In such system, intelligent loosely coupled nodes plan and execute their control tasks through service discovery, self-organization and effective communication. This is a significant shift from currently hierarchical control systems in which the control nodes perform the tasks assigned to them by a supervisor within a well defined and fixed configuration.

While the concept is logically feasible and offers a scalable architecture lending itself to large systems, the implication of applying it to a large number of nodes is not well understood in this project, due to the limited scope of the research.

8.3 First Generation Test-bed

The first generation test-bed designed and developed in this work is based on two recently established industry standards of IEEE1451 and Bluetooth technology. The architecture emerged as the result of synergy between these two technologies, well satisfies most of the requirements defined for WACNets model. In particular, the developed nodes offer:

(a) True ad-hoc structure, simplifying the design, maintenance, extensibility, and scalability of the system built based on them.

(b) True peer-to-peer communication with no central supervisor.

(c) Cost effective scalability.

(d) Cost effective wireless communication, which significantly reduced cost of commissioning and maintenance of a control system.

(e) Provision for an evolutionary system through its ability to reconfigure in an autonomous fashion and to provide an optimal distribution of resources.

(f) Robustness with respect to both disturbances and uncertainties.

(g) Interoperability in heterogeneous system environments.

In the first generation test-bed, the design of the nodes has not considered the constraints of cost, power, and size. Hence, the cost of the nodes is not low enough for their mass utilisation. They have relatively high power usage and work on the DC current supplied by a power supply rather than a battery. While small compared to other similar microcontroller boards, the nodes have a larger size compared with the characteristics typically defined for the nodes used in sensor networks.

8.4 Validation of the Concept

In the course of the project, the overall performance of the WACNet test-bed, and the algorithms and protocols developed for it were validated through experimental work. This included ensuring the correct operation of the Monitor, STIM nodes and the NCAP nodes, as well as the Service Discovery Protocol developed for WACNet.

The main tool used in the validation was an applet designed and developed for the Monitor. It provided a graphical interface to the status of the nodes in the WACNet and the activities taking place in it through four fields including the Command Log displaying a list of the commands used, control buttons to issue direct instruction to NCAP or a STIM to provide a particular information, and Information field in which the responses of WACNet to the received commands are displayed.

The validation work provided an insight into the operation of the test-bed and proved that it correctly carried out the functions built into it. It, however, did not sufficiently examine the feasibility of all the generic characteristics defined for the WACNet model. This was due to two factors of the limited time of the project and the small number of nodes available in the test-bed.

8.5 Future Work

Full study and implementation of WACNet is a long term project since many key questions should be answered and critical requirements described in the previous chapters of the thesis to be satisfied. The work carried out in this thesis has been a major step towards understanding of how to design and carry out the future studies and developments of WACNet.

Further research on the project can address the following important areas:

(a) The design of the nodes should be reviewed to accommodate the other requirements defined for WACNet nodes including a more compact circuit to minimize the size of STIM nodes, improving the energy efficiency of the nodes, and reducing their cost. The possibility of deploying an alternative wireless technology such as ZigBee to satisfy these requirements should be considered. A larger number of nodes should be produced in the second generation WACNet test-bed.

(b) The efficiency of communication in WACNets particularly when control loops are implemented across two or more nodes should to be studied. This could result in the development of special routing algorithms for the transfer of sensory and control data from one cluster in the network to the others.

(c) The thesis has not studied in depth the concept of self-organization of the nodes during the evolution and operation of the network. This is one of the most important and critical characteristics of WACNet which requires quality study and research.

(d) The impact of delay caused by communication in the network on the control systems implemented across multiple nodes should be studied. This will result in certain constraints which should govern the self-organization algorithms to ensure that infeasible clusters are not formed in the network.

(e) The effectiveness of WACNet in comparison with other control systems architectures should be examined through a number of case studies.

Publications Based on Worked Performed on This Thesis

I. Shengrong Bu, Fazel Naghdy, "Wireless ad-hoc Control Networks," submitted to *IEEE transactions on Industrial Informatics* in 2005.

II. Shengrong Bu, Fazel Naghdy, "Service Discovery of Wireless ad-hoc Control Networks," accepted by *Second International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 5-8 December 2005, Melbourne, Australia.

III. Shengrong Bu, Fazel Naghdy, "Wireless ad-hoc Control Networks," *3rd International IEEE conference on Industrial informatics*, Perth, 10-12, August 2005.

IV. S. Bu, F.Naghdy, "Physical Layer of Wireless ad-hoc Control Networks," *Proc. International Manufacturing Leaders Forum (IMLF-2005)*, Adelaide, 27 Feb-2 March 2005.

References

[1] *ECHELON* [Homepage of Echelon], [Online]. (05 January 2003 – last update). Available: http://www.echelon.com [Accessed 2003, July 23].

[2] *DDC Online* [Homepage of DDC-Online], [Online]. (12 December 2002 – last update). Available: http://www.ddc-online.org [Accessed 2003, August 12].

[3] *Controller Area Network – CAN information* [Online]. (No date). Available: http://www.algonet.se/~staffann/developer/CAN.htm [Accessed 2003, August 15].

[4] F.-L. Lian, J. R. Moyne, and P.M.Tilbury, "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet," *IEEE Control Systems*, vol. 21, pp.66-83, February 2001.

[5] *ControlNet Home* [Homepage of Rockwell Automation], [Online]. (02 November 2002 – last update). Available: http://www.ab.com/networks/controlnet.html [Accessed 2003, September 01].

[6] E. Tovar and F. Vasques, "Cycle time properties of the Profibus timed token protocol," *Computer Communication*, vol. 22, pp. 1206–1216, 1999.

[7] E. Tovar and F. Vaspues, "Real-time fieldbus communications using Profibus networks," *IEEE Transactions on Industrial Electronics*, vol. 46, pp. 1241–1251, Dec. 1999.

[8] S. Vitturi, "Some features of two fieldbuses of the IEC 61158 standard," *Computer Standards Interfaces*, vol. 22, pp. 203–215, 2000.

[9] *Profibus Specification—Normative Parts of Profibus-FMS, -DP, -PA According*

to the European Standard, EN 50170, 1998.

[10] S. H. Hong, “Experimental performance evaluation of Profibus-FMS,” *IEEE Robotic Automation Magazine*, pp. 64–72, Dec. 2000.

[11] S. H. Hong and Y. C. Kim, “Implementation of a bandwidth allocation scheme in a token-passing fieldbus network,” *IEEE Transactions on Instrumentation and Measurement*, vol. 51, pp. 246-251, Apr. 2002.

[12] S. Lee, K. C. Lee, M. C. Han, and J. S. Yoon, “On-line Fuzzy performance management of Profibus networks,” *Comput. Ind.*, vol. 46, no. 2, pp. 123-137, Sept. 2001.

[13] P. Otanez, J. Parrott, J. Moyne, and D. Tilbury, “The implications of Ethernet as a control network,” in *Proc. of the Global Powertrain Conference*, August 2002.

[14] K. C. Lee and S. Lee, “Performance evaluation of switched Ethernet for networked control systems,” *IEEE 28th Annual Conference of the Industrial Electronics Society*, Vol. 4, pp. 3170-3175, Nov. 2002.

[15] S. K. Kweon, K. G. Shin and Q. Zheng, “Statistical real-time communication over Ethernet for manufacturing automation systems,” *Processing of the Fifth IEEE Read-Time Technology and Applications Symposium*, pp.192 – 202, June 1999.

[16] C. Venkatramani and T. Chiueh, “Supporting real-time traffic on Ethernet,” *Proceedings of Real-Time Systems Symposium*, pp.282 – 286, Dec. 7-9, 1994.

[17] S. K. Kweon and K. G. Shin, “Ethernet-based real time control networks for manufacturing automation systems”, *University of Michigan CSE technical report,* 2000.

[18] J. K. Yook, D.M. Tilbury, H.S.Wong and N.R. Soparkar, “Trading computations for bandwidth: State estimators for reduced communication in distributed control

systems," *Proceedings of the Japan-USA Symposium on Flexible Automation*, July, 2000.

[19] H. Shahnasser, Q. Wang, "controlling industrial devices over TCP/IP by using LonWorks," *Proc. IEEE Global Telecommunications Conference,* Volume 2, pp.1304-1314, 1998.

[20] A. Lim, "Support for reliability in self-organizing sensor networks," *Proc. Fifth International Conference on Information fusion*, pp.973-980, Vol. 2, 2002.

[21] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *33rd Hawaii International Conference on System Sciences*, Vol. 02, Jan. 2000.

[22] L. Clare, G. Pottie, and J. Agre, "Self-organizing distributed sensor networks, " in *Proc. SPIE Conf. Unattended Ground Sensor Technologies and Applications*, vol. 3713, Orlando, FL, Apr. 1999, pp. 229--237.

[23] G. Asada, M. Dong, T. S. :om. F. Newberg, G. Pottie, W. J. Kaiser, and H. O. Marcy, "Wireless Integrated Network Sensors: Lower Power Systems on a Chip," *Proc. 1998 European Solid State Circuits conference*, pp.9-16, September 1998.

[24] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communication Magazines*, pp.102-114, August 2002.

[25] Robert N. Johnson, IEEE-1451.2 Update, *Sensors*, Vol.17, No.1, January 2000.

[26] Zhang, W., Branicky, M.S. & Phillips, S.M., "Stability of networked control systems," *IEEE Control Systems Magazine*, Vol. 21, Iss. 1, pp 84-99, 2001.

[27] Lim, etc. "Support for reliability in self-organizing sensor networks," *Proc. Fifth International Conference on Information fusion*, Vol 2, pp 973-980, 2000.

[28] *Microchip* [Homepage of Microchip Technology InC.], [Online]. (25 May 2003 – last update). Available: Http://www.microchip.com [Accessed 2003, October 3].

[29] *USART- Using the USART in Asynchronous Mode* [Online]. (No date). Available: Http://ww1.microchip.com/downloads/en/DeviceDoc/usart.pdf [Accessed 2003, Nov. 11].

[30] *Maxim* [Homepage of Dallas Semiconductor], [Online]. (21 September 2003 – last update). Available: Http://www.maxim-ic.com [Accessed 2003, October 4].

[31] *Ericsson* [Homepage of Ericsson], [Online]. (07 September 2003 – last update). Available: http://www.ericsson.com [Accessed 2003, October 12].

[32] *Teleca* [Homepage of Teleca], [Online]. (11 January 2003 – last update). Available: http://www.comtec.teleca.se/index.asp [Accessed 2003, October 1].

[33] *ChipDocs* [Homepage of Datasheets for Electronic Components], [Online]. (12 September – last update). Available: http://www.chipdocs.com/manufacturers/ERSON.html [Accessed 2003, October 2].

[34] *SUYIN* [homepage of SUYIN-connector], [Online]. (23 November 2002 – last update). Available: http://www.suyin-europe.com [Accessed 2003, October 14].

[35] *BluetoothWeb* [Homepage of the Wireless Directory], [Online]. (11 November 2002 – last update). Available: http://www.thewirelessdirectory.com/Bluetooth-Development/index.htm [Accessed 2003, October 11].

[36] Bray, Jennifer, Bluetooth: Connect without cables, Upper Saddle River, N.J.: Prentice Hall, c2002

[37] *Logical Families / Objectives* [Online]. (No data). Available: http://www.ul.ie/~rinne/ee6471/ee6471%20wk6.pdf.

[38] F. Mattern, "State of the Art and Future Trends in Distributed Systems and Ubiquitous Computing," *Vontobel TeKnoBase*, August 2000.

[39] Christian Bettstetter and Christoph Renner, "A Comparison of Service Discovery Protocols and Implementation of the Service Location," *Proceedings of 6th EUNICE Open European Summer School: Innovative Internet Application (EUNICE)*, Twente, Netherlands, Sept 13-15, 2000.

[40] *Bluetooth* [Homepage of the Official Bluetooth Website], [Online]. (01 July 2003 – last update). Available: http://www.bluetooth.com [Accessed 2003, October 10].

[41] *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats* [Online]. (No date). Available: http://ieeexplore.ieee.org.ezproxy.uow.edu.au:2048/xpl/standards.jsp?findtitle=1451.2&letter=1451.2.

[42] *Smart Software Consulting* [Homepage of Smart Software Consulting], [Online]. (21 November 2001 – last update). Available: http://www.smartsc.com [Access 2003, October 15].

[43] *Bluetooth* [Homepage of the official Bluetooth Membership site], [Online]. (11 June 2003 – last update). Available: https://www.bluetooth.org [Access 2003, October 12].

[44] *ZigBee* [Homepage of ZigBee Alliance], [Online]. (20 November 2004 – last update). Available: http://www.zigbee.org/en/index.asp [Access 2005, April 11].

Appendix A Bluetooth Overview

Source: the Bluetooth System Specification. http://www.bluetooth.com.

Bluetooth, a short-range radio link, has been designed to replace cables between electronic devices within 10 m. Bluetooth operates on the radio-license free 2.4 GHz Industrial, Scientific and Medical (ISM) band. A frequency-hopping scheme is deployed to combat interference and fading in the Bluetooth, which enables it to work in point-to-point and multi-point connects.

Bluetooth is based on a stacked protocol model, where the communication is divided into several layers, which are shown in Figure A.1. The lower layers of stack consist of the Host Control Interface (HCI), Audio, Link Manager (LM), Baseband and Bluetooth Radio, while the higher layers includes L2CAP, RFCOMM, SDP and etc.

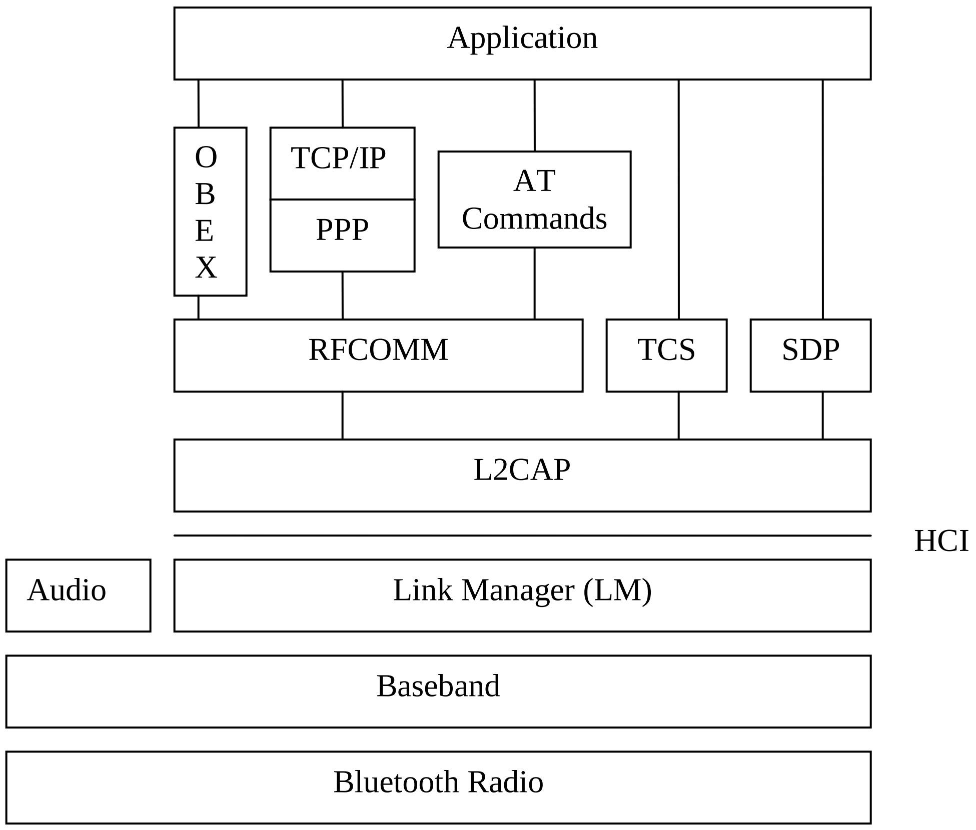


Figure A. 1: Bluetooth Layers

There are three different types of HCI packets: HCI command packets, HCI event packets and HCI data packets, whose formats are shown in the A.2, A.3 and A.4 respectively.

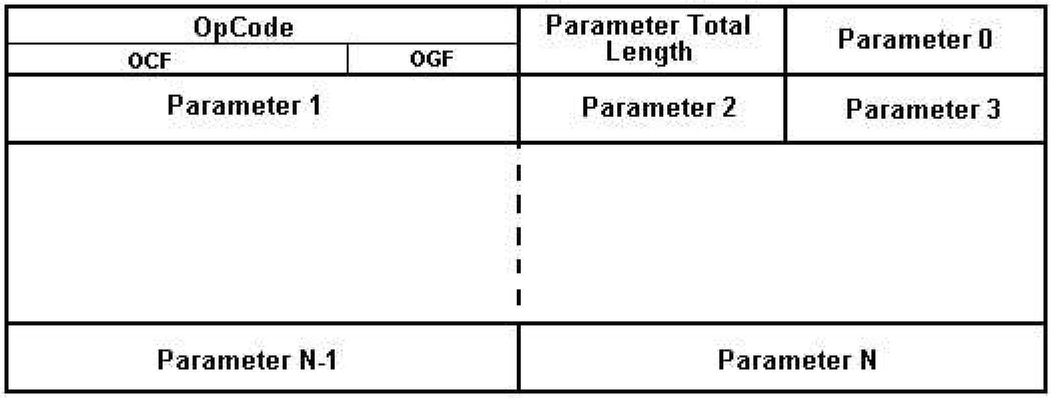


Figure A.2: HCI Command Packet

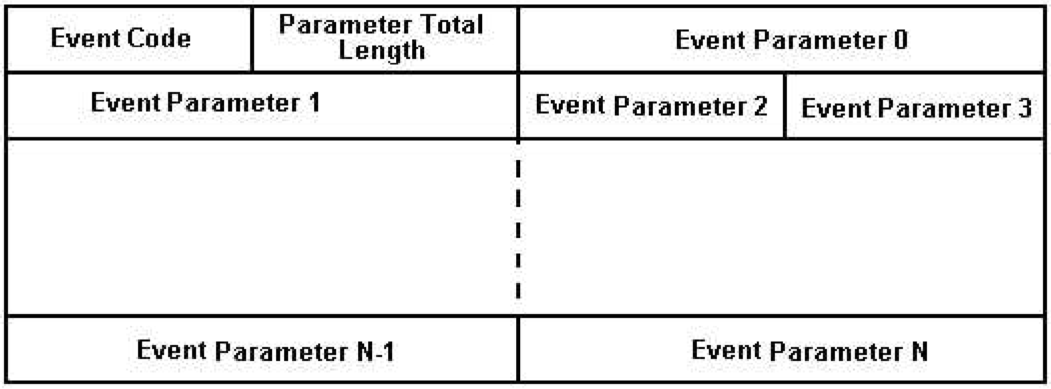


Figure A.3: HCI Event packet

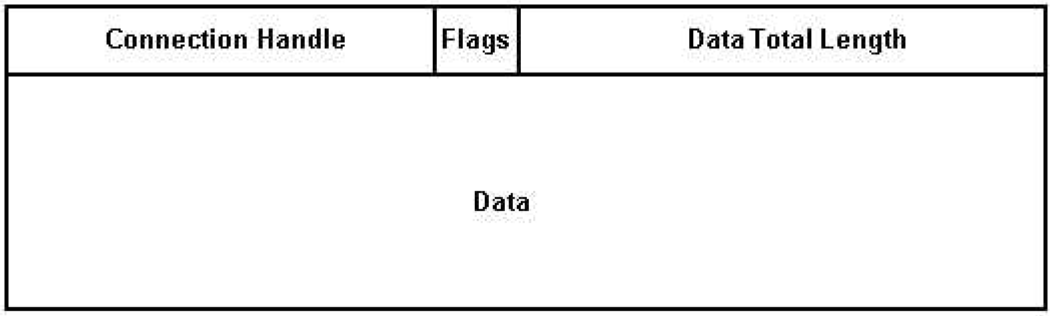


Figure A.4: HCI Data Packet

Appendix B IEEE 1451.2 Standards

Sources: IEEE 1451.2 Standard Datasheets. http://ieeexplore.ieee.org.ezproxy.uow.edu.au:2048/xpl/standardstoc.jsp?isnumber=15569&isYear=1998

IEEE 1451.2 standard has defined a smart transducer interface model (STIM), a transducer electronic data sheet (TEDS) and a digital interface to access the data. Its schematic structure of the model is illustrated in Figure 1.

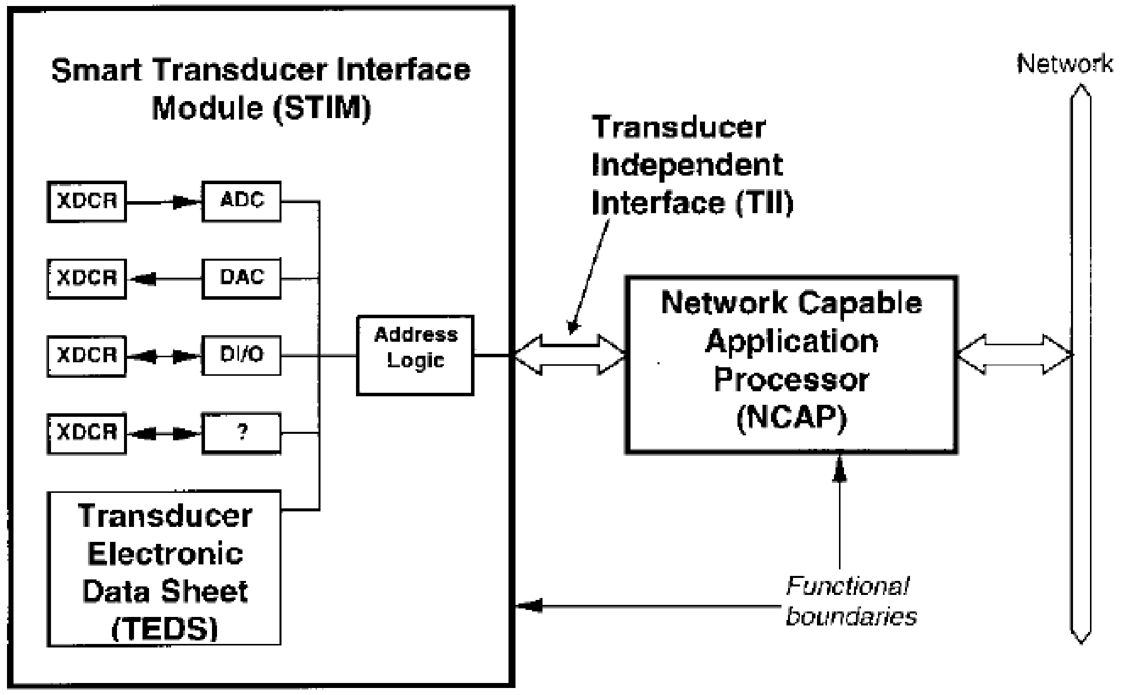
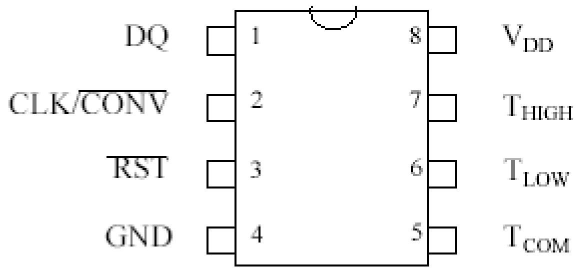

Smart Transducer Interface Module (STIM)
XDCR
ADC
XDCR
DAC
XDCR
DI/O
XDCR
?
Transducer Electronic Data Sheet (TEDS)
Address Logic
Transducer Independent Interface (TII)
Network Capable Application Processor (NCAP)
Network
Functional boundaries

Figure 1—Context for the transducer interface specification

TEDS, embedded within STIM, are electronic datasheets, which describes the properties of the total STIM unit and transducer channels it controls. In each STIM node, TEDS consists of one meta-TEDS that provides common information for all of the transducers and several channel-TEDSs that provide information about each transducer.

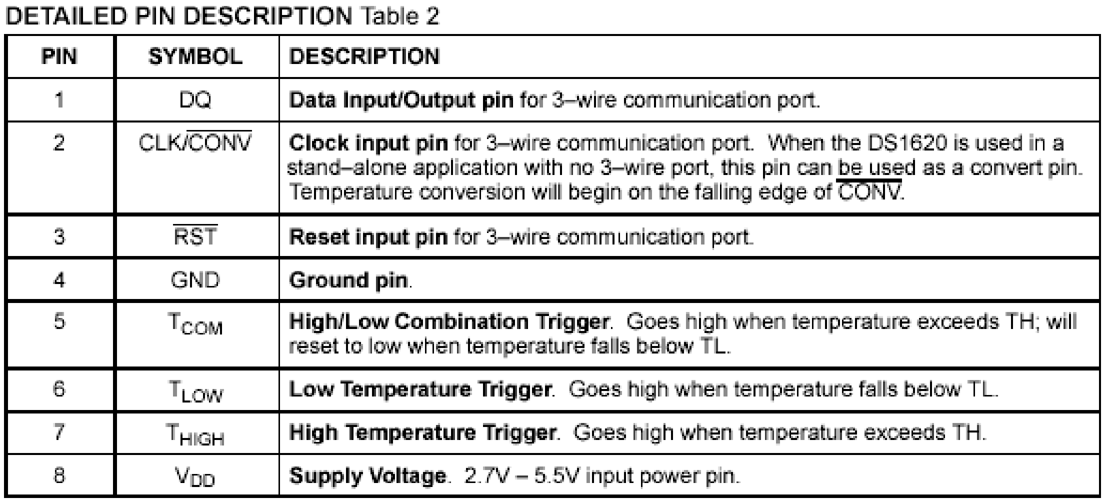
Appendix C Extract from the DS1620 Specifications

Source: DS1620 datasheet. http://www.glotov.pp.ru/filebase/cpu/1620.PDF



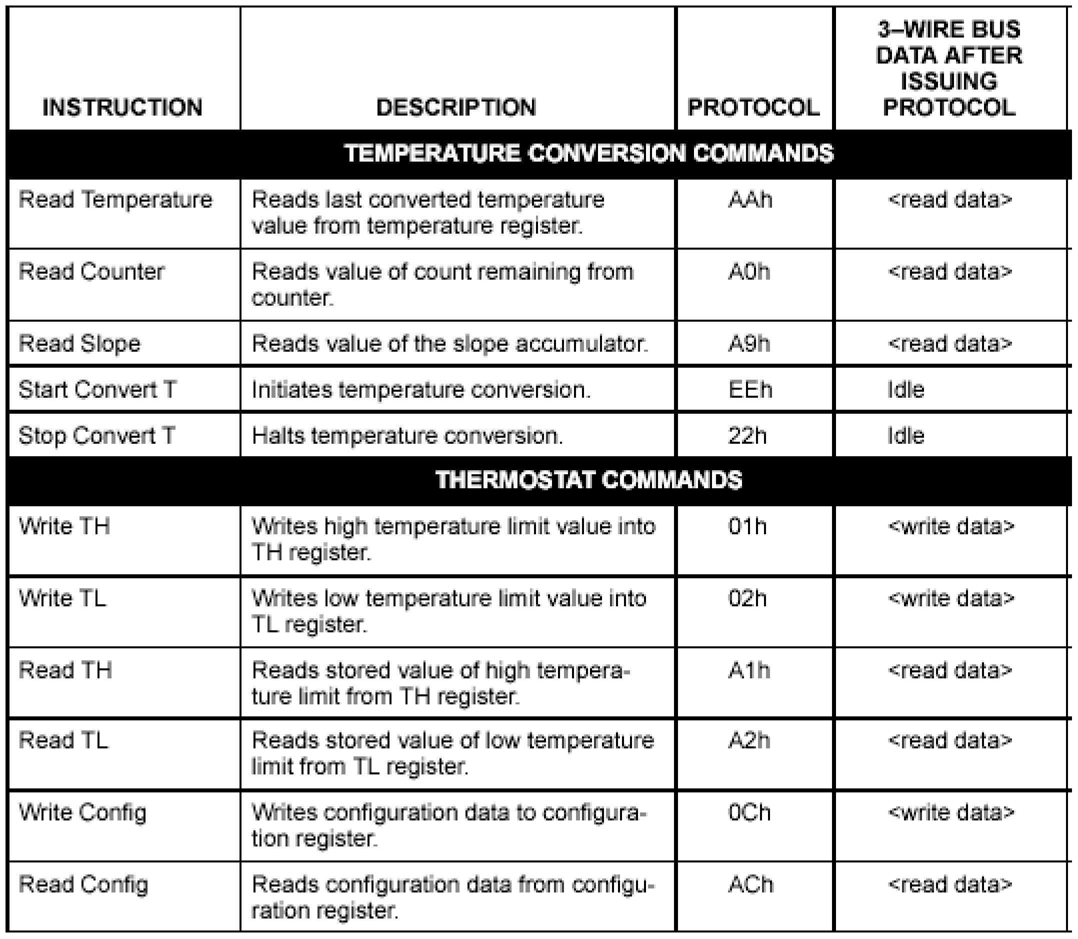
DS1620 8-Pin DIP (300-mil)
See Mech Drawings Section

The DS1620 provides 9-bit temperature readings which indicate the temperature of the device from -55 ^oC to +125 ^oC in 0.5 ^oC increments. Pins and command set of DS 1620 are described in the table 2 and 3 respectively.



DETAILED PIN DESCRIPTION Table 2

PIN	SYMBOL	DESCRIPTION
1	DQ	**Data Input/Output pin** for 3–wire communication port.
2	CLK/$\overline{\text{CONV}}$	**Clock input pin** for 3–wire communication port. When the DS1620 is used in a stand–alone application with no 3–wire port, this pin can be used as a convert pin. Temperature conversion will begin on the falling edge of $\overline{\text{CONV}}$.
3	$\overline{\text{RST}}$	**Reset input pin** for 3–wire communication port.
4	GND	**Ground pin**.
5	T_{COM}	**High/Low Combination Trigger**. Goes high when temperature exceeds TH; will reset to low when temperature falls below TL.
6	T_{LOW}	**Low Temperature Trigger**. Goes high when temperature falls below TL.
7	T_{HIGH}	**High Temperature Trigger**. Goes high when temperature exceeds TH.
8	V_{DD}	**Supply Voltage**. 2.7V – 5.5V input power pin.

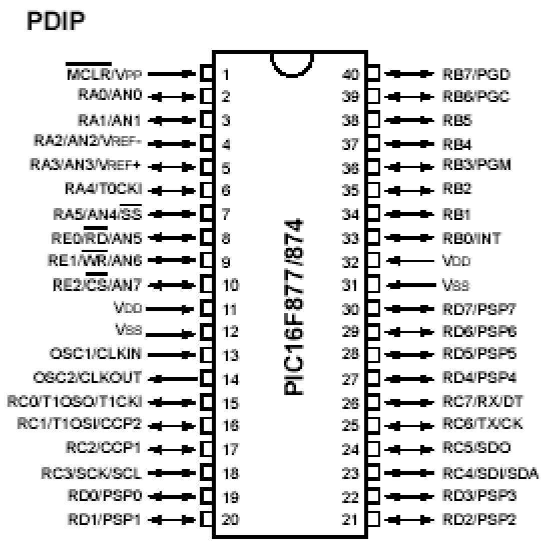


DS1620 COMMAND SET Table 3

INSTRUCTION	DESCRIPTION	PROTOCOL	3–WIRE BUS DATA AFTER ISSUING PROTOCOL
TEMPERATURE CONVERSION COMMANDS			
Read Temperature	Reads last converted temperature value from temperature register.	AAh	<read data>
Read Counter	Reads value of count remaining from counter.	A0h	<read data>
Read Slope	Reads value of the slope accumulator.	A9h	<read data>
Start Convert T	Initiates temperature conversion.	EEh	Idle
Stop Convert T	Halts temperature conversion.	22h	Idle
THERMOSTAT COMMANDS			
Write TH	Writes high temperature limit value into TH register.	01h	<write data>
Write TL	Writes low temperature limit value into TL register.	02h	<write data>
Read TH	Reads stored value of high tempera-ture limit from TH register.	A1h	<read data>
Read TL	Reads stored value of low temperature limit from TL register.	A2h	<read data>
Write Config	Writes configuration data to configura-tion register.	0Ch	<write data>
Read Config	Reads configuration data from configu-ration register.	ACh	<read data>

Appendix D Extract from the PIC16F877 Specifications

Source: PIC16F877 datasheet.
http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf



Some of the key features for PIC16F877 are described in the following table.

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

Appendix E Ericsson ROK 101007 Module Specifications

Source: Ericsson ROK 101 007 Module Datasheet.

ROK 101007 is designed to implement Bluetooth functionality into various electronic devices. The Module mainly includes a Baseband control, a flash memory and a radio, and also supports data and voice transmission in the globally available 2.4-2.5 free ISM band. Its Electrical characteristics and mechanical Specification are described in the following:

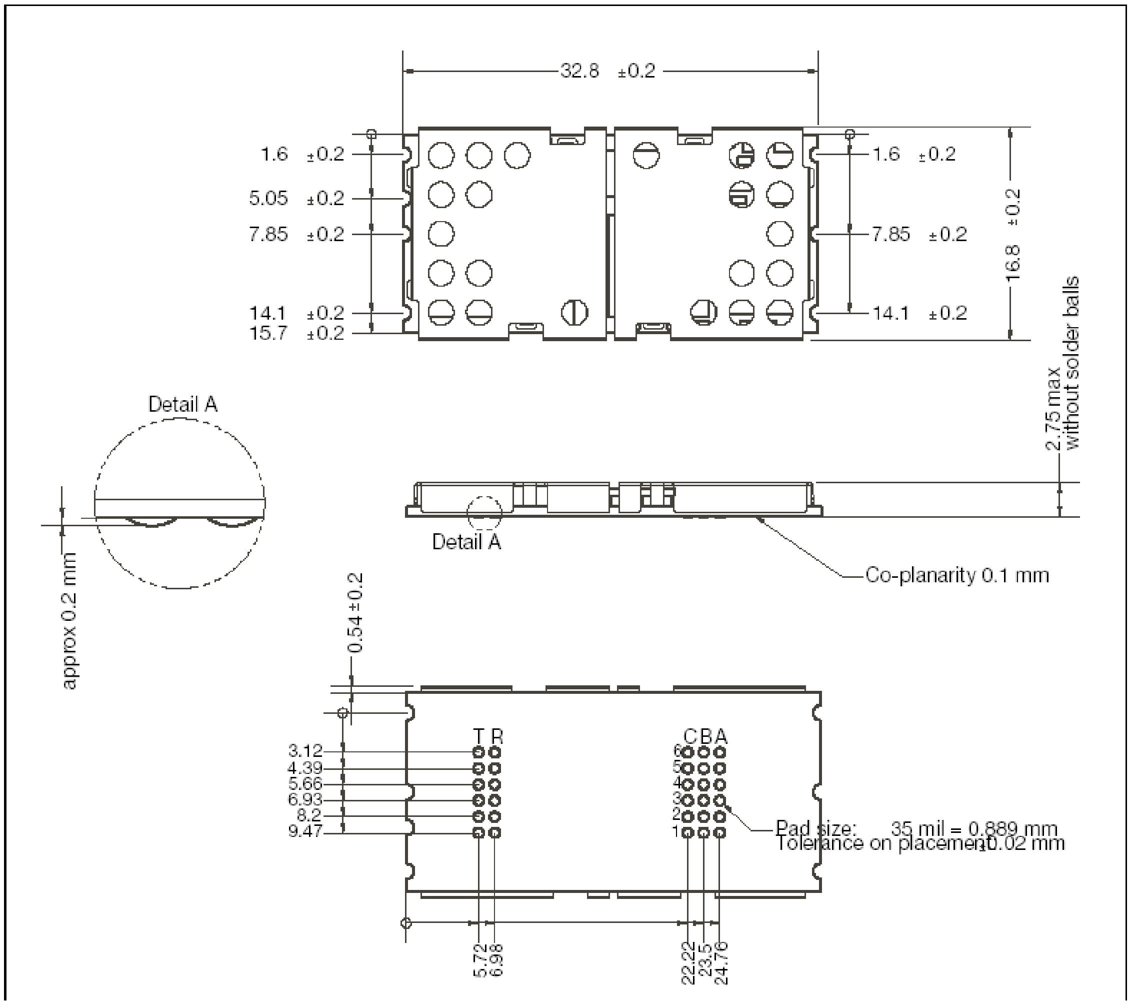
Electrical Characteristics

DC Specifications

Unless otherwise noted, the specification applies for T_{Amb} = 0 to +75°C, $3.175 < V_{CC} < 5.25V$

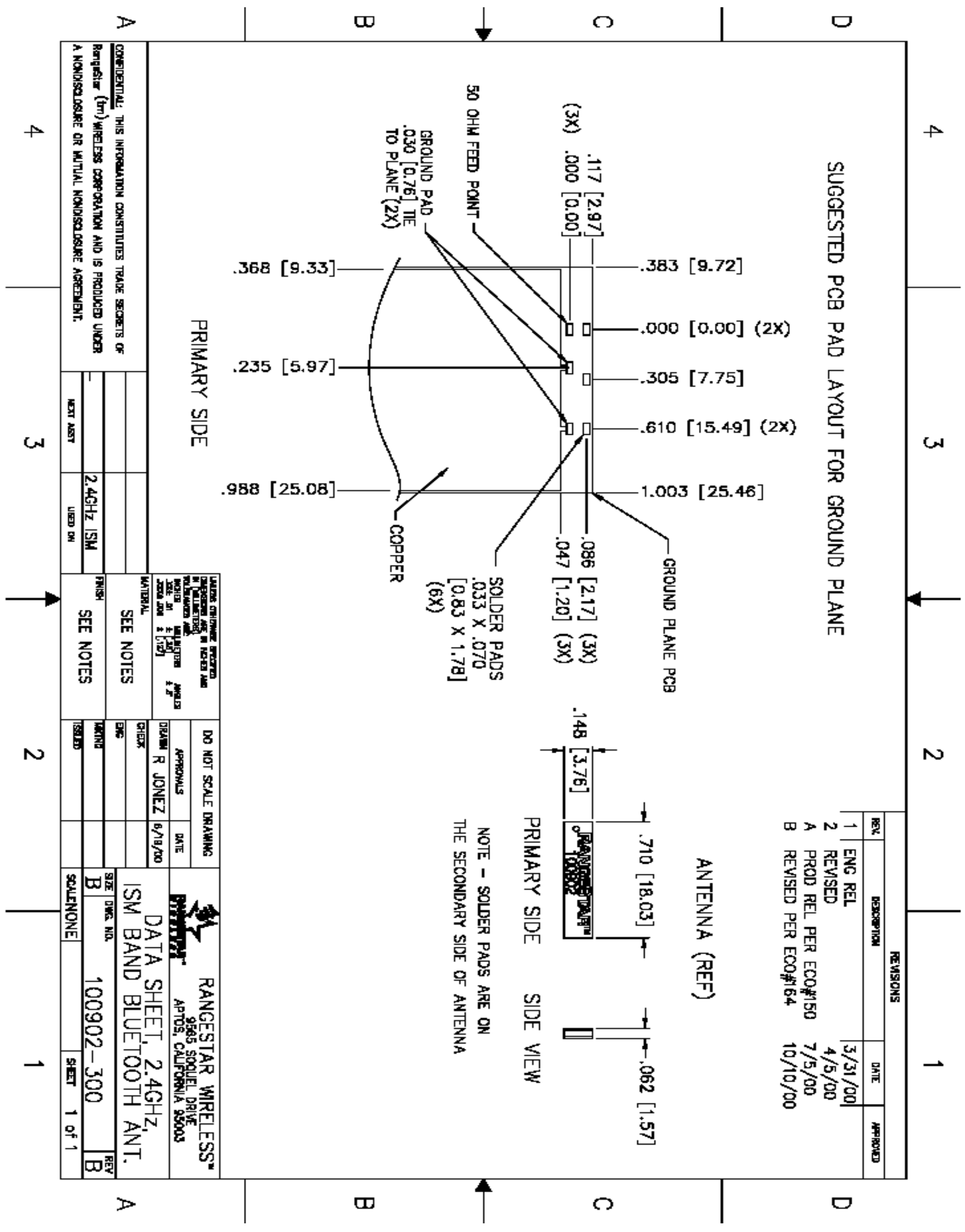
Parameter	Condition	Symbol	Min	Typ	Max	Unit
Power Supply						
Supply Voltage		V_{CC}	3.175	3.3	5.25	V
I/O Ports Supply Voltage	See note 10	V_{CC_IO}	2.7	3.3	3.6	V
Digital Inputs						
Logical Input High	Except ON signal	V_{IH1}	0.7 x V_{CC_IO}		V_{CC_IO}	V
Logical Input Low	Except ON signal	V_{IL2}	0		0.3 x V_{CC_IO}	V
Logical Input High	ON signal only	V_{IH2}	2.0		V_{CC}	V
Logical Input Low	ON signal only	V_{IL2}	0		0.4	V
Digital Outputs						
Logical Output High		V_{OH}	0.9 x V_{CC_IO}		V_{CC_IO}	V
Logical Output Low		V_{OL}	0		0.1 x V_{CC_IO}	V

Mechanical Specification



Appendix F Pad Recommendation for RangeStar 100902 Antenna

Source: RangeStar Antenna Specifications for the Bluetooth Developer



SUGGESTED PCB PAD LAYOUT FOR GROUND PLANE
PRIMARY SIDE
50 OHM FEED POINT
GROUND PAD .030 [0.76] TIE TO PLANE (2X)
.117 [2.97]
(3X) .000 [0.00]
.383 [9.72]
.000 [0.00] (2X)
.305 [7.75]
.610 [15.49] (2X)
1.003 [25.46]
.368 [9.33]
.235 [5.97]
.988 [25.08]
COPPER
GROUND PLANE PCB
.086 [2.17] (3X)
.047 [1.20] (3X)
SOLDER PADS .033 X .070 [0.83 X 1.78] (6X)
ANTENNA (REF)
.148 [3.76]
.710 [18.03]
.062 [1.57]
PRIMARY SIDE
SIDE VIEW
NOTE - SOLDER PADS ARE ON THE SECONDARY SIDE OF ANTENNA
REVISIONS
1 ENG REL 3/31/00
2 REVISED 4/5/00
A PROD REL PER ECO#150 7/5/00
B REVISED PER ECO#164 10/10/00
CONFIDENTIAL: THIS INFORMATION CONSTITUTES TRADE SECRETS OF RangeStar (tm) WIRELESS CORPORATION AND IS PRODUCED UNDER A NONDISCLOSURE OR MUTUAL NONDISCLOSURE AGREEMENT.
2.4GHz ISM
SEE NOTES
SEE NOTES
DO NOT SCALE DRAWING
DRAWN R JONEZ 6/19/00
RANGESTAR WIRELESS™ 9565 SOQUEL DRIVE APTOS, CALIFORNIA 95003
DATA SHEET, 2.4GHz, ISM BAND BLUETOOTH ANT.
B 100902-300 B
SCALE NONE SHEET 1 of 1

Appendix G BT Carrier Socket Specifications

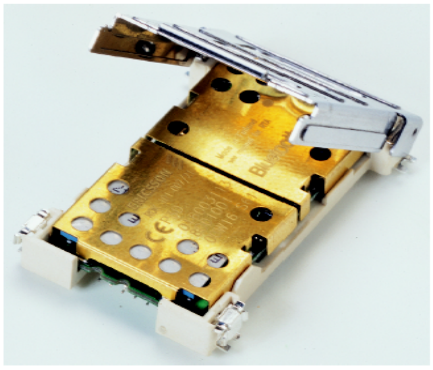
Source: Bluetooth Carrier Socket. http://www.suyin.de

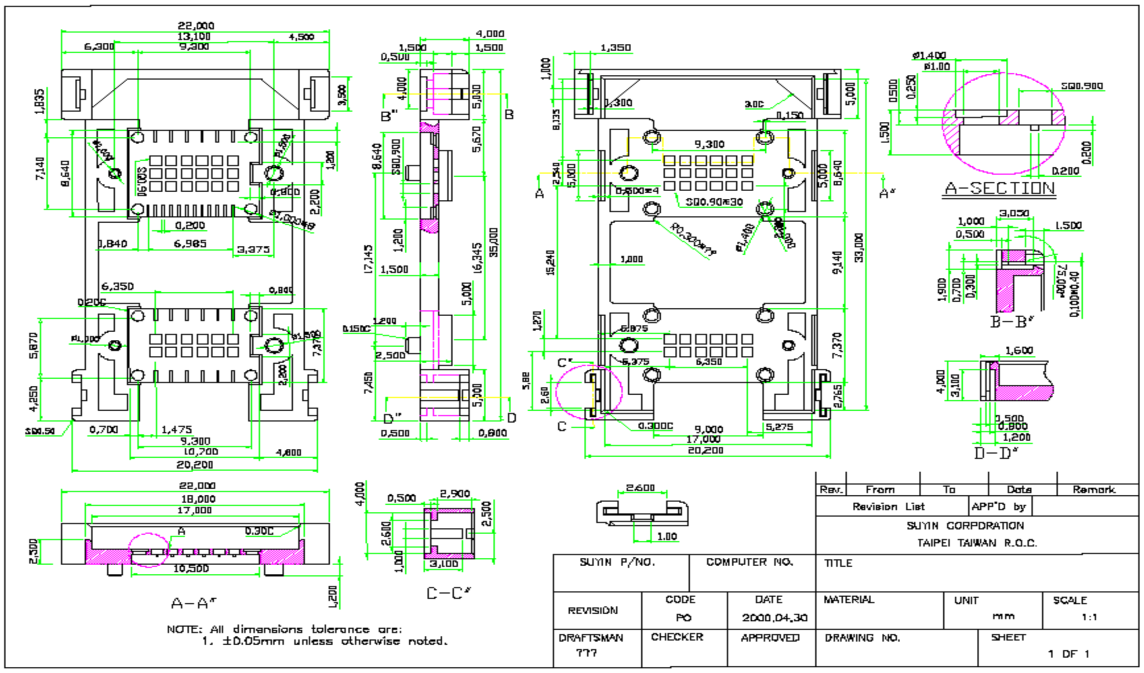
SUYIN CONNECTOR

TÜV ISO 9002

BLUETOOTH CARRIER SOCKET
FOR ERICSSON MODUL ROK101007

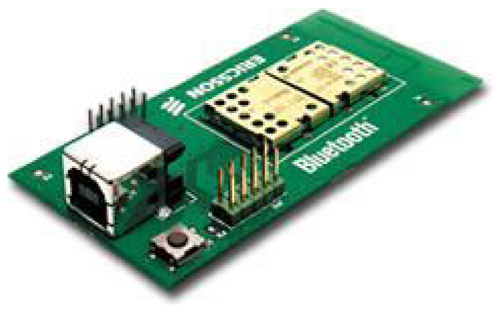
P/N	**: BT001A-30G2T**
Specifications	**:**
· PIN Number	**: 30**
· Insulator Material	**: LCP UL94V - White**
· Terminal Material	**: Pho. Bronze**
· Plating	**: sel. Gold over Nickel**
· Operating Temperature	**: -40° to +105°C**





Appendix H Overview of Ericsson BT Toolkit

Source: CD for Bluetooth Application & Training Tool Kit



As illustrated in the above figure, Ericsson Bluetooth Toolkit mainly consists of a two-layer printed circuit board, a UART buffer, a voltage regulator and the Bluetooth Module of Ericsson. The Ericsson Radio Module device, the Ericsson Baseband device and a Flash Memory are included in this Bluetooth Module. Figure 2 shows the location of main components on the Tool Kit board, which includes a jumper area to make easy access possible to certain signals.

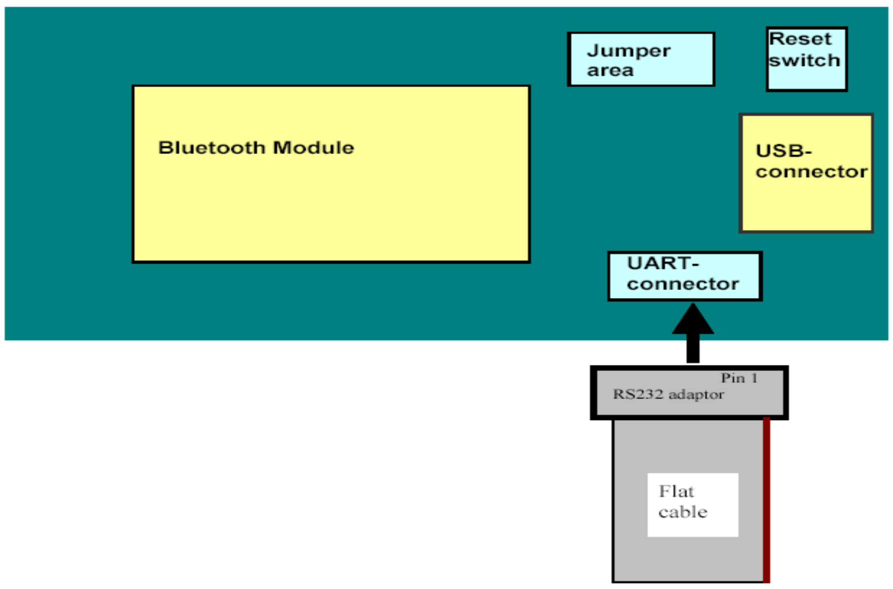


Figure 2: The Tool Kit circuit board

Appendix I TINI Specifications

Source: Tini specification and Developer's Guide. http://www.maxim-ic.com/TINIplatform.cfm

TINI Runtime Environment

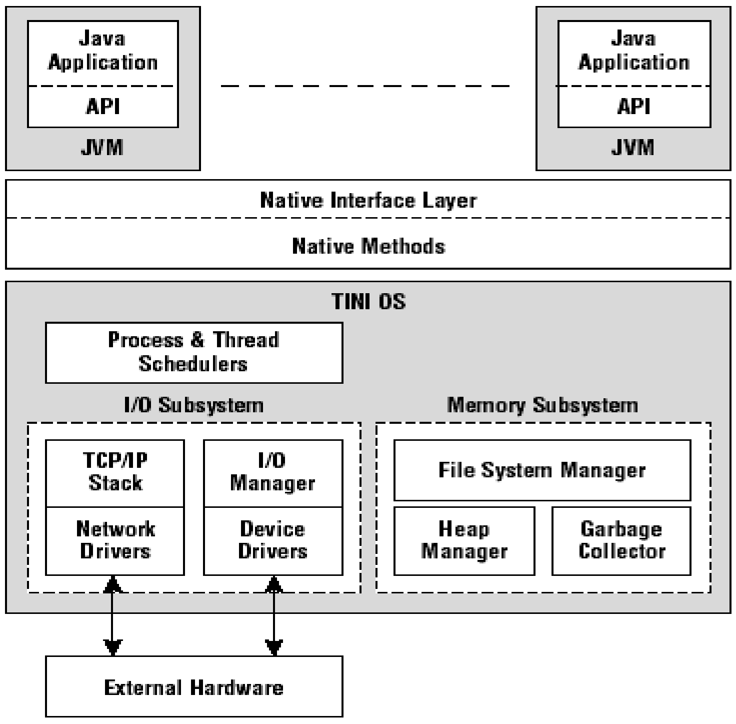


Figure 1.5 The TINI runtime environment

Figure I.1 the TINI runtime environment

Developed by Dallas Semiconductor, Tiny InterNet Interface (TINI) is platform with a Java programmable runtime environment illustrated in figure I.1., which makes a wide variety of hardware devices easy to connect directly to home and corporate networks.

The TINI Board Model 390 (TBM390), used in this project, is a compact 72-pin SIMM board, only 31.8 mm × 102.9 mm, which is illustrated in the figure I.2.

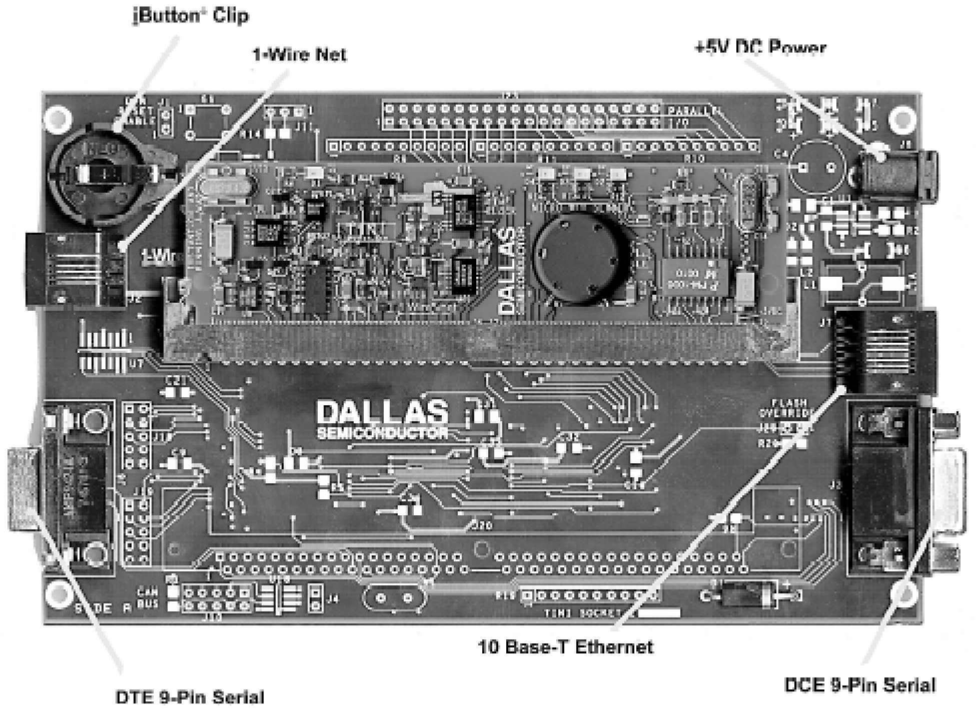


Figure I.2 the E10 socket with TINI board

As shown in the figure, TINI board mainly consists of DC power port, 10 Base-T Ethernet, DCE 9-Pin Serial port, DTE 9-pin serial port, dual 1-wire net interfaces, Dual Controller Area Network (CAN) and the DS80C390 microcontroller.

Appendix J Software Installed

The following software has been used in this project:

- Tini 1.02b
- TiniHttpServer 0.16
- Ant
- Tiniant
- Java commAPI
- J2SDK
- Java2 runtime Environment
- Microchip MPLAB IDE
- HI-TECH software
- Microsoft visual C++
- Bootloader hex files to burn into PIC
- PICdownloader