A&QT-R 2002 (THETA 13)
2002 IEEE-TTTC International Conference on Automation, Quality and Testing, Robotics
May 23 – 25, 2002, Cluj-Napoca, Romania

INTERNET Interface for Industrial Processes

Emil Voişan, Florin Drăgan, Onuț Lungu

Department of Automation and Industrial Information
Faculty of Automatics and Computer Sciences
"Politehnica" University of Timisoara
Bd. V. Parvan, no. 2, 1900, Timisoara, Romania
Tel.: 40-56-204333
Fax: 40-56-192-049
Email: evoisan@aut.utt.ro, fdragan@aut.utt.ro, olungu@aut.utt.ro

Abstract: Over the past few years it could be noticed a growth of appliances in automation and electronic fields which uses the INTERNET network to give the remote control over different processes – from industrial to natural ones. These appliances are based now on a wide range of controllers which offers an increased process capabilities and possibility to connect in a different kind of networks at a relatively small price. So, based on this kind of controllers it could be interfaced with the INTERNET network a wide types of devices such as measurement devices, ATM's, home devices. What it offers new is possibility to surveillance through INTERNET devices and processes of industrial automation.

Keywords: Internet interface, Tini, SNAP.

1. INTRODUCTION

Early versions of hardware architectures, which were providing access throughout Internet network to industrial devices, were based on a computer, usually PC. A solution of this kind was provided by *emWare*. This standard presumes to associate each device with a DLM (Device Link Module), which manages low level connections (hardware) with devices throughout serial lines RS232, RS485, Ethernet or others. A web server offers objects, which represents the interface between device and user. This involves using a computer (PC) as a gateway.

Dallas Semiconductor provides a much interesting solution. Tiny InterNet Interface (TINI) is a platform built around a C390 controller, which integrates all sort of peripherals – four serial interfaces, microLan, two CAN (Controller Area Network) controllers, Ethernet interface and a real time clock, along with set of Java application programming interfaces. Purpose of this platform is to offer a interface into the INTERNET for small devices connected to it and this allows the devices to be monitored, controlled and managed remotely. Controller works at 33Mhz and it has 512KB static memory. Operating system, TINIOS, is based on Java™ virtual machine. In this case in no need for PC, only if is necessary to storage data's in a database.

2. PACKAGE SPECIFICATIONS

2.1. *TINI main specifications*

- TINI Chipset assembled on a standard 72pin SIMM
- Hosts the TINI runtime environment in validated hardware design
- The extensive I/O capabilities of the DS80C390 microcontroller are exposed through Java TM APIs:
  - Dual serial ports 1
  - Dual 1-Wire net interfaces
  - Dual CAN (Controller Area Network) controllers
  - 2-wire synchronous serial bus
  - General-purpose digital I/O
  - Easy system expansion using a parallel bus interface
- Direct connection to the 10 or 10/100Base-T Ethernet network
- Real Time clock for time stamping
- Flash ROM for storage and execution of runtime environment
- 512K/1MByte nonvolatile SRAM provides fast, unlimited write operations and persistent data storage
- Level translator provides RS232 voltages
- Wide operating temperature range, -20°C to +70°C
- Requires only a single +5V power supply

2.2. *S.N.A.P. networks*

Scalable Node Address Protocol is *HTH* proprietary and was initially designed for home automation system *PLM-24*. Request was for a simple and adaptable protocol, both for simple and complex appliances, result was scalable protocol.
S.N.A.P. allow different packet data length and different complexity. It can be used as simple protocol without flags and error corrections or it can use up to 24 flags and any detection error method defined, depending on requests. Since S.N.A.P is scaleable, both simple (read as cheap) and sophisticated nodes can communicate with each other in the network.
A short description of the S.N.A.P. protocol:

- Easy to learn, use and implement.
- Free and open network protocol.
- Free development tools available.
- Scaleable binary protocol with small overhead.
- Requires minimal microcontroller resources to implement.
- Up to 16.7 million node addresses.
- Up to 24 protocol specific flags.
- Optional ACK/NAK request.
- Optional command mode.
- 8 different error detecting methods (Checksum, CRC, FEC etc.).
- Can be used in master/slave and/or peer-to-peer.
- Supports broadcast messages.

- Media independent (power line, RF, TP, IR etc.).
- Works with simplex, half-, full- duplex links.
- Header is scaleable from 3-12 bytes.
- User specified number of preamble bytes (0-n).
- Works with synchronous and asynchronous communication.

S.N.A.P. was developed primarily for use in home automation and control systems but it is a generic protocol and not limited to this. S.N.A.P can be used in any type of applications where an easy to learn and flexible network protocol is needed.

S.N.A.P can be implemented in almost any microcontroller available today. The first S.N.A.P test program ran on a BASIC Stamp I from Parallax Inc. It sent 1 Byte data and used 16-bit CRC as error detection method. This tiny microcontroller has only 14 Bytes of available RAM. As another node in the network was used a standard Pentium II PC and the nodes were communicating over the mains using PLM-24Power Line Modems.

All communication between network nodes is in the form of packets. These packets can be of different length. The total packet length will depend on how many address bytes (0-6 Bytes) is necessary to use, how many flags bytes (0-3 Bytes), how much data you want to send (0-512 Bytes) and what error detection method you use. All of this is defined in the header definition bytes (HDB2 and HDB1).

Each packet could be preceded with optional preamble bytes (0-n). The packet starts with a unique byte (01010100). This byte is called the synchronization byte. Any type of preamble characters can be used as long as they are not the same as the synchronization byte (fig 1).

SYNC	Synchronization byte
HDB2	Header Definition Byte 2
HDB1	Header Definition Byte 1
DAB1	Destination Address Byte
SAB1	Source Address Byte
DB1	Data Byte 1
CRC2	High byte of CRC-16
CRC1	Low byte of CRC-16

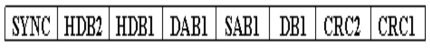

Fig. 1 – S.N.A.P. packet example

The total length of this packet would be 8 Bytes (excluding the optional preamble bytes). All bytes within a group are positioned with the least significant byte to the right.

3. PRACTICAL IMPLEMENTATION

A possible solution for a home automation or control system on industrial field using TINI modules and S.N.A.P. nodes it is given next. Purpose of this part is to present a concept about what this system is intended to be, limitations and instruments types that can be used in such a application.

There are a few characteristic of this kind of system.

3.1. *Structure of the system*

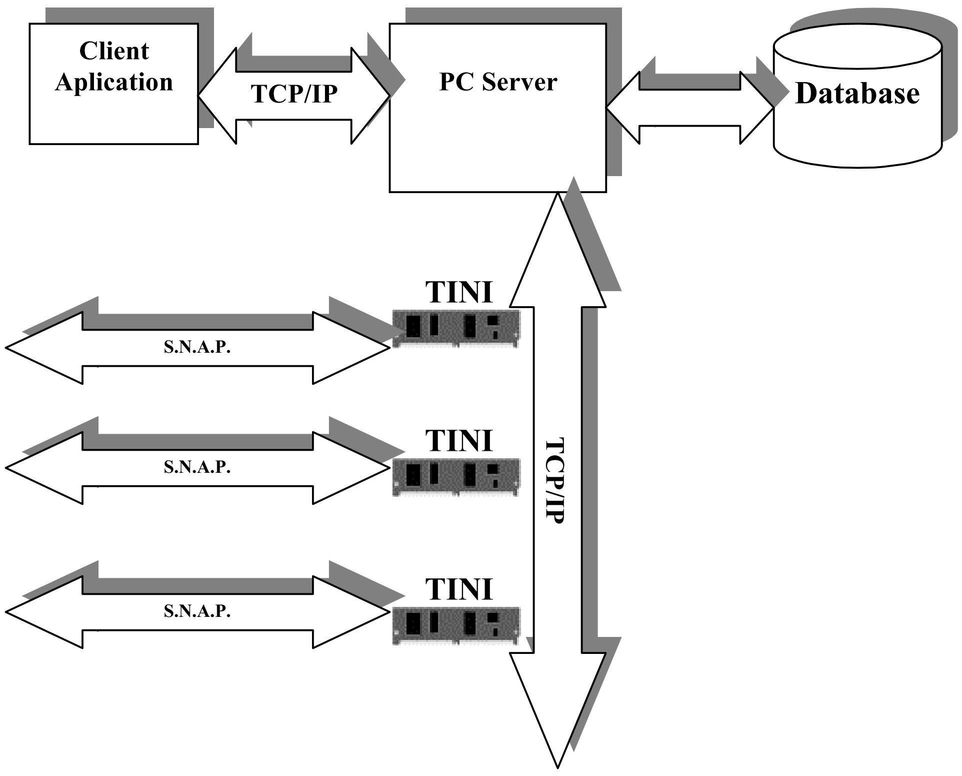

Fig. 2 – Functional scheme

Client application

This application is running on a workstation either standalone, either is running from a browser (*Java Applet*) and offers to the users the possibility to access data collected by TINI modules from S.N.A.P. networks. Users from the administrator group have the possibility to add new TINI modules to the system or to configure instruments from S.N.A.P. networks.

PC Server

The server contains a module that periodically gather data from TINI modules and depending on user options saves it on database. Users options are reflected by jobs registered on server. Users authentication is also the server responsibility. Implementations of several access levels to devices and data is necessary. Minimum would be two access levels:

- users that only can read data and register history jobs;

- users (administrator group) that can also configure S.N.A.P. devices or TINI modules;

TINI Modules

TINI modules are connected with the PC server by an Ethernet network. Each module can work with 253 S.N.A.P. nodes. Those nodes are scanned periodically by each module. Each module contains configuartion for each device and the last readed value. The S.N.A.P. network is just an alternative for connecting to different devices, its advantage is serial interface RS485 that allows long lines of communication. Other available solutions are CAN (*Controller Area Network*) and MicroLan(*1-Wire*), solutions for which TINI's API and hardware is offering support.

On this system are three kinds of hardware that communicates: PC's, TINI modules and S.N.A.P. nodes. That hardware can use different protocols to communicate. For example between client and server it could be used RMI (Remote Method Invocation) and between TINI modules and server could be made through XML (Extended Markup Language).

S.N.A.P. nodes could be measurement instruments, sensors or nodes that can interact with the controlled process.

3.2. *Target processes*

This kind of system it could be useful on industrial areas where are important statistics about evolution of different processes. For example an industrial process in which counts the evolution of temperature or pressure. This system offers not only a modern approach of data acquire, but also could be created different access levels to information for users.

This system is not intended to control high complex industrial processes or time critical ones. The purpose is to made an interface between users and process, to offer data in an easy to interpret way and to offer remote access through Internet to this data.

3.3. *Restrictions and limitations*

This system is made from three kind of hardware that communicates between them. Control for fast processes is not possible because of the delays in S.N.A.P. network ~5s (253 nodes, ~20ms/node) and because of delays from the others networks. Other restriction is the number of servers on which a TINI module can be registered, this is given by the relatively small number sockets that ca be opened on a TINI module.

3.4. *Applications that justify this kind of system*

System limitations from the process speed represent the factor that determines if this system is suitable for one particular process or not. Major advantage for this kind of application is the fact that can cover a wide area, practically in any place where is available an Internet connection it can be placed a TINI module with a SNAP network. Data from each node is centralized on server form, which is available anywhere in Internet. Other point of view is the economical one. For example, if PC's would be used as gateways for SNAP networks the costs would be not justified. TINI makes possible a network that covers a wide area at a reasonable cost. A typical application for this system is a weather station – covering an area with temperature and pressure sensors.

4. CONCLUSIONS

- Wide geographical coverage.

Major advantage of this application is possibility to cover large areas. TINI module can be connected to Internet making use either of it's built in Ethernet interface, or by PPP protocol. There are three ways to do that:

- Ethernet network – useful for not so large areas where liability it is critical.
- Rented line – for large areas, and where is possible installation of those kinds of lines.
- Dial-up connections – large areas where costs for rented lines are not justified, home or industrial automation where time it is less critical.

- System availability

Depends on the work environment and communication media between modules and server, and depends directly of SNAP device's availability. Whatever, the most important is the Internet connection, practically one module becomes unavailable if the ISP it stops its activity. From server point of view its performances are give by the Java virtual machine and it is running on PC, which is not designed for industrial applications.

- Fault tolerance

Faults that could influent the system are mainly given by the communication between the system parts. At the SNAP level fault tolerance is acquired through detection errors technique and requests retransmissions. Anyway at this level is the highest probability for errors, because of the industrial environment. TCP/IP connections between the server and TINI modules would be remade in case of failure.

- Extensibility

System it is easy extensible, it could be started with a minimal configuration and other modules will be added later. Adding new modules does not require server restart and SNAP devices will self-registered.

- Modularity

System is constituted from three independent modules. Each module can be enhanced without any modification to other modules as long as the communication protocol it is not affected.

5. REFERENCES

[1] http://www.hth.com
[2] http://www.dalsemi.com
[3] http://java.sun.com/
[4] Elliot R. Harold, [2000], Java Network Programming, O'Reilly.
[5] Brian Jepson, [1996], Java Database Programming.
[6] Brett McLaughlin, [2000], Java and XML, O'Reilly.