

**DEPARTMENT OF COMPUTER ENGINEERING
SCHOOL OF ELECTRICAL ENGINEERING
UNIVERSITY OF BELGRADE**

AD-HOC MULTIHOP SENSOR NETWORK

**GVOZDEN MARINKOVIĆ (TEAM MENTOR)
ALEKSANDAR BERIĆ
ALEKSANDAR RADOVANOVIĆ
BRANISLAV ČUKANOVIĆ
NENAD TEŠOVIĆ
NIKOLA MILANOVIĆ**

**COMPUTER SOCIETY INTERNATIONAL DESIGN COMPETITION
MAY 2001**

1. Abstract

In this report, we will describe a new application of mobile communications – a wireless, sensor, multihop ad-hoc network for data acquisition and remote administration.

The goal of this project is to create a complete hardware/software specification for replacing and/or upgrading the existing wire systems for data acquisition and process control. The idea is based upon the expectation that Bluetooth modules, due to their performance and price, will be the standard parts of most electronic systems: from mobile phones, personal digital assistants and industry control units to home appliances. However, because Bluetooth devices are short-range radio devices and, as such, are not suitable for large-scale networks, a multihop ad-hoc routing protocol is required to extend the range of a Bluetooth device by enabling routing of messages through intermediate mobile Bluetooth nodes, thus creating a multihop mobile network. Therefore, we started with the following intention: to connect a Bluetooth module with a microcontroller that will execute the routing protocol, forming a universal base for the wireless ad-hoc communication. The final goal is the integration of the routing protocol on the base Bluetooth chip, which eliminates the need for additional microcontrollers. Every device equipped with a Bluetooth chip would then be a potential node in a global ad-hoc network.

The basic advantages of wireless ad-hoc systems are: easy installation and upgrade, modest requirements for existing infrastructure, low cost and maintenance, great flexibility. Data acquisition systems are only one way to use such an environment. The project presents a universal platform for wireless integration of any system that requires remote administration and generates analogue and/or digital signals. The implementation of this project in such a system automatically guarantees two premises: stable and universal hardware platform, and reliable and easy replaceable software. The system that is once installed has the capability of dynamic reconfiguration and adaptation to the new working requirements and conditions.

2. System Overview

In this project an open data acquisition and processing system based on a wireless ad-hoc multihop sensor network was designed. Network was realized using Bluetooth modules with routing protocol implemented on a separate hardware module (Interface and Routing Module – IFRM) based on Microchip's microcontroller PIC17C756A. In order to emphasize open architecture of our system, a hardware Digital Signal Processing System (DSPS) based on a Texas Instruments TMS320LF2407 Digital Signal Processor was designed as a sensor example, and a Personal Digital Assistant (PDA), which enables fast on-line control of any process. The process control is done by the software for data acquisition (Shell) and the expert system for decision-making. All relevant system activities are stored in an Internet accessible database. The system overview and the cost of the additional hardware are shown on figures 2a and 2b, respectively.

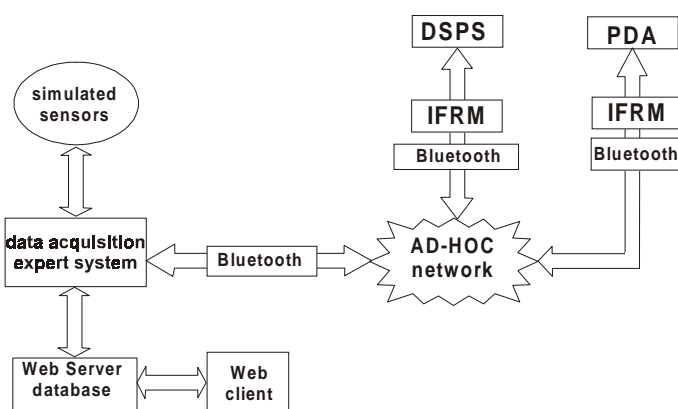


Figure 2a: System Overview

Part Type	Pcs	Unit price (USD)	Price (USD)
PIC17C756-33L (OTP)	2	12.38	24.76
TMS320LF2407PGE	1	11.09	11.09
74HC573	4	0.30	1.20
28F010	4	3.74	14.96
6264	4	2.70	10.80
Quartz osc. 14.7456 MHz	2	2.09	4.18
Crystal 6MHz	1	0.55	0.55
7805	3	0.32	0.96
LM317	1	0.53	0.53
MAX3225CPP	4	2.86	11.44
PCA82C250	1	2.62	2.62
LCD display LM020L	1	12.97	12.97
Keyboard	1	5.17	5.17
Connectors, headers, sockets	42		20.66
Passive components	157		12.13
PCB	3		27.33
TOTAL			161.35

Figure 2b: Bill of material

Multihop ad-hoc sensor network and IFRM: The basic idea of an ad-hoc network is establishing communication without a predefined network infrastructure. In other words, routing is dynamic, and data packets are hopping through other nodes that act as switches. The role of the network subsystem is to provide a communication between the other components of a system (server, DSPS, PDA...). In order to be scalable and flexible, the

network was designed as a multihop ad-hoc wireless network, which means that nodes perform routing of packets. The appropriate routing protocol was designed and optimized for execution on a microcontroller with very limited resources. In order to enable integration of existing data acquisition devices into the system, the interface to the Bluetooth module was designed to give such devices an illusion that they are in direct cable connection with the server.

Personal digital assistant: The role of the PDA is to inform the person in charge about the status of the process he/she is monitoring. Also, PDA enables a person to send the commands to the remote parts of the system. Communication with the rest of the system is realized through the interface that is universal for each component that has the access to the ad-hoc network.

Digital Signal Processing System (DSPS): A device with a TI DSP chip for digital motor control and signal processing was designed as an example of the sensor that can be connected to this system. The basic role of this device is gathering of information from different peripheral devices, real time processing of gathered data and transfer of processed data to the server (through the IFRM and the ad-hoc network), where the monitoring and additional processing are available.

Data acquisition and the expert system: The software for data acquisition and the expert system for decision making reside on the server. If any irregularity is discovered, expert system tries to solve the problem by sending control commands to the part of the system that caused the problem. If the system is unable to fix the problem, the person in charge is informed. If no response is received, a backup messaging system is activated. Every activity of the system is also logged in a database.

Database: All relevant activities of the system are recorded in a database: received sensor data, discovered irregularities and emergencies. The database is accessible from the

Internet. The idea is to provide remote diagnostics and administration, beside plain recording of the data, when the expert on site is unable to solve the problem on his own.

The basic advantage of this idea is that it offers universal and open platform, which can be implemented in any environment with small adjustments. For example, using this low cost kit, it is possible to upgrade factory sensor infrastructure. The existing wire sensors can be easily incorporated into the network, while it is possible to add any number of wireless sensors for each production process that is being monitored. Engineer in charge would be equipped with a PDA, which shortens the response time for any anomaly discovered. We can identify several possible scenarios. Expert system discovers some irregularity in the incoming data flow from the sensors and tries to remedy the situation. At the same time the message is being sent to the person in charge, who can manually override the expert system and send the remote command from the PDA. In that way, regulation of the anomaly has begun before the expert has made his way to the site that caused the problem. If the person in charge cannot contain the situation alone, another expert can access the database from the Internet, gain insight into the nature of the problem, and help expert on site to solve it. It is possible to picture numerous similar situations in various institutions, from hospitals and power plants to the applications in the open air – rescue actions and researches of inhospitable terrain.

The main problem that we face when trying to realize such a system is providing critical data transmission rate and stable ad-hoc network. The first problem was solved using the Bluetooth modules, which ensure fast enough transmission. The question of stability was addressed with careful design and implementation of the ad-hoc routing protocol. Testing and detail simulations, the results of which are presented in this paper, showed that this protocol, with realized hardware and software infrastructure, is capable of operating in extreme load conditions, which is a prerequisite for eventual industrial exploitation.

3.Implementation

3.1.Ad-hoc multihop routing protocol

Regarding the critical application of the system and the limited resources of the microcontroller on which it's being executed, the routing protocol was designed with the following guidelines in mind: speed, reliability and simplicity. As we will see, solutions are proposed for each of these items, based on the results of intensive simulations.

3.1.1.Basic problems and existing solutions

The ad-hoc networks are the next step in the evolution of the existing networks. Their main characteristic is the establishment of communication between mobile hosts without the necessity of the existing network infrastructure. Today's ad-hoc networks are mainly what we call the **single hop** networks – that is, the networks with base stations (fixed hosts) that act like a bridge between the classic wire network and the wireless network. In such networks there is no direct communication between two mobile hosts. Our idea is the development of a **multihop** ad-hoc network, where mobile nodes act like switches during the transport. Therefore, it is clear that every host must have routing capabilities, that is, every mobile host is also a router at the same time.

The main problem regarding the ad-hoc networks is the routing of the packets. We started the development of the protocol by examining the existing solutions. We examined the following protocols: Dynamic Source Routing (DSR), Ad-hoc On-Demand Distant Vector Routing (AODV) and Temporally-Oriented Routing Algorithm (TORA). It turned out that DSR and TORA failed to fulfill our requirements. Namely, DSR broadcasts route request to all nearby nodes, the request then propagates through the network and gathers the addresses of all the nodes that it passes through. The drawback is obvious: in networks with many nodes, route requests become too big and the network is overloaded. The big limitation of TORA is the demand for some external timing system (such as GPS). AODV algorithm

offers an acceptable solution, although it doesn't use all the benefits of the Bluetooth technology. Therefore, we adopted it as a starting point and modified it in shape of Bluetooth capabilities and the specific attributes of the sensor networks.

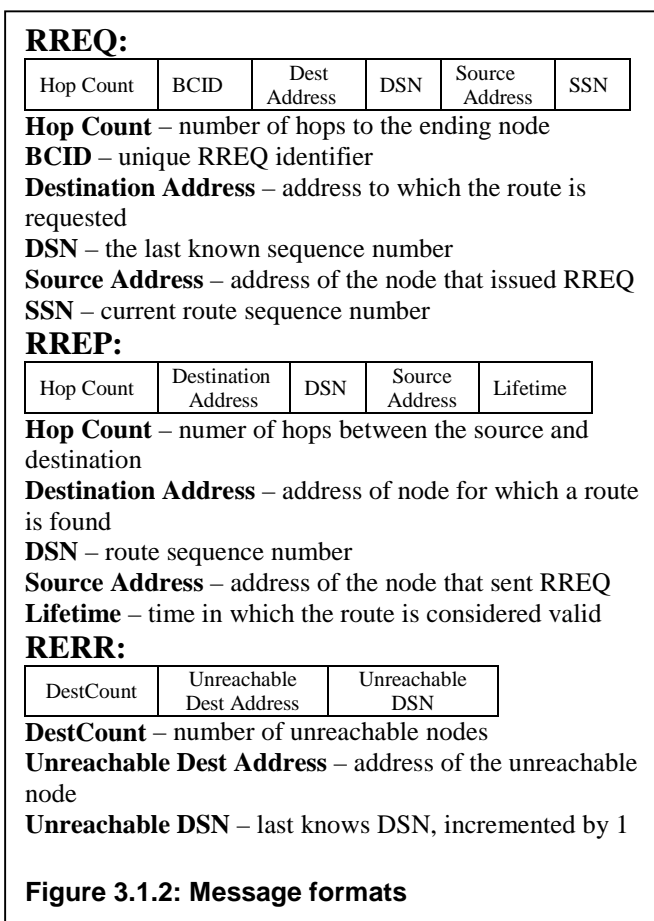
3.1.2.Message formats

The routing algorithm defines three types of messages: route requests (RREQ), route replies (RREP) and route errors (RERR). The formats are given in the following figure:

3.1.3.Functioning of the protocol

The algorithm can distinguish between three types of nodes: master, gateway and plain node. It uses the inherent capabilities of the Bluetooth modules to form piconets with master-slave communication. The nodes that are members of two or more piconets are considered the

gateway nodes. All communication is done exclusively on the master-slave level. The plain node sends packets to the master, while master then forwards them to the appropriate gateway. This kind of forwarding is continued until the packet reaches the piconet with destination node. Server (the node on which the expert system resides) communicates with all other nodes (sensors, PDA), while other nodes send data to the server only (which is inherent to the sensor networks – the protocol itself is not limited to this kind of



communication). Let us observe the following situation in the network:

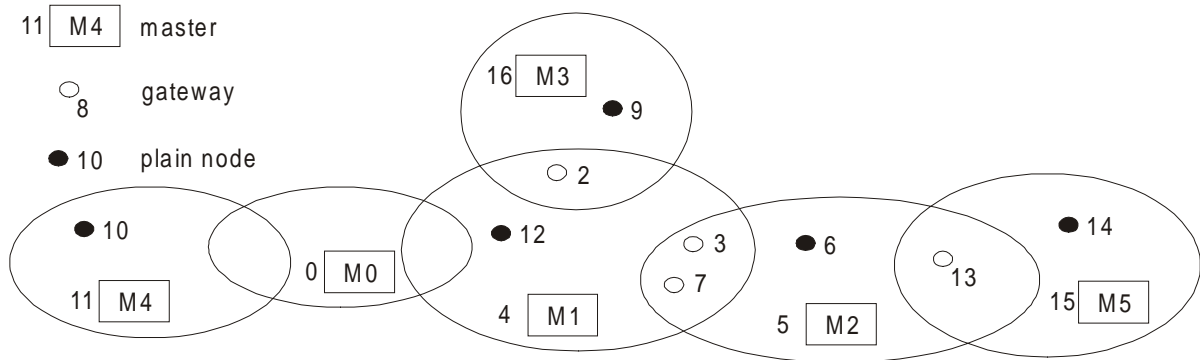


Figure 3.1.3a: Possible network topology

The algorithm works in the following manner: during system initialization the piconets are formed, every node creates the neighbor table (nodes in the same piconet) and the empty routing table.

When a node wants to send a message to other node, it first searches the neighbor table to see whether the destination node is in the same piconet. If so, the communication is performed

Dest Address	DSN	Hop Count	Last Hop Count	Next Hop	Precursors	Lifetime
Dest Address – address of the destination node						
DSN – destination sequence number						
Hop Count – number of hops to destination						
Last Hop Count – hop count before route invalidation						
Precursors – list of forwarding nodes						
Lifetime – time for which route is valid						

Figure 3.1.3b: Routing table entry

instantly, through the master of that piconet. If the destination node is not in the neighbor table, the node searches the routing table. If there exists an active

route to the destination node, the packet is forwarded to the node specified in the Next Hop field of the corresponding routing table entry. If there is no valid entry in the routing table, the node sends route request packet (RREQ) which propagates through the network, but not as a plain broadcast – all subsequent communication is done at the master-gateway-master level. There is no need for master to broadcasts RREQ to its slave nodes, because it has all the necessary information about them in the neighbor table. When the RREQ reaches the destination node (or the node that has the valid route to destination node), the route reply (RREP) message is generated, which travels back to the node that issued the RREQ. At the same time, all nodes that were in the path of the RREQ and RREP messages update their routing tables. When sending RREQ, the node fills the DSN field with the last known value

for the desired destination (initially zero). If during the communication there happens to exist a routing table entry with a DSN value that is greater than this, that means that the route is invalid.

When a node receives RREQ, it checks whether there is an active route to destination. If such route does not exist, RREQ is resent, while the DSN value is set to the maximum of DSN from RREQ and DSN from the routing table entry (if exists). Otherwise, the RREP is generated.

The node sends route error (RERR) message in three cases: (1) the link is broken, (2) the packet is received, but there is no active route to the destination, (3) RERR is received from a neighboring node. Cases (1) and (2) are handled by incrementing the corresponding DSN entry. This solves the problem of dynamic changes of the network topology. Then RERR is forwarded with the list of unreachable nodes, and the new DSN value. In case (3), for each unreachable node from RERR, the check is performed to see whether the source node (Source Address) is the Next Hop in any of the routes to the unreachable node. If so, the following procedure is required: the DSN value is updated, Hop Count is set to infinity (thus invalidating the route entry), the old value is copied to Last Hop Count, the list of the precursors to this destination is checked, and if not empty, RERR is forwarded again. After that, the list of precursors is deleted.

3.1.4.Simulation

For the purpose of testing, before the hardware implementation of the protocol (discussed in the next chapter), a simulator was written in Java. The starting network topology can be specified in the configuration file. The number of nodes is not limited. For each node, a separate thread of execution is created. One node is declared as a server (on which the expert system resides), and afterwards the messages and the destinations are being generated in a random fashion. Also, some nodes traverse from one piconet to another,

simulating the movement of a person carrying the PDA (or any other mobile nodes). Since the routing tables are empty at the beginning of the simulation, the processes of route discovery and routing table filling can be easily monitored.

The analysis of the protocol can be achieved by entering the configuration from the figure 3.1.3. into the simulator and following the route discovery process. Assume that node M0 (server) wants to send a message to node 6. A server searches the neighbor table and concludes that node 6 is not in the same piconet. Then it searches the routing table. Since this is the beginning of the simulation, the routing table is empty, so there is no entry for node 6. Therefore, node M0 sends RREQ to all gateway nodes it can reach. Since there was no previous route to node 6, the value of DSN field is set to zero. The gateway nodes 8 and 1 receive the RREQ. The subsequent process of route discovery is shown on the following two figures:

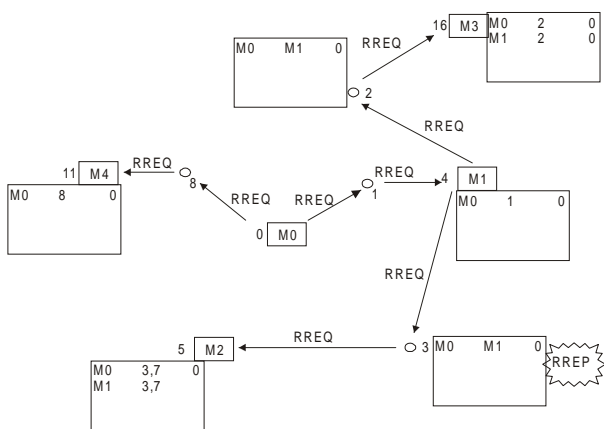


Figure 3.1.4a: Route discovery (RREQ forwarding)

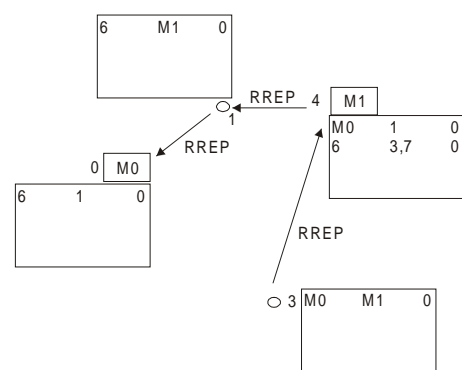


Figure 3.1.4b: Route discovery (RREP forwarding)

When the process of route discovery is over, node M0 can send a message to node 6 over newly established route through node 1.

3.1.5. Usability analysis

The basic thing that should be noted is how a pair of RREQ / RREP messages can induce the creation of many routing table entries in all the nodes through which these messages have passed. In this case, the single route request has caused creation of 11 new

entries in the routing tables of the intermediate nodes, without broadcasting the RREQ. This is a clear advantage compared with various broadcast algorithms. This advantage is accomplished by simultaneous maintenance of neighbor tables, so there is no need to broadcast the route request within the piconet. The known network topology thereafter remains the same, while the number of connections lowers proportional to the number of piconets and nodes within. On this level, it is not possible to establish analytical model that can show the quantitative advantage of this approach, since we do not have the data about the overhead that is necessary for the maintenance of the piconets, that is, neighbor tables. However, if we accept the number of established connections during the route discovery as a parameter, we obtain the following result [fig 3.1.5a]:

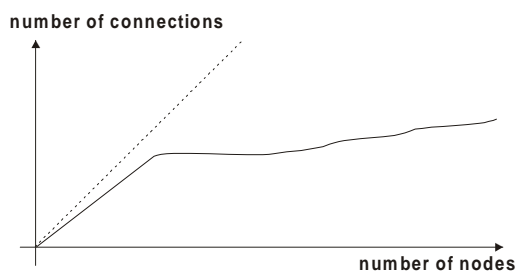


Figure 3.1.5a: Average number of connections needed for a route discovery

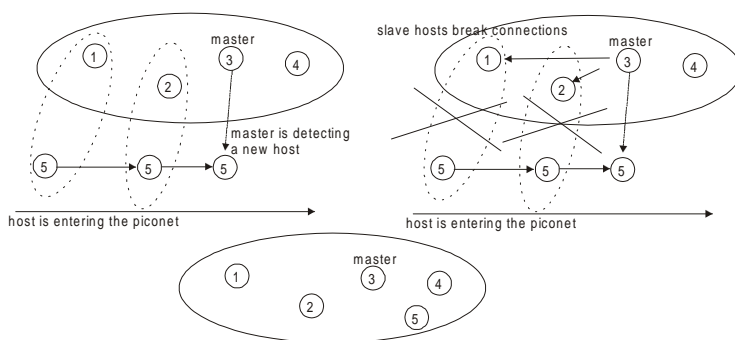
It is clear that when there is only small number of nodes in a network, the proposed mechanism resembles the classical broadcast, since the majority of the piconets are, in fact, 1-1 connections. However, as the number of nodes increases (that is, the piconets become

more and more populated), the algorithm shows its advantages, since the average number of connections needed for route discovery start to rise more slowly.

Once again, we must state that these results do not originate from 100% reliable model, because the model assumes that the time needed for maintaining the piconets can be considered very small compared to the time needed for message delivery. We hope that we'll soon have relevant data, based on which we'll be able to simulate the process of piconet maintenance.

Regardless of what has been said in the previous paragraph, a technique is introduced in the algorithm that optimizes the entrance of a new node in the piconet. This optimization is

based on the elimination of redundant piconets. Why is that important? First, the reduction of redundant piconets automatically means that there are fewer connections in the route discovery process. Second, Bluetooth uses frequency hopping spread spectrum technology, and with the increase of the number of connections, the probability of collision on the channel also increases, which degrades performance. The general idea is: when a new node is entering the piconet, the first node that detects it opens a new connection (piconet) to the incoming node. The same process continues until the master of the piconet detects the new node. The master then updates its neighbor table, broadcasts the information that the new node has entered the piconet to all other nodes in the original piconet. Upon receiving this



broadcast, all slave nodes that have established the connection to new node break these connections. In that way, all redundant piconets have been eliminated.

Figure 3.1.5b: Elimination of redundant piconets

The final problem that has

to be addressed in this part is synchronization. Artificial synchronization is introduced in the simulator, using the Java multithreading synchronization, which eliminates the potential collisions. For now, there is no way to discover how good the synchronization of the Bluetooth modules works in real world applications, that is, how many piconets can be packed in a confined space. That remains one of the guidelines for further study.

3.1.6. Ending considerations

Further development of the simulator is planned to incorporate existing proposals for other multihop ad-hoc routing protocols (DSR, AODV, TORA, LANDMARK, etc.). That would enable comparative analysis that could lead to the refinement of the proposed protocol.

Because our team received the Bluetooth modules with a huge delay (March 2001 instead of December 2000), the work on the protocol and the routing hardware had to be based on public documentation available on the Web [1]. Unfortunately, received Bluetooth modules support only point-to-point communication, which is not compliant with the mentioned official documentation. However, the hardware requires, and was designed to support point-to-multipoint communication. The mentioned limitation means that we weren't able to test the protocol in real world conditions. This simulator proves that the proposed protocol is valid, since we can only establish simple 1-1 connections with the received Bluetooth modules. According to Ericsson, point-to-multipoint communication is expected in the next generation of Bluetooth, and we are hoping that we'll soon be able to demonstrate the full power of designed hardware.

3.2. Hardware Implementation of the Mobile Ad-hoc Multihop Network

In order to provide the exchange of information between mobile components of the system, it is necessary to design a fast and reliable mobile ad-hoc network, capable of quickly adapting to topology changes. Considering the limited range of the Bluetooth modules, this network must be realized as multihop ad-hoc network, that is, every node must be capable of forwarding messages. Therefore, it is necessary to implement an adequate routing protocol. It would be ideal if this protocol could be implemented in the baseband controller of the Bluetooth modules, but since we didn't have direct program access to the baseband microprocessor, the routing protocol was implemented in a separate hardware module that communicates with the Bluetooth module through a serial (UART) connection, on the HCI level.

3.2.1. Interface and Routing Module (IFRM)

The main functions of the IFRM are: forming of a wireless multihop ad-hoc network, link maintenance, packet routing and route discovery; providing a transparent interface between

the ad-hoc network and any serial (RS-232) device, in a way that a device has an illusion that it is communicating with the rest of the system over a cable connection; providing PDA functionality

The core of the IFRM is Microchip’s PIC17C756A micro-controller, chosen for its fast RISC architecture and because it has all the necessary peripherals integrated on chip – two high speed (1 Mbps) universal synchronous / asynchronous receiver / transmitter (USART) modules, WatchDog timer, In-Circuit serial programming capability and 32 Kwords on chip ROM.

For the purpose of storing the routing tables and FIFO buffers, 16 KB of external RAM was added with two static 6264 RAMs. Also, there are 32Kwords of external FLASH ROM, added to facilitate the development of the software. In the final product, this memory won’t be needed, because the whole program will be stored in on-chip ROM.

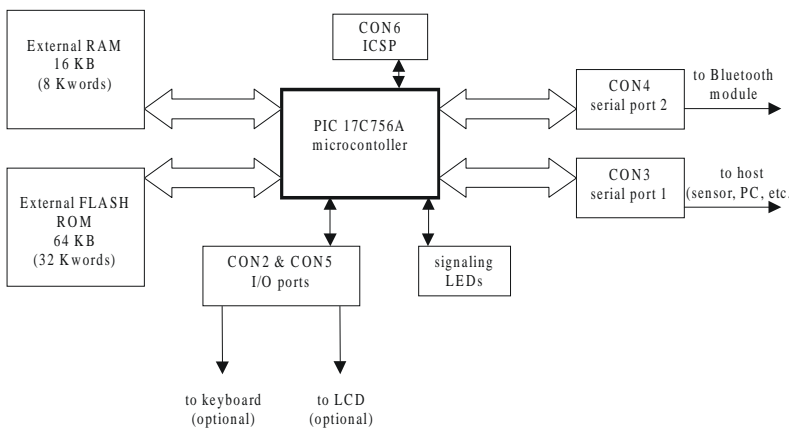


Figure 3.2.1: Interface and routing module

IFRM communicates with the rest of the system through two high-speed serial ports (CON3 and CON4). CON4 is used for the communication with the Bluetooth module, while CON3 is used for the

communication with the host (any device equipped with a RS-232 port, e.g. sensors, PC, EEG, etc.). Both serial ports support hardware (RTS/CTS) flow control, and have high-speed RS-232 MAX3225 drivers, that enable transfer rate up to 1Mbps. IFRM also has two general-purpose 8-bit I/O ports (CON2 and CON5), that are used to set the configuration parameters (using jumpers) or for connecting the LCD or keyboard (when working as PDA).

3.2.2. Routing protocol – host side

The modified AODV protocol (described in section 3.1) is implemented on the IFRM, with the following restrictions:

1. The size of the routing tables is limited to 79 entries (memory considerations)
2. Communication in the network is server centric (star topology with server in the center), i.e. the communication is possible from server to any host (sensor, PDA), and from any host to the server, but end-to-end communication between the hosts is not possible (apart from the case of packet forwarding – Figure 3.2.2.).
3. End-to-end flow control is omitted (there is only point-to-point control on the baseband Bluetooth level)
4. CRC checking and retransmission are not implemented (it is assumed that Bluetooth enables reliable transmission)
5. Packet sequence is not checked (we assumed that packets arrive in the original order)
6. Restrictions due to the fact that delivered Bluetooth modules do not conform to the Bluetooth V1.0B specification

The reason for restriction (2) is to provide network transparency for the hosts attached to IFRM. In that way, IFRM always forwards the data received from the host to the server. Therefore, this eliminates the need to have intimate knowledge of data sent from host, i.e. the destination address doesn't have to be determined from the received byte stream in order to forward the data – it is always sent to the server, which is assigned a unique address of all zeros. Because of this, it is possible to connect any serial device to the IFRM, regardless of the protocol used by that device. Serial byte stream received from the host is segmented into packets; each packet is assigned a destination address of all zeros and sent over the ad-hoc network to the server, which decides how to interpret the received byte stream at a higher level. In the opposite direction (server-to-host), destination address for a packet is implicitly

determined in software, again without intimate knowledge of the byte stream transmitted. It is very important to note that this is not the limitation of a protocol itself – each node stores destination to other nodes in its routing table, because it is necessary in the server-host direction. Considering that server can send a message to any host, intermediate nodes must know the route to that host, in order to forward packets to it. The only reason for imposing this restriction is to make routing independent of data being sent, i.e. to simplify the addressing scheme. If needed, the protocol can be easily reconfigured to support end-to-end

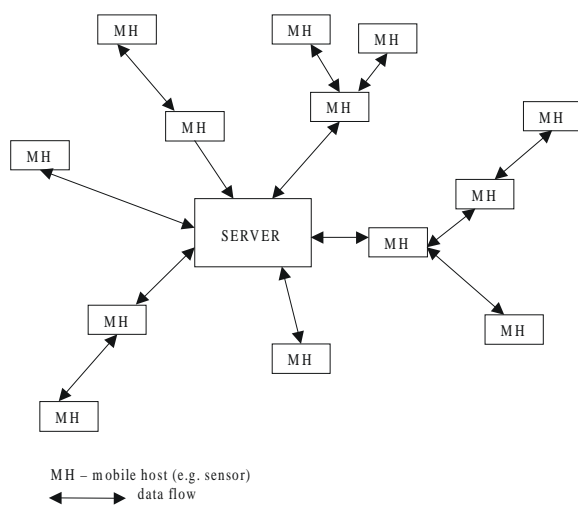


Figure 3.2.2a: Star network topology

communication between any given nodes, using a more complex addressing scheme (e.g. TCP/IP).

The reason for restrictions (6) is the fact that delivered Bluetooth modules are actually early development stage modules, and are not conformant to Bluetooth V1.0B specification. Most important of all, these

modules do not support point-to-multipoint connections that are essential to this project. Therefore, in the design of the router great care has been taken to ensure that it becomes fully operative in the moment when the Bluetooth modules which conform to the specification appear. Because point-to-multipoint capabilities of the Bluetooth couldn't be tested, the following assumptions are adopted:

1. In the course of full-duplex communication there is no implicit master-slave switch
2. It is possible to perform inquiry and paging of the new nodes during the active connection, without interfering with the current transmission
3. Only the master of the piconet can broadcast on the piconet
4. A node can transparently be a member in more piconets (scatternet), i.e. Bluetooth

baseband controller is capable of performing transparent time-division multiplexing

5. It is possible to communicate with more than 7 nodes in a piconet, i.e. parking and unparking of nodes is done transparently by the baseband controller

Due to the fact that several HCI commands that are very important for this project (e.g. Link Policy Commands) have not yet been implemented on the modules we received, and because of discovered bugs in the Bluetooth firmware, several workarounds have been implemented. For example, sending two consecutive broadcast packets causes a reset of the Bluetooth modules. Therefore, after every broadcast packet, one dummy point-to-point packet is sent, in order to avoid reset. Also, during maximum full-duplex transfer rate transmission, packets or parts of the packets disappear, so the attention has been paid to resynchronization of the byte stream after the missing bytes have been detected.

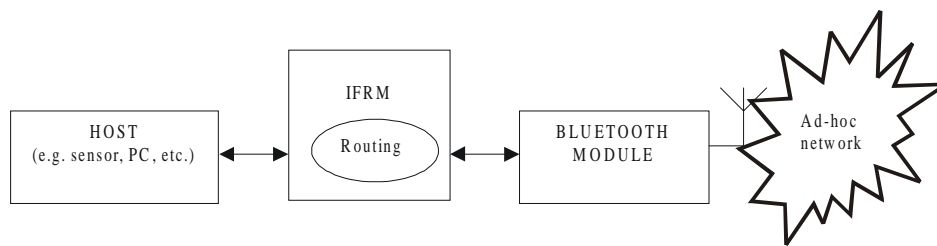


Figure 3.2.2b: Mobile host

3.2.3. Routing protocol – server side

The implementation of the routing protocol on the server side differs from the IFRM implementation, because on the server side there is no routing module. The software on the server (PC) performs the routing. A special mechanism for implicit discovery of the packet destination address is needed, since the address is not determined from the data. Also, received packets are not forwarded to RS-232 device (which is the case with IFRM); instead they are forwarded to the expert system. Therefore, it is necessary to provide a communication between the expert system and the ad-hoc network.

The chosen mechanisms are TCP ports. For each mobile host in the network there exists a corresponding TCP port on the server, starting from port 10000. In a network with 3

nodes, there would be 3 ports on the server – 10001 for the first host, 10002 for the second and 10003 for the third. All communication with the mobile hosts is performed over the TCP/IP connections on these ports. For example, the data flow at connection server:10003 is redirected over the ad-hoc network to mobile host number 3. The mapping between the port number and hardware address of the Bluetooth module on the mobile host is static, and is done via configuration file.

For example, when the packet from the expert system destined for node 3 arrives at port server:10003, routing protocol searches the configuration file for the hardware address of the module on host 3. The found address is put in the destination address field of the packet and the packet is then send over the ad-hoc network. Therefore, the destination

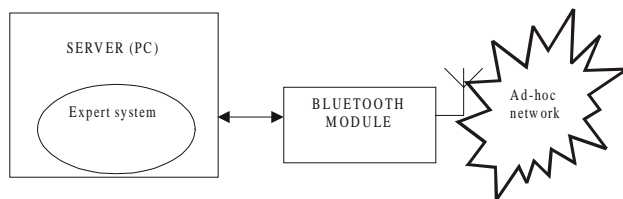


Figure 3.2.3: Expert system and server side

address is determined implicitly, based on the port number and the configuration file, again without the interpretation of the data to be sent.

For any application to communicate with any host in the mobile network, it is sufficient to open a TCP connection to the appropriate port on the server and all subsequent communication with the mobile host is done transparently, over the ad-hoc network.

3.3.PDA

The role of the PDA is to provide mobility to the expert, so he is free to conduct activities other than simple surveillance in front of the server. PDA is capable of receiving two types of messages:

1. Certain sensors have reported irregular data, but the expert system has managed to stabilize the system. The confirmation from the expert is required.
2. Certain sensors have reported irregular data and the expert system didn't manage to stabilize the system. The remote command and presence is required from the expert.

According to this, two types of messages can be sent from PDA:

1. Confirmation of received warning.
2. Command that initiates some action in the system, based on the received data

PDA is completely realized with the interfacing and routing module (IFRM). It differs from the sensor node in few things: keyboard with 16 keys is connected to the port CON2, while the 16-character LCD is connected to the port CON5. All remaining differences are in the software – beside the routing protocol, there is a part that sends received packets on screen (instead of forwarding them to the serial port), and the input data is received from the keyboard, not from the sensor.

In this phase of the project, the internal battery power supply is not designed.

3.4. Digital Signal Processing System (DSPS)

DSPS is realized in hardware, as an example of one type of device that can be connected to this environment. It is based on Texas Instruments TMS320LF2407 Digital Signal Processor (DSP), which belongs to the family of digital motor control and signal processing chips. The main role of this system is gathering of the information from different peripherals, real time processing and transferring to server over the interface module and ad-

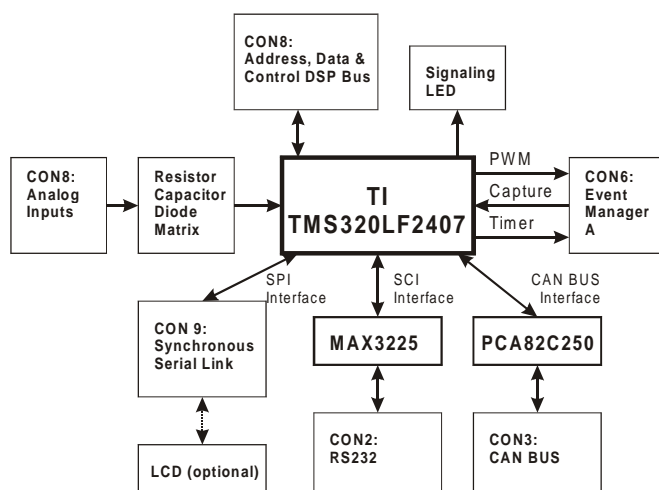


Figure 3.4: DSPS

hoc network. Additional processing and advanced monitoring are available on the server. The program code resides in the internal FLASH EEPROM of the DSP. It is used for decoding instructions that come from IFRM and for spectrum analysis using Fast Fourier Transformation (FFT) of analog inputs. The main parts of the DSPS shown on Figure 3.4. are:

- Texas Instruments DSP TMS320LF2407;
- RS232 driver (MAXIM's MAX3225cpp);
- Controller Area Network driver (PHILIPS's PCA82C250);
- Resistor-Capacitor-Diode Matrix;
- Signaling LED;
- Various connectors that enable link of the DSP with its environment:
 - External Memory Interface, i.e. DSP BUS (Address, Data and Control BUS);
 - Synchronous Serial Peripheral Interface (SPI), used for LCD in the future;
 - Serial Communication Interface (SCI), used for RS232 Interface;
 - Controller Area Network (CAN) Interface, important for industry support;
 - Event Manager A of the DSP.

The TMS320LF2407 is a fixed-point DSP with high-performance processing capabilities and integrated advanced peripherals such as: Event Managers A&B optimized for multiple motor and-or converter control and power conversion applications like centered- and/or edge-aligned PWM generation, 16-channel high performance (500 ns) 10-bit ADC, SPI, SCI, CAN. Here we used Event Manager A (PWM, Capture and Timer blocks), 8 analog channels, SPI, SCI and CAN interface. DSPS is realized in two layer PCB, so there were no routing possibilities for support of Event Manager B and other 8 AD channels. We also enabled External Memory Interface intended for future applications (various peripherals). DSP has on chip 32 K words of FLASH EEPROM and 2.5 K words of Data-Program RAM (544 words are Dual Access RAM), so external RAM and (E)eproms is not on-board supported. Important fact for us was that this DSP can be programmed, among other ways (i.e. JTAG), via asynchronous or synchronous serial interface using a PC as the host. Bootloader software (located in ROM section of DSP) is used to transfer compiled code from PC to the DSP's FLASH with the speed of 38400 baud. We found this way of programming ideal for

laboratory testing and prototype designs. The PCB is designed in such a way that changing the position of two jumpers enables the bootloader, otherwise, program starts its execution.

MAX3225cpp is responsible for the standard RS232 connection (via null modem cable) with PC (programming FLASH memory or testing) or with the interface module (integration in the Bluetooth environment). The industry support is provided using the Philips PCA82C250 CAN driver. DSP can also communicate with the outside world via synchronous serial connection (SPI interface). This can be used for future upgrades to attach LCD to the DSP, for the purpose of on-line monitoring. The analogue inputs to AD converters are

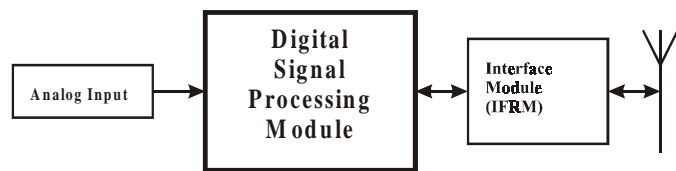


Figure 3.4b: Connection with the interface and routing module

protected with the diode matrix, and an adequate resistor-capacitor network is provided which ensures different voltage levels

that can be converted, along with the noise filtering.

The program code stored in the FLASH memory has two functions. First, it works as a message interpreter which analyses data received from the interface module. When a command (message) is decoded, the execution begins. The format of the sent message is the same as the format of the received, and both are conformant to the IEEE 999-1992 SCADA specification. Second aim of program is sampling input analog voltage, creating the packets of 2^n samples, and calculating the spectrum of the given signal using FFT. The calculated spectrum is stored in the on chip RAM and is constantly being refreshed. When the command for spectrum request arrives, stored spectrum information are transferred from RAM to the server.

Basicaly, the idea was to use one of the big advantages of digital signal processing - ease of implementation of the Fast Fourie Transform algorithm - to find the spectrum of some analog signal, to transfer this data to PC via the fully transparent Ad-hoc network (because

the DSP sees only RS232 interface); after that, the software on the server enables real time monitoring of the spectrum.

After analysing the received data, the expert system can return the command to the DSP what to do next, thus creating a positive feedback system. Here we showed just one example of many purposes DSPS can do. Capabilities of the this particular DSP, fast programming of its FLASH and design of the DSPS enable quick change of the purpose of whole system.

3.5. The data acquisition software and the expert system

The software part of the project unifies the components of the system. A multipurpose

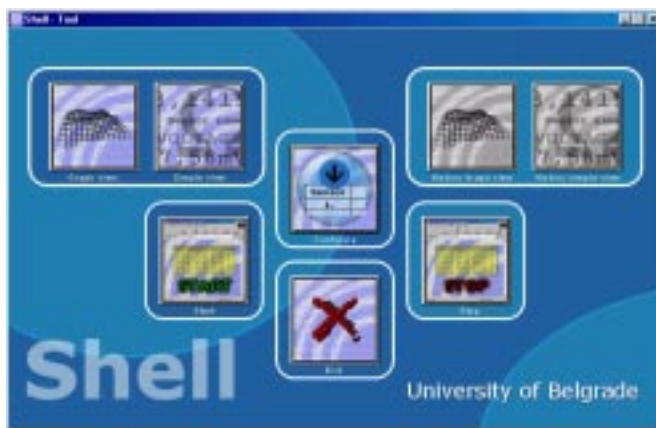


Figure 3.5: The main application screen

application (Shell) was developed for data acquisition, decision-making, signal processing, alarming, tracking the current state of the system and database administration. The basic role of this software is data acquisition from the sensors and measuring devices that are connected to the server over the ad-hoc network. In order to perform this function with maximum efficiency, a detailed setup of the sensor configuration is available. All parameters relevant to the identification and reading analysis can be input.

Real time tracking of the readings is available, and data can be presented in a graphic form. In that way, the time dependence between different readings can be easily determined.

All readings are recorded in a database, so the history analysis of the system is possible. One part of the system enables browsing and querying this database, so a detailed analysis can be made for any given moment in the past. The database is also accessible from the Internet. This feature opens a number of new possibilities, and offers many benefits that

will be discussed later in this paper.

The software was developed using MS Visual Studio and MS Visual C++. The capabilities of Microsoft Foundation Classes were used. The interface was designed to be user-friendly, and very complicated sensor configurations can be set up quickly. The testing and starting of the configuration is performed with a single mouse click.

3.5.1.The communication with sensors

The Bluetooth standard and the designed ad-hoc network provide a transparent system for the communication between the computer, sensors and other measuring devices. By opening a proper port, a socket is created over which the two-way communication with the device is performed. Socket communication uses TCP, and the final connection between the sensor and computer conforms to the IEEE 999-1992. SCADA specification.

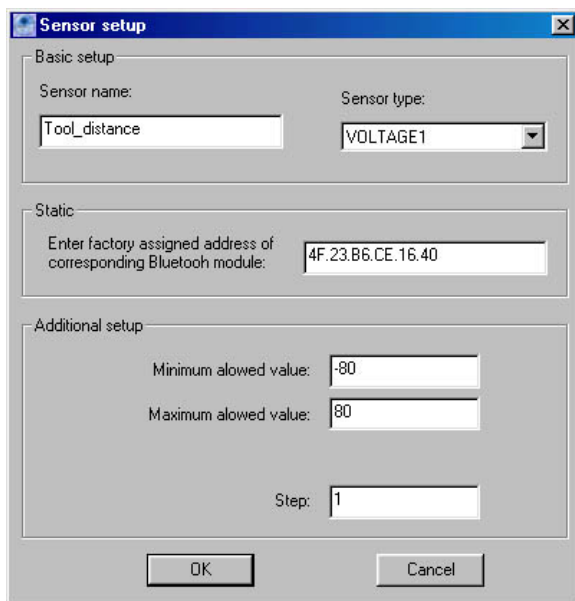


Figure 3.5.2a: Sensor setup dialog box

introduce a new device into the ad-hoc network, and it will automatically become accessible. Therefore, it can be concluded that the setup of a new device (including the physical mounting) can take less than a minute.

3.5.3.Readings display

After the system has been properly configured, the process of reading and logging can be initiated. Then, it's possible to monitor current values on the graph, if the time dependence

3.5.2.System configuration

Adding new devices into the system is very easy. The user clicks one button and opens Sensor setup dialog box, where he can define type and name of the sensor, address and the range of the expected values. Right after that, the system is ready to accept real-time readings from the added device.

The whole system is easily expandable. Beside the described procedure, it is sufficient to

is needed, or in the simple view with numerical values displayed.

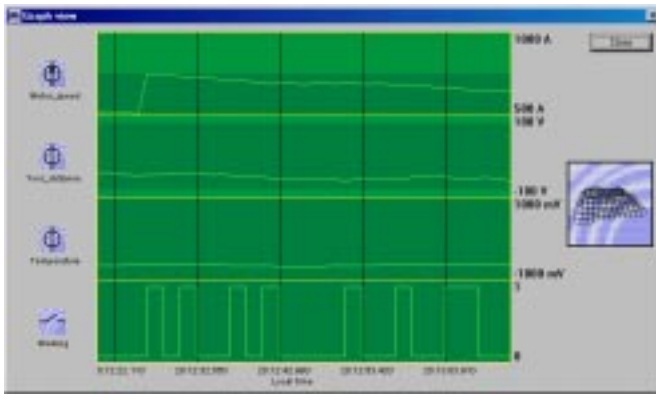


Figure 3.5.3a: Graph view

It is also possible to analyze the history of the system. Using simple queries, it's possible to obtain data from any given moment in the past. This information can also be viewed in mentioned modes (numeric and graphic) and the display can even be redirected to a Web page. If

graphical display is chosen, the slider for moving through time appears. The real-time graphic display is refreshed after each sensor reading. That means that real-time monitoring of time dependencies is possible, with the irregular values clearly marked. When the graph enters a

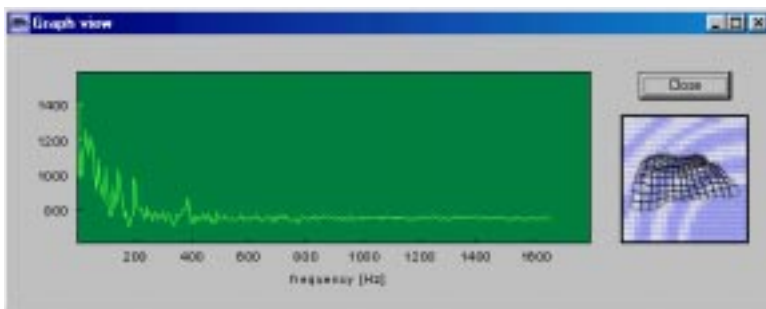


Figure 3.5.3b: Spectrum analysis received from DSPS

forbidden zone, an alarm is activated. Figure 3.5.3a shows readings from the simulated sensors, while figure 3.5.3b shows spectrum analysis received from DSPS.

3.5.4.Database

The database, in which values and times of readings for each sensor are recorded, is realized with MySQL Server. After creating, the database is application and platform independent, and can be easily accessed from the Internet.

The database administration is performed using ODBC with the standard classes from MFC collection. Thus, the system is flexible and independent of the DMBS used. In case of transferring to some other SQL Server, the system will create new database and tables, and continue working as if nothing has happened.

3.5.5. The expert system

While the readings are being recorded in the database, they are at the same time being analyzed by the expert system. Partly due to the limited time we had for completing the project, and partly because we wished to maintain the universal applicability of the whole system, the simple expert system was designed for monitoring (controlling) with the knowledge base containing several dozen rules. The algorithm of direct chaining is applied. Based on the preconditions (sensor readings) and using the rules from the knowledge base, the expert system reaches the decision whether to try fixing the problem autonomously or to alarm the person in charge (expert with PDA).

This is just a simple example of what should be realized in practice. Depending on the actual application, any expert system with its knowledge base can be easily integrated in this software.

3.6. Internet Connectivity

The system database can be accessed from a Web client, with the purpose of shortening the response time in critical situations. In case of an emergency, the expert who is not on site and/or doesn't carry a PDA can easily be consulted. All readings are instantly available, and it is sufficient to have a computer connected to the Internet to access them.

The database is realized using MySQL Server. This is a free product with a large user population and excellent documentation, and these are the reasons that helped us choose this DBMS. However, a big limitation of MySQL is that it doesn't support transactions. Therefore, in cases of big workload we recommend use of some other DMBS (such as Microsoft SQL Server, Oracle, etc).

For communicating with the database, PHP 4.0.4 along with Apache Web Server 1.3.14 was used. Both products are free, reliable and well documented. PHP proved to be very handy because of built-in support for many databases, MySQL included.

The system uses a standard three-tier architecture for database access: Web client, server-side script and DBMS.

The client formulates a database query, and sends it to the Web server in the form of a HTTP request. On the server, the request is handled by a PHP script, which creates an SQL query. The query is then forwarded to DBMS which performs the database access and retrieves the records. The script receives a recordset (that can be empty) as a result, forms a



Date	value1	value2	status
2003-05-07 16:25:03	18	362	0
2003-05-07 16:25:06	140	354	0
2003-05-07 16:25:08	270	354	0
2003-05-07 16:25:12	480	358	0
2003-05-07 16:25:13	590	359	0
2003-05-07 16:25:18	720	355	0
2003-05-07 16:25:21	850	355	0
2003-05-07 16:25:24	980	349	0
2003-05-07 16:25:28	130	353	0
2003-05-07 16:25:31	190	353	0

Figure 3.6b: Query results page

HTTP reply, forwards it to the Web server which then sends it back to the client. In that way, the dynamic page creation on the client side is performed.

Database can be accessed with the correct username and password only. Unauthorized persons can't

access the potentially classified data (power plants, hospitals...). The user who successfully logs in can choose sensor(s) he/she wants to access, and then enter a simple query that is later translated into SQL. Possible searches are by time (the user enters the time interval for which he/she wants to display the sensor readings) or by value (the user enters the value interval in which he/she wants to conduct a search). Depending on the application, it is possible to implement any type of search (multiple tables and attributes, etc).

We are planning to extend this simple system in order to enable the administration of the sensor configuration and operation. The idea is to control some parts of the system from a Web client, that is, to achieve some degree of Internet automation. In that way, we could implement full remote administration, which would eliminate the need for the physical presence on the server that executes the data acquisition software.

4. Testing and Integration

The testing was performed in three phases: separate component testing (protocol, routing module, data acquisition software, database, dsp sensor), component integration and final system integration

4.1. Separate Component Testing

The ad-hoc multihop routing protocol has been tested using the simulator which was written in Java for that purpose. The results are given in 3.1.5. Usability Analysis. As for the technical side of the simulator, it has been tested with more than 100 mobile nodes (threads). No problems were reported. The number of nodes is limited by the processor speed, due to the principle of time-sharing between threads.

As for IFRM and DSPS, PCB design and power supply were tested first. Then, test programs were written for checking RS232, signal LEDs and external RAM (IFRM). After correcting several problems with the WatchDog Timer, all tests passed. After that, adequate routines for communication with the keyboard and LCD were written (IFRM). While this testing was going on, the server version of the routing protocol was written and tested in a point-to-point connection between two computers. Then, the hardware module was tested in the same way (with implemented routing protocol).

For the purpose of Shell (data acquisition software) testing, a simple simulation of the sensors and the PDA was designed. The program simulated external devices by connecting to the several ports and waiting to be called from Shell. During these tests, beside the discovery and correction of many minor bugs, it became clear that the Windows timer is not precise enough for the measurement of the sensor reading cycles. Therefore, it was decided that this clock should be taken directly from the processor, but this part has yet to be implemented.

Internet connectivity has been tested in a laboratory with several dozen computers. The server was assigned an existing IP address, and subsequent simultaneous access from

other computers caused no problems. Next, the Web Server and MySQL Server were left running for a few days, and were accessed multiple times from a different networks, again without any problems.

4.2.Component integration

In the first phase of the integration, the data acquisition software (Shell) and the database were connected together. After that, the feature of database access from the Internet was added. Finally, the communication between the Shell and the interfacing and routing module was tested.

A driver has been written in ANSI C that enabled us to attach the Bluetooth to the server side without any modifications to the Shell. Thus, the Bluetooth occupied one port and waited for the incoming data. The other Bluetooth was attached to the other computer in the same way. The communication between the module on the server side and the other module passed without problems. After that, the dsp sensor was attached in place of the second computer and the transmission of the test data was successful.

4.3.The final integration

Finally, the test of the complete system was performed. Because of the problems we already mentioned (the modules that are not conformant to Bluetooth v1.0b specification, and the delay in the project kit delivery), we weren't able to test the routing module in the real situation. However, we resorted to a compromise solution and used the simulator (which was originally written for the testing of the ad-hoc protocol) for the system testing. The simulator was modified in order to accept one routing module attached to the serial port and treat it equal with the other simulated nodes in a network. Thus we created an ad-hoc network with as many as 100 nodes, one of them being the hardware designed interfacing and routing module. This kind of test can't be equal to the test in the real working conditions, but is the best alternative we could devise under these circumstances.

5. Summary

The goal of this project was not to provide a complete hardware/software infrastructure for the wireless sensor multihop ad-hoc network, because it wasn't possible with the given time schedule and the available equipment. Instead, we wanted to indicate a new course of development of the wireless communications: integration of the different electronic devices in a single information network (based on the Bluetooth environment) with the universal routing protocol.

However, the designed hw/sw platform does provide a good base that can be used in critical systems (such as health-care institutions or industry) with some minor modifications. Beside, it's important to note that the system is open: any device that is capable of serial communication can be connected to the designed routing module. Therefore, it is possible to create custom ad-hoc networks, and not only data acquisition sensor networks described in this paper. We demonstrated this by designing the PDA, which communicates with the server via interface and routing module.

The next step in this area is development of Java-enabled microcontrollers, that is, introduction of the JVM chips. That will enable the execution of the universal routing protocol. Also, a Web server capability could be incorporated within each network node.

Many improvements are possible in this project: the routing protocol demands detail research (analytical and simulation) in order that drawbacks could be identified and fixed; the interface and routing module is yet to prove itself in the real-world application (with the introduction of point-to-multipoint Bluetooth modules); the additional effort has to be made in making the software even more modular, so that it follows the guidelines of open and easily replaceable system; it is necessary to develop several classes of sensors for the most frequent measurements, while the potential integration with the GPS systems would create a number of new application possibilities.

References:

- [1] Bluetooth Core Specification v1.1,
http://www.bluetooth.com/developer/specification/Bluetooth_11_Specifications_Book.pdf
Bluetooth SIG, 2001
- [2] Miller, B.A, Bisdikian, C., "Bluetooth Revealed", Prentice Hall, 2001
- [3] IETF, Manet Group, "Ad Hoc On Demand Distance Vector (AODV) Routing", 2001
<http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-08.txt>
- [4] IETF, Manet Group, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks", 2001
<http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-05.txt>
- [5] IETF, Manet Group, "Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification", 2001
<http://www.ietf.org/internet-drafts/draft-ietf-manet-tora-spec-03.txt>
- [6] IETF, Manet Group, "Landmark Routing Protocol (LANMAR) for Large Scale Ad Hoc Networks", 2001
<http://www.ietf.org/internet-drafts/draft-ietf-manet-lanmar-00.txt>
- [7] Microchip Corporation, "PIC17CXX Datasheet", 2000
<http://www.microchip.com/Download/lit/pline/picmicro/families/17c75x/devices/17c756a/30289b.pdf>
- [8] Microchip Corporation, "PIC17C7XX Programming Specifications", 2000
<http://www.microchip.com/Download/lit/suppdoc/specs/30274b.pdf>
- [9] Maxim Corporation, "MAX3225 RS-232 Transceivers Datasheet", 2000
<http://pdfserv.maxim-ic.com/arpdf/1782.pdf>
- [10] Velašević, D., Bojić, D., "Zbirka zadataka iz ekspertskih sistema", Elektrotehnički Fakultet, Beograd, 1996
- [11] R. Leinecker, R., Archer, T., "Visual C++ 6 Biblija", Mikroknjiga, Beograd, 2000
- [12] Microsoft Corporation, "Microsoft Developer Network Library"
<http://msdn.microsoft.com>
- [13] Texas Instruments, "TMS320LF2407, TMS320LF2406, TMS320LF2402 DSP controllers"
<http://www-s.ti.com/sc/psheets/sprs094f/sprs094f.pdf>
- [14] Texas Instruments, "TMS320LF240x Flash Programming", TI DSP CD
- [15] Texas Instruments, "TMS320LF/LC240x DSP Controllers Systems and Peripherals Reference Guide"
<http://www-s.ti.com/sc/psheets/spru357a/spru357a.pdf>
- [16] Texas Instruments, "TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide", TI DSP CD
- [17] Dr Miodrag V Popović, "Digitalna obrada signala", Nauka, Beograd, 1999
- [18] "MySQL Reference Manual (ver. 3.23.22)"
<http://www.mysql.com/downloads/manual/manual.pdf>
- [19] Stig Saether Bakken et al., "PHP Manual"
<http://www.php.net/manual/en/>