CGI-based applications for distributed embedded systems for monitoring temperature and humidity

Grisha Spasov, Nikolay Kakanakov

Abstract: *The paper discusses the using of Common Gateway Interface in developing web-based distributed embedded systems. It shows the tree-layer model in developing client-server applications. An example using a BECK microcontroller SC12 in application for monitoring temperature and relative humidity in rooms with controlled microclimate is shown. SC12 is a system-on-chip with embedded RTOS, TCP/IP stack and CGI software.*

Key words: *Distributed Embedded Systems, Common Gateway Interface, CGI, Distributed Automation and Control, temperature and humidity measurements*

INTRODUCTION

Developing distributed automation systems based on Internet and Web technologies is a relative new trend in computer technologies. It gives a common and well-known interface to the client (web browser) and a standard way for communication between devices. All this is a result of invasion of computer network technologies and standards in the field of automation and Networked Control Systems (NCS) [1]. These technologies are used in business information systems, also known as Data processing. Typical examples of distributed automation standards, based on network technologies, are Industrial Ethernet, Ethernet/IP, Ethernet/IDA, PROInet V 2.0 [2, 3].

Common Gateway Interface (CGI) is a way of creating HTML pages with dynamic contents. It forces the server to execute some code and returns the result instead of the resource in URL. So using CGI, the client can monitor the internal states of the controlled device or/and send it commands.

This paper discusses the using of CGI in these two ways of communication. In such NCS every device is a server that can show and change its internal state on a client's request.

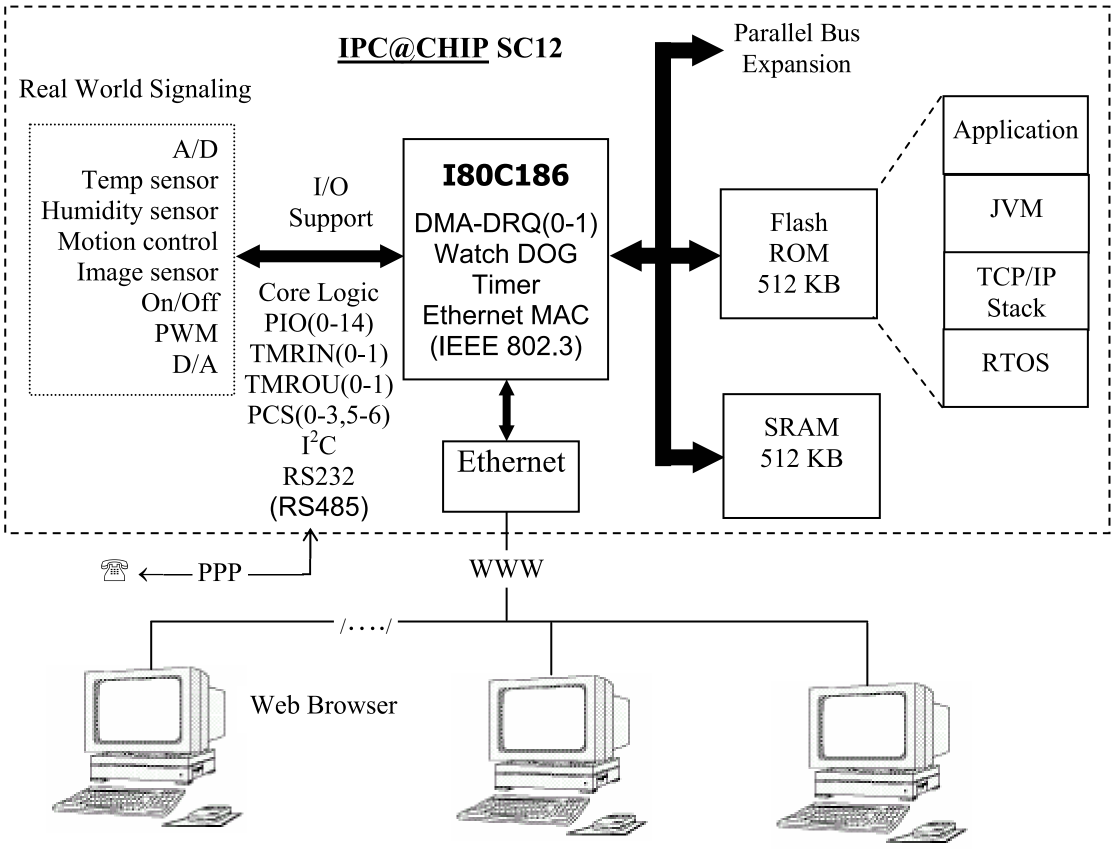

Figure 1. Microcontroller for embedded Internet.

Web-based, Internet-integrated embedded systems

The BECK's microcontroller IPC@CHIP SC12 is a new generation system-on-chip that has its own embedded hardware and software means for integrating into computer networks, based on TCP/IP communication [4].

The IPC@CHIP is a combination of hardware and software. The software is pre-installed and consists of a Real Time Operating System (RTOS) with file system, complete TCP/IP stack, PPP Server and client, web server, FTP server, Telnet Server and Hardware interface layer.

With this strategy, much of the required functionality is part of the operating system. Only the essential application specific functionality is implemented as a 16 Bit DOS application that is transferred to the file system per FTP.

The hardware consists of a 16 bit 186 CPU, RAM, Flash, Ethernet, Watchdog and powerfail detection. The CPU has functions such as timers, I/O lines, serial ports etc integrated. This hardware is available in a small package with a reduced pin count.

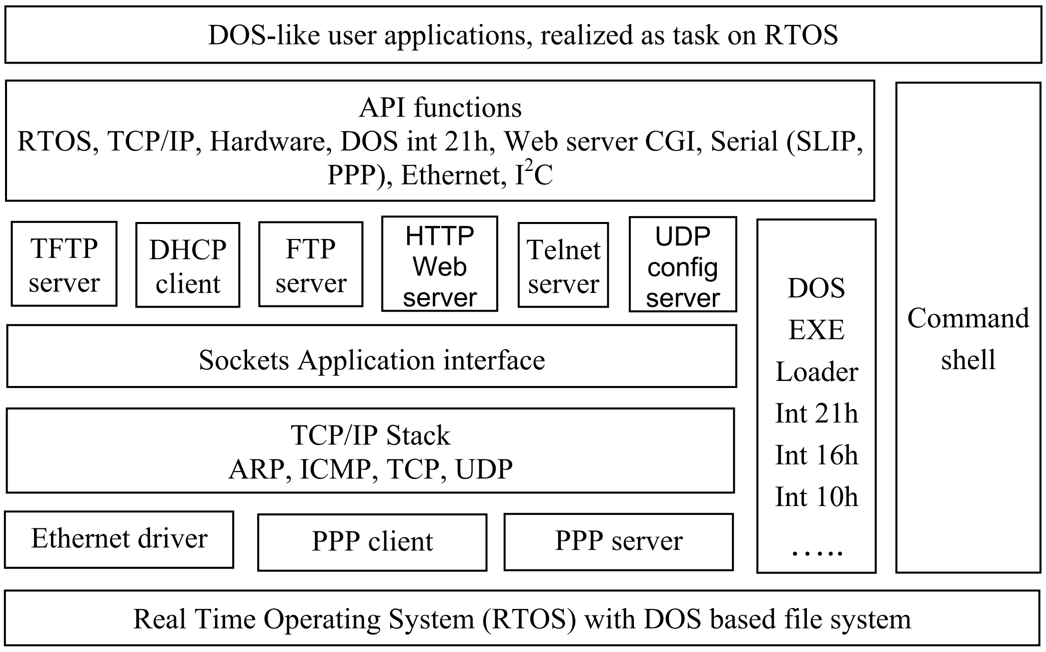
DOS-like user applications, realized as task on RTOS
API functions
RTOS, TCP/IP, Hardware, DOS int 21h, Web server CGI, Serial (SLIP, PPP), Ethernet, I2C
TFTP server
DHCP client
FTP server
HTTP Web server
Telnet server
UDP config server
DOS EXE Loader Int 21h Int 16h Int 10h …..
Command shell
Sockets Application interface
TCP/IP Stack
ARP, ICMP, TCP, UDP
Ethernet driver
PPP client
PPP server
Real Time Operating System (RTOS) with DOS based file system

Figure 2. Software platform of IPC@CHIP.

The TCP/IP Stack is a high performance stack with little compromises. It offers TCP, UDP, ARP, ICMP, Socket interface, 64 Sockets and 3 device interfaces (Ethernet, PPP server and PPP client). With the PPP client, your application can initiate a connection to the Internet. Over this PPP connection, the TCP/IP protocols and services as available over the Ethernet, now become available over long distance connections.

The software for IPC@CHIP is based on API functions from its RTOS and BIOS. For debugging, compiling and code generation developer can use every tool that can generate 16-bit applications for Intel 80186. The system allows developing in C, C++, Pascal, Assembler and even Java. For using Java applications on IPC@CHIP a special simple real-time java virtual machine (JVM) is needed. Such JVM is presented by RTJ Computing [http://www.rtjcom.com].

Three-layer client-server model in developing distributed embedded systems.

The client-server model is the base model in developing internet-based information systems. In this model client processes request data from the server or there are two modules: client – the one that requests service, and server – the one that executes the service. Communication between the client and the server is based on the exchange of HTML (Hyper Text Markup Language) documents. Transportation of these documents is based on the protocol HTTP (Hyper Text Transfer Protocol).

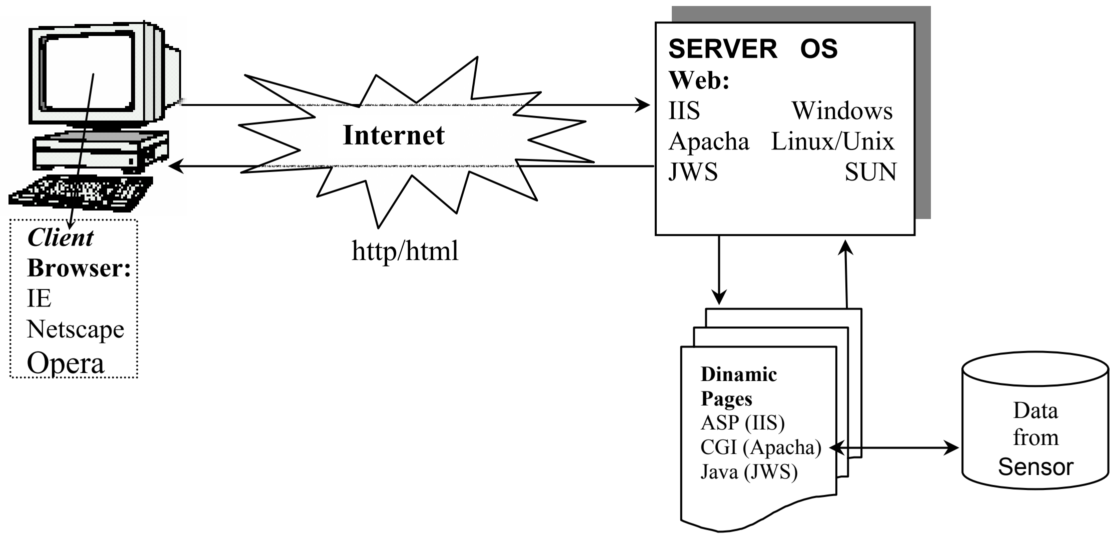

Figure 3. Three-layer model for client-server applications.

In most of the cases, especially in the data processing systems, the three layer model is used.

The first layer is the client that functions by the well-known browsers – Internet explorer, Netscape, Opera, etc.

The second layer is the Web server that works with dynamic HTML pages – IIS (Windows), Apache (Unix/Linux), JWS (SUN).

The third layer is a database system, or in embedded systems – a module that realizes the automation and/or control of a device.

The main advantage of this model is the dynamic generation of the information. On the client's request, the Web server starts a process that generates dynamically HTML page, based on the data from the third layer.

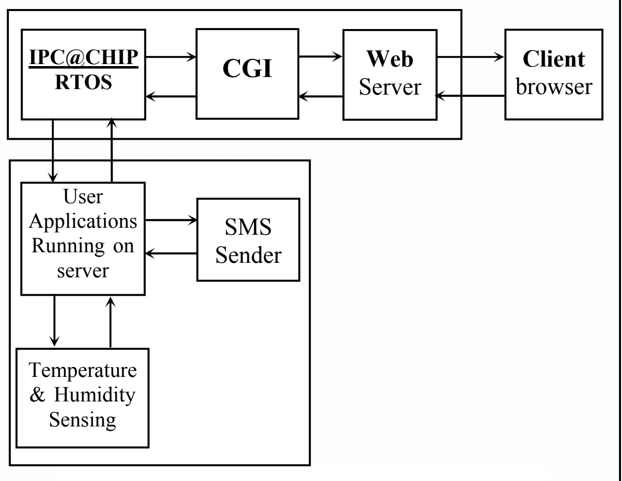
IPC@CHIP
RTOS
CGI
Web
Server
Client
browser
User Applications Running on server
SMS Sender
Temperature & Humidity Sensing

Figure 4. Functional blocks and relations

Figure 4 shows the three layer model in the distributed system for monitoring of temperature and humidity, based on IPC@Chip [5].

The functions of the system are separated between the four main blocks which corresponds with the three-layer on the following way:

- The internet browser assures that the system has a familiar user interface. The visualization of the monitored parameters is placed here.
- The integrated Web server in conjunction with the CGI deals with the control of the remote processes and transfer of dynamic HTML pages to the client. The real-time operating system (RTOS) of the IPC@Chip manages with the Web server's

tasks, TCP/IP communication, local peripherals, user tasks and interconnection between them.

- User's applications and tasks, running on the controller, are used for automation purposes. In our case, this is the sensor driving and management of the SMS communication.

The software of this system consists of several blocks. One block is for temperature and humidity measurement, one – for communication with the GSM gateway and sending SMS and one is for generating dynamic HTML documents with the data from the sensor.

The communication with the GSM gateway is based on the "AT" commands (Attentions commands) that are sent trough the serial interface of the microcontroller. The block for this communication is started as a task on the RTOS and converts the data to a SMS PDU (Protocol Data Unit) format to be readable by the GSM.

The measurement block consists of two identical parts, one – for temperature measurement and one – for humidity measurement. The microcontroller sends a command to the sensor to get data. Then it normalizes this data and stores it in the operating memory. The two identical parts are executed sequentially and are repeated together. Between every two iterations of this cycle the task "sleeps" for about one minute to free the processor for tasks with low priority (like FTP, Web, Telnet).

The communication between the controller and the sensor is trough two-line interface (data and synchronization). A C-library based on the software interrupts of the RTOS is made for this communication. This library has the following functions:

- shttransstart() – starts a transmission;
- shtreset() – for restarting of the sensor;
- shtinit() – initializes the sensor interface;
- shtsend() – sends a command;
- shtrecv() – receives data from sensor.

The last software block gets the data from the measurement block and generates dynamically HTML document containing this data. This block communicates with the client trough web interface.

The client sends a HTTP request to the web server and the server sends a HTML document as a reply. CGI is the thing that connects the web server with the internal data on the controller.

CGI is a standard interface between an external application (gateway) and information server (HTTP) [7]. When the client writes a URL address in the browser window, the browser connects with the HTTP server using the domain part of the URL and sends a request. The server finds the requested resource using the PATH part of the URL and transfers the file (resource) to the client. If the resource is an HTML document the browser visualize it, if not – the browser searches the appropriate program to open it. When using CGI, the server again finds the resource file, but it now executes it instead of sending it. After this the server sends to the client the results of the execution. In this way the client can force the server to execute some code and use the results. Using CGI, the client can send data to the server trough "environment variables" with the HTTP methods "Get' and "Post". This is a very common way for creating HTML documents depending on the internal state of the server and data send by the client.

In the IPC@Chip CGI has slightly different organization. The requested files are not executed on the request but are running as RTOS tasks on the controller. On request the RTOS switches to the corresponding CGI task and the task returns the results in the operating memory. The CGI reply consists of HTTP result code, return data type, data length and real data.

In the presented example, there is one CGI task running on the IPC@Chip. It generates a short HTML document containing the temperature and humidity values from the last measurement. This task do not use data received from the client. It uses data from the measurement block. This transfer of data is carried out with a block of shared memory. The measurement block writes data in this memory and the CGI task reads and use this data. There will be a problem if the two tasks address this block of memory at the same time. To avoid this problem and synchronize the work of the two tasks a semaphore is used. When one of the tasks gain access to this memory it reserves a semaphore and after it finishes its work it release the semaphore (figure 5). When one of the tasks tries to reserve already reserved semaphore the RTOS makes this task wait.

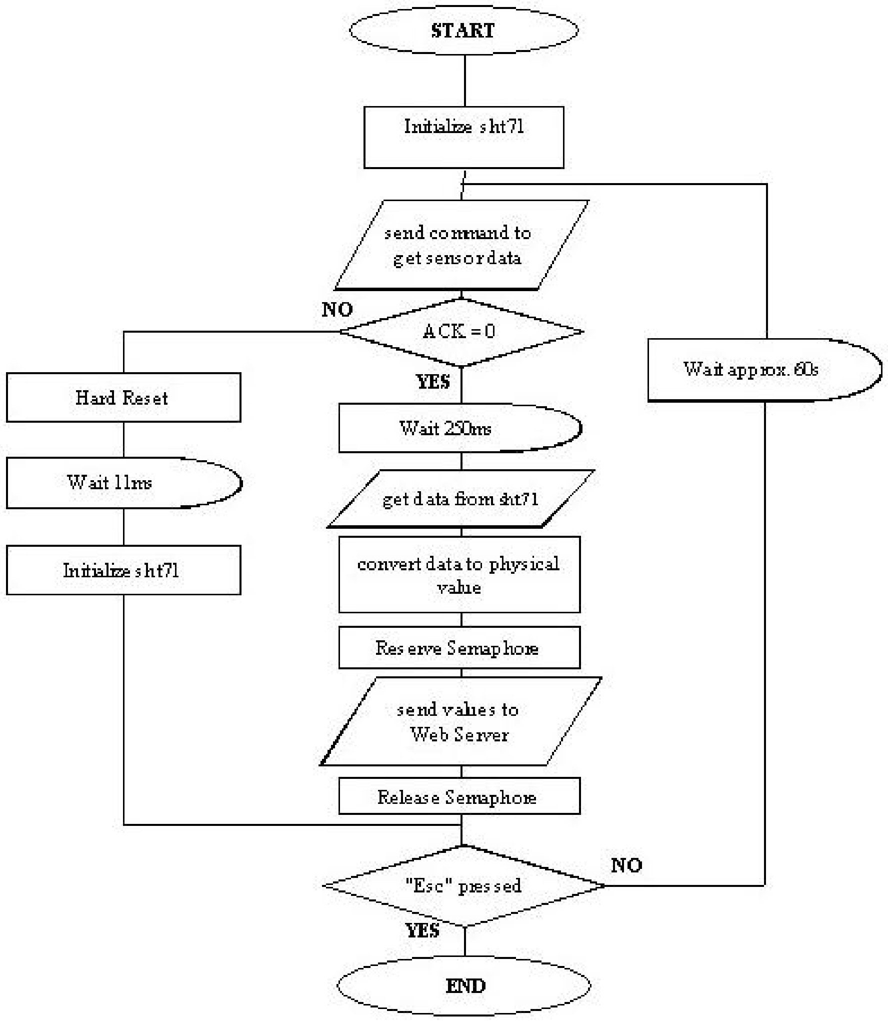

Figure 5. Using semaphores in sending measurement data to CGI task

Semaphores are a way of synchronization between processes that relies on the operating system. The RTOS of the IPC@Chip has API functions for creating, reserving and releasing of semaphores. It can supply up to 60 semaphores at the same time.

For controlling the parallel execution of this task a small program is written. This program read initialization data from files on the controller memory, installs and uninstalls the CGI processes, declares a semaphore from RTOS and frees it, releases the unused memory and stops the unneeded tasks. The other purpose of this program is to allow adding a serity to the project using identification and authorization.

CONCLUSIONS AND FUTURE WORK

The represented system successfully integrates the functionality of the sensor, GSM gateway and embedded web server and RTOS. The client-server model guarantees the openness of the project and ability to expand. Expanding can flow in the following ways:

- adding microcontrollers – the only limitation here is the number of nodes in a LAN;
- adding more clients stations – this stations can be with different platforms. The only limitation is that their operating systems has a TCP/IP protocol stacks;
- ability to develop hierarchical systems;
- ability to upload different user application on the controller (easy upgrade)

To add more functionality to this project, a forth layer can be add. This layer is a server that controls all microcontrollers in a local network. In this layer is integrated the security of the sensor network and it synchronizes the work of the nodes in the network. The communication between controller nodes and this serer is supposed to be based on XML(eXtended Mark-up Language).

Another way of expanding the functionality of the system is integrating a JVM (Java Virtual Machine) in the controller that will be used instead of the CGI. This will allow the developers to create more flexible applications.

REFERENCES

[1] Djiev, S. *Communication Networks in controlling systems* Automation and Informatics,No.2,pp 13-17, 2003.

[2] Buesgen, R. and Felf, J. *Real time on the Ethernet: how PROInet v2.0 improves on v1.0*, Control Engineering Europe, October, pp. 27-27, 2002.

[3] *www.ida-group.org/* IDA Specification ver 1.1.

[4] *www.beck-ipc,com/ipc/* IPC@Chip's Home page – Product info, datasheets, examples.

[5] Spasov G., Kakanakov N., *Distributed Embedded System for monitoring of temperature and humidity*, Elektrotehnika & Elektronika, 2004, [in press].

[6] Tanenbaum A., Steen M., *Distributed Systems Principles and Paradigms*, Prentice Hall, 2002.

[7] *www.CGI101.com/class/ -* CGI Tutorial.

ABOUT THE AUTHORS

Grisha Spasov , PhD, Department of Computer Systems, Technical University Sofia, branch Plovdiv, Phone: +359 32 659576, E-mail: gvs@tu-plovdiv.bg

Nikolay Kakanakov, master engineer, Department of Computer Systems, Technical University Sofia, branch Plovdiv, Phone: +359 88 7265505, E-mail: kakanak@tu-plovdiv.bg