The ITB Journal

Volume 5 | Issue 1 Article 28

2004

Web Enabled Embedded Devices

Anthony Keane
Institute of Technology Blanchardstown, anthony.keane@itb.ie

Brian Myler
School oflnformatics and Engineering, Institute of Technology Blanchardstown, Dublin 15.

Web Enabled Embedded Devices

Brian Myler and Dr. Anthony Keane
School of Informatics and Engineering, Institute of Technology Blanchardstown, Dublin 15
Anthony.Keane@itb.ie

Abstract

The trend in manufacturing of computerised control systems has been to miniaturise the components while increasing the functionality of the systems. This has led to the development of small inexpensive hand-held computer devices coupled with the availability of a user friendly application development language, Java and public cost-effect communication networks has given the developer a programmable web-enabled embedded device. This paper investigates the steps involved in programming the Tiny InterNet Interface platform and analyses the limitations imposed by miniaturisation on this device.

Introduction

Historically we have seen the early stand-alone computational machines quickly gave way to large and expensive central processing computers that allowed many users to run programs and communicate using remotely connected dumb terminals. In time, the microcomputer answered the demand for generalised computing with cheap processing power under the control of the individual. Networking these microcomputers allowed the individuals to communicate via email and to share files, like on the expensive mainframes. The programs and technology protocols that were developed to allow networking of computers has evolved into a global system, called the Internet, where any web-enabled device can participate, Manders et al. (2002). Embedded devices are small computing control systems that have many parts in common with computers like the CPU, memory, circuitry, power source and interfaces, among others. The main differences lie in the limitations of the components and the purpose of the device. Many computers are designed as general purpose machines offering multipurpose usage whereas embedded devices are often designed as stand-alone control systems with a particular simple role, like an alarm system, control of a washing machine, fridge, etc. Advanced embedded control systems are used by car manufacturers, military and manufacturing industry, especially where automation is required.

Today, over 90% of all microprocessors are used for real-time and embedded applications. Manufactures have long recognised the convergence of technologies and communications but have been prevented for exploiting it due to the unavailability of low-cost high bandwidth networks. Also there was no standardisation across the industry of hardware or software development tools. And embedded systems require real-time programming skills which tend to be found on specialist courses for systems engineers. Recently, some manufacturers are starting to provide a simple, flexible and cost effective means to design a wide variety of hardware devices able to connect directly to corporate and home networks by using a combination of a small but powerful chipset with a Java programmable runtime environment.

These platforms of embedded devices can now easily be networked using Ethernet interfaces or using a built-in serial port that can drive an external modem. By allowing them to attach to the Web, client browser screens can be indirectly interfaced with sensors and other embedded systems for management and control. One such communication service is the GSM network where embedded systems can send notification messages via short message service (SMS) or receive data the same way. SMS is well suited for interactions of mobile and ubiquitous computing devices when the input from users can be restricted to simple decisions, confirmations, or input that is based on a set of predefined answers. Sun Microsystems originally developed Java (Arnold at al. 2000) to address the need of a high-level programming language with build-in tools (Application Programming Interfaces) for web-enabled devices. Combining the flexibility of Java with the Web technologies gives the industry and developers a standard set of tools to easily and cheaply create web-enabled embedded monitoring and control systems.

Description of Tini-InterNet Interface (TINI)

Dallas Semiconductor created a microcontroller chipset to provide system designers and software developers with a software development platform to interface with wide variety of hardware devices and to be able to connect directly to networks. TINI is based on the DS80C390 microcontroller which integrates support for several distinct forms of I/O including serial, 1-Wire and Controller Area Network (CAN) bus. The chipset contains flash ROM for the runtime environment and static RAM for system data file storage. The hardware is accessed by the software developer using Java's application programming interfaces while the chipset provide for processing control, communication and networking capabilities, see figure 1.

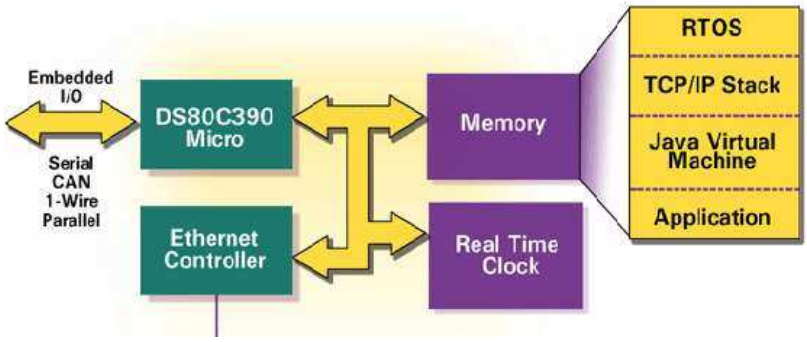

Figure 1: Components of the embedded system

The TINI platform allows everything from small sensors and actuators to factory automation equipment and legacy hardware access to the network. The combination of broad-based I/O capability, a TCP/IP network protocol stack and object-oriented programming environment

enables programmers to easily create applications that provide local and remote control of TINI-based devices through standard network applications such as Web browsers, see figure 2.

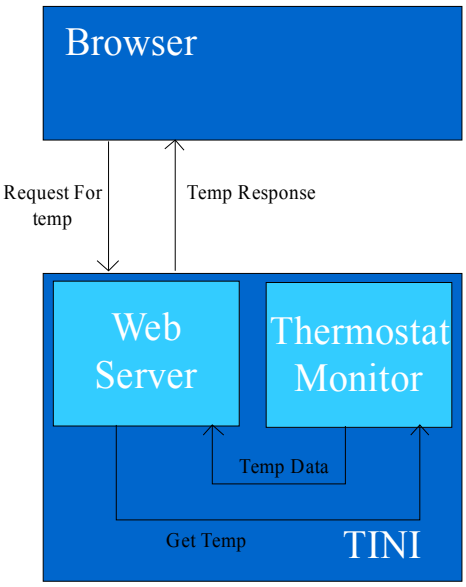

Figure 2: Software components of system

Developing on TINI

Remote Temperature Sensor Monitoring and Reporting System

To investigate the TINI platform, it was decided to build a simple temperature monitoring system that would allow remote monitoring to take place via any browser on the Internet and also for the system to inform the user of exception events, like a temperature above a threshold, using the SMS service. This project involved interfacing several hardware devices to the TINI platform, programming the board to accept the signals, analyse the signals and create an output signal informing of the results.

Configuration of TINI

The TINI platform consists of a €50 chipset consisting of a 33MHz processor, 1MB RAM, and various interfaces including RS232 DTE and DCE, Ethernet 10Mbps, 1-wire and iButton, see figure 3. TINI requires a 6V DC power supply which allows it to be battery operated. Initially, the firmware and operating system are downloaded to the TINI board from a PC using the RS-232 interface. Once this is completed you can give an ip address to TINI and access the board via TELNET and the Ethernet interface.

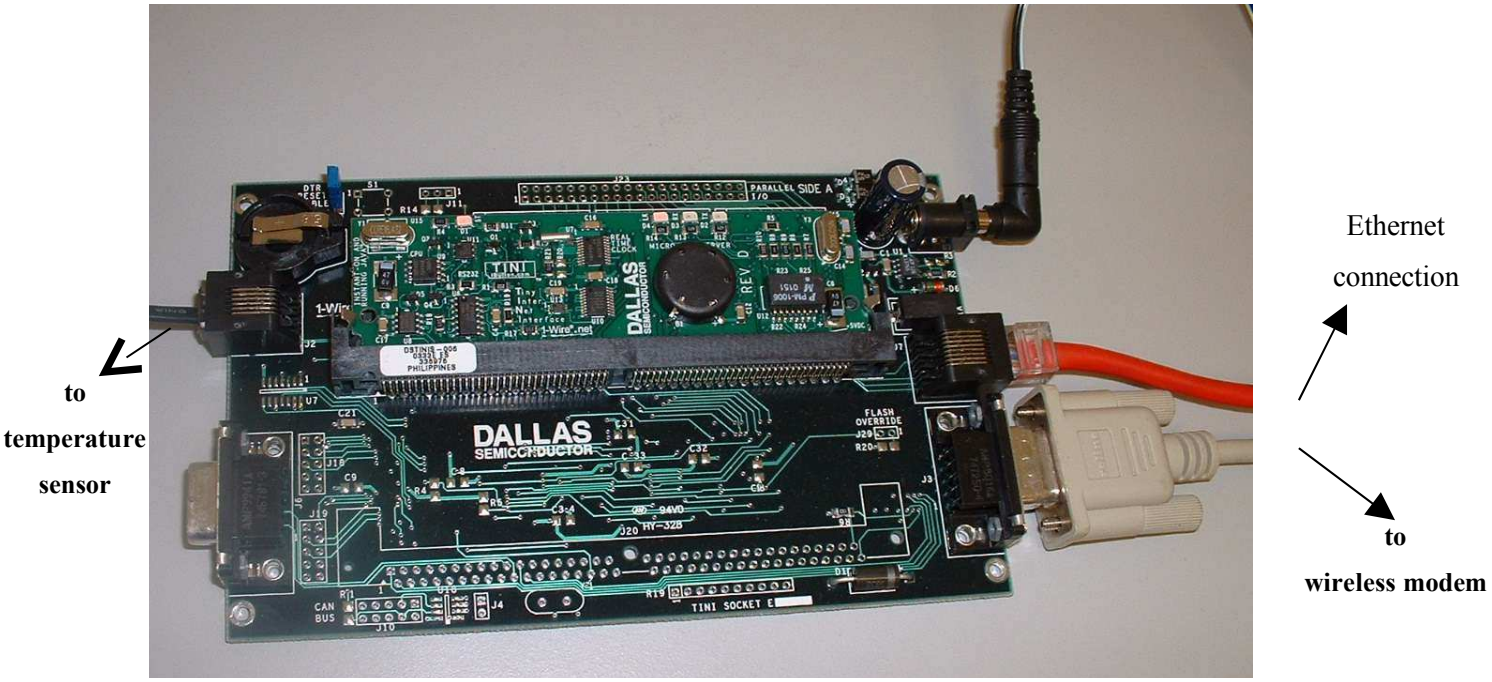

Figure 3: TINI board with attachments

The TINI OS incorporates a FTP server which allows applications to be developed on a host machine and the class files copied to the TINI board for execution. Although TINI supports java, it will not run class files, instead it uses another format called tini files. The difference is a class file will hold the compiled bytecode of a single Java class or interface whereas the tini file can contain several Java classes, the TINI native libraries, and associated resource files.

The tini file is created by converting the Java classes from class and jar files, and copying in other resources. Converting Java classes mainly involves modifying the constant pool. Many UTF8 strings are removed, including the class names that would otherwise support Java's dynamic classloading and reflection features. The memory space constraint imposes the need for compact code thus requiring the support for the bytecode verifier and the debugging information to be removed.

Two utilities provided by Dallas Semiconductors for creating tini files are the TINIConvertor and the BuildDependancy. The TINIConvertor is used to build the TINI binary and can be used directly if the program does not depend on any classes that are not included in the standard TINI API, otherwise BuildDependency must be used. This process builds in extra dependencies before calling TINIConvertor to build the binary.

Another way of automating the build process of TINI applications is to useTiniAnt, which is an extention to Apache Ant the cross platform build tool. TiniAnt adds a new task type to ant for building large TINI projects with complex dependencies.

Choosing a Web Server

Servertec Internet Server TINI Edition is a full-featured Application/Web Server written entirely in Java designed to run on Dallas Semiconductor TINI boards. It has the following features that make it a good choice; it is platform independence and uses open standards, gives high performance using multi-threading and has a full-featured Servlet engine. Fault tolerance is provided as crash protecting and recovery technology automatically traps, recovers from and logs exceptions. Additionally Servertec Internet Server TINI Edition protects the integrity of the server environment by preventing exceptions occurring in one request handler from affecting other requests that the server is processing. The other advantages are the size, less than 50KB, and is easy to install.

Figure 4 is a screen capture image showing the client browser screen that is remotely connected with the TINI web server and temperature data from the sensor is relayed. This is an example of remote monitoring. The application running on the TINI board can analyse the temperature to see if it is within a range and activate an exception message if the measured data exceeds a threshold value. The exception report can be sent either by landline connection or wireless via SMS should the fixed line not be available. This is an example of affordable redundancy in communication links.

Figure 4: Web browser page

Limitations of TINI

The development limitations of the TINI platform were mainly due to the hardware configuration where the small memory and slow processor speed were easily saturated if multiple signals from different sensors and multiple applications are competing for the resources. Application developers today are used to the generous resources available on PCs and can afford to ignore the bounds of memory and processing power since the typical application's requirements falls short of the limits. When confronted with an embedded system with limited physical resources and only a subset of the JDK API's available, the developer is forced to tighten up their coding design and become more efficient in implementing their application. This can be achieved by building and debugging the application in stages rather than trying the whole thing at once. TINI has proved to be an excellent inexpensive networked development platform where proof-of-concept ideas could be tried out, inexpensively and quickly. Also, the modular nature of the TINI platform could be used to increase available resources by daisy chaining several TINI boards together, with each board being responsible for a different sensor.

Conclusions

The TINI platform's integrated I/O demonstrates the flexibility that web-enabled embedded chipsets with ease in programming using the Java technology can dramatically increase the development cycle. Many manufacturers have followed this approach by providing development platforms of their own, similar to TINI but increasing the capacity of the resources provided on the chipset. The common thread with all the platforms is Java and Web availability. Recent developments in the mobile communication networks have opened a new affordable public gateway to the services that can be provided by and from embedded systems. We have demonstrated the usefulness of having a programmable inexpensive web enabled embedded device that can easily be configured and interfaced to a sensor. Coupling this with the available access using multiple public communication links allows for an effective monitoring, information and control system.

Acknowledgements

The authors wish to thank Declan Barber for the loan of the wireless modem in the testing of this project and Conor Gildea for the useful discussions and helpful advice regarding the configuration of the TINI board system.

References

K. Arnold, J.Gosling and D.Homes

The Java Language, Boston: Addison-Wesley, 2000

M.F.A.Manders, P.J.F.Peters, J.J.Lakkien and L.M.G.Feijs

Taxonomy, Architecture and Protocols for Web-enabled Embedded System

Proceedings of PROGRESS Embedded Systems Symposium 2002

http://www.ibutton.com/TINI/

http://www.maxim-ic.com/TINIplatform.cfm