Remote Sensing and Control by Means of the Parallel Port, Java and Recycled e-Waste

Jose Alberto Hernández ,[1], Alvaro Zamudio Lara[1], Jesús Escobedo[1], Gennadiy Burlak[1], José Alfredo Hernández[1], David Juárez Romero[1], Yoel Ledo[2],Víctor Mendoza[2]

[1] Centro de Investigación en Ingeniería y Ciencias Aplicadas,
Universidad Autonoma del Estado de Morelos
Av Universidad No. 1000. Col Chamilpa, Cuernavaca, Morelos
jose_hernandez@uaem.mx
[2] Universidad de las Américas, A.C. cd. de México
Information Technology Deparment
Puebla No. 223. Col Cuauhtémoc, México, D.F.
yledo@udla.mx

Abstract. We discuss a prototype multiplatform system to perform remote sensing and control of physical variables i.e. temperature, based on the use of the parallel port, application software developed on java and gnu software, working on old recycled computers Pentium II (now considered as computer garbage or e-waste). In this paper, we analyze the electronic circuit that allows sensing until sixteen multiplexed different signals, and its introduction to computer by means of parallel port, we analyze the program on java that makes possible the execution of remote sensing and control over the Internet, as well as the system's implementation on old computers.

Keywords: Remote sensing, Java programming, computer recycling, e-waste

1 Introduction

Remote Sensing in the most generally accepted meaning refers to "instrument-based techniques employed in the acquisition and measurement of spatially organized (most commonly, geographically distributed) data/information on some property(ies) (spectral; spatial; physical) of an array of target points (pixels) within the sensed scene that correspond to features, objects, and materials, doing this by applying one or more recording devices not in physical, intimate contact with the item(s) under surveillance (thus at a finite distance from the observed target, in which the spatial arrangement is preserved); techniques involve amassing knowledge pertinent to the sensed scene (target) by utilizing electromagnetic radiation, force fields, or acoustic energy sensed by recording cameras, radiometers and scanners, lasers, radio frequency receivers, radar

[1] Please note that the LNCS Editorial assumes that all authors have used the western naming convention, with given names preceding surnames. This determines the structure of the names in the running heads and the author index.

systems, sonar, thermal devices, sound detectors, seismographs, magnetometers, gravimeters, scintillometers, and other instruments" [9].

On this paper we will focus on remote sensing of temperature by means temperature sensors, gnu free software and old recycled computers (considered as e-waste). In [1] and [4] is discussed the way to deal with electronic waste (e-waste), derived from domestic and office appliances like refrigerators, micro ovens, printers and computers, now considered as a health risk by the way these appliances are finally disposed in third world countries like India, several at Africa, China and Latin America, where mostly of e-waste is burned or recycled in uncontrolled environments contaminating the air, soil and water, with high contents of mercury and acids; given the high toxicity of these pollutants is considered as an emerging problem. But we prefer to see this problem as in [5] like an opportunity, not merely by the valuable materials in them like iron, copper, aluminium, gold and other metals, but specifically in the case of computers, like reusable components in useful applications.

Lets suppose we require automating the lecture of variables involved in some process, a common task in several laboratories, by force brute is required the waste of valuable time of high skilled human resources that spent most of their time in "dynamics" on which is required to measure each specific *xi* time several variables. Now consider extremely dangerous process difficult to measure like the resulting of acids manipulation or reactions producing high temperatures. These situations make us thinking about the quality of the lectures, the human error and the heavy and not value-added task that this activity can result for some researchers. Although there exists proprietary software (i.e. Labview) that allows the control of such process, the solution can result expensive and for some research projects prohibited due high license and training costs. If we focus just in the measure of single variables like temperatures, we could find at the internet some interesting approaches that deal with this problem. In [3] is discussed the use of GP-3 board, circuitry and C# to measure and control temperatures over the internet, but the system requires relatively new computers, third party hardware and .NET to work. In [7] is used recycled old computers, a multi meter and Linux, to treat with the same problem, despite code is multiplatform due the use of PERL, reach is limited by the use of single purpose multi meter that read merely temperatures and display lectures in a tiny web browser over a LAN.

1.1 Problem at hands

Considering the above exposed situation, is our desire to contribute with the design and implementation of a low cost hardware and software application that allows the monitoring and control of several digital signals (temperatures) by means of the parallel port -we remark that this port is available in most of old computers due it was used to send data to printers- and free gnu software, specifically Linux, Netbeans, Java and necessary drivers to manage this port.

On the first section of this paper, we discuss the proposed Client-Server technology by means of block diagrams; we discuss the electronic circuit that will multiplex the signals coming from different temperature sensors, we analyze the electronic diagram, and discuss its basic performance. Then we analyze the class diagram to manage the parallel port, to read the information from sensors as well as the writing of control sequences, in this section, we present the Java code employed and created for the application, at the end on this section we describe specific details about implementation (hardware and software) on old computers and the tuning of the application to work over the internet. We present our preliminary results and finally our conclusions.

2 The Client-Server technology

Java is actually quite extensive including capability for http, ftp, and tcp. The part of the Client-Server architecture that we will be using in this paper takes advantage of tcp communication and consists of two pieces, a server piece that reads the temperature (until sixteen different sensors) at the location we are interested (by means of a remote IP address), and a client piece that will remotely access and –if desire- to control the temperature.

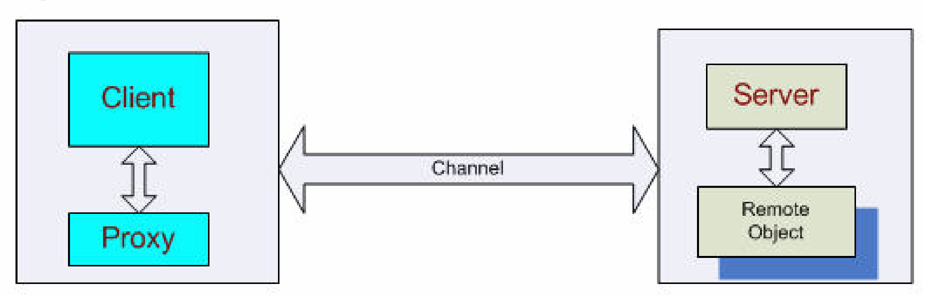


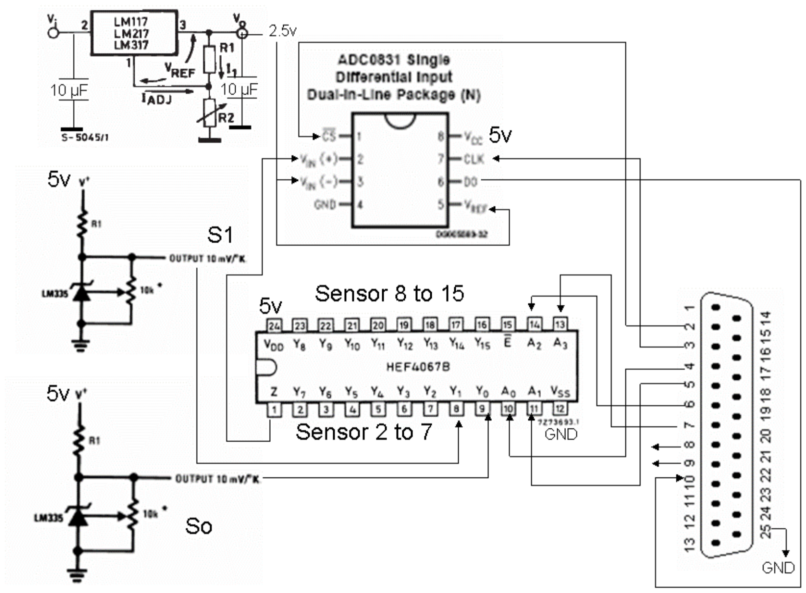
Fig. 1. Block Diagram of the proposed Client-Server Technology

The client piece in our application is generated by means of a remote terminal that runs a remote .JAR program which uses a timer that requests a temperature reading every 1 second from different sensors. Depending of Network configuration at Client Side, it uses a proxy to call the remote object on the Server side and the call is made as if the remote object were living on the same machine as the Client. The remote object in our example takes the form of an instance of a class. The server job is simply to register and run the content of the remote object (a .JAR program) and provide a channel (via SSH) to the remote object so that the Client knows about them. The Remote Object does all the important work by controlling and sensing the hardware through the circuit that we designed and implemented (see next section). To get a little more detailed idea of the architecture for the current project, take a look at the UML diagram in figure 4 showing the three individual assemblies needed to complete the remote picture. Note that the Client piece (on the left side) lives on the Client machine and the remote object and the server piece live on the Server machine.

2.1 Temperature sensing circuitry and the parallel port of the computer

Figure 2, shows the low cost implementation circuitry we designed, at left hand is located the sensors array, for simplicity purposes we use temperature sensors LM135 (-

55ºC to +155 ºC temperature range), at the center, the multiplexing module consisting of a HEF4067B, sixteen signals multiplexer; in the center at top, an eight bit resolution ADC (Analog to Digital Converter) 0831CCN connected to a voltage regulator LM317 that provides a voltage of reference Vref of 2.5 Volts. Finally at left hand the parallel port of the computer represented by a DB25 connector, lines are used to represent the cable bus required to monitor and control circuitry.



Fi g. 2. Circuit to read sixteen temperatures and control some parameters by means of parallel port

The array of sensors provides the sixteen input signals for the HEF4067B (Y0 to Y15) meanwhile parallel port control the selected multiplexed signal A2 to A5 (pins 4 to 7), the multiplexed signal is selected as the Vin to the ADC0831, which is going to be converted to an eight bit serial data, considering a Vref of 2.5Volts provide by LM·327 Voltage regulator. The ADC CS signal is enabled by the parallel port DO (pin 2), and the bit selection is controlled by the parallel port D1 (pin 3) by generating the clock for the ADC. Reading of converted signal ADC's DO is done at parallel port' ACK (pin 10). Ground is referenced at parallel port GND (pin 25). Control signals are provided by D6 y D7 (pins 8 and 9 respectively).

Figure 3 shows the implementation of the above analyzed circuitry on a proto board.

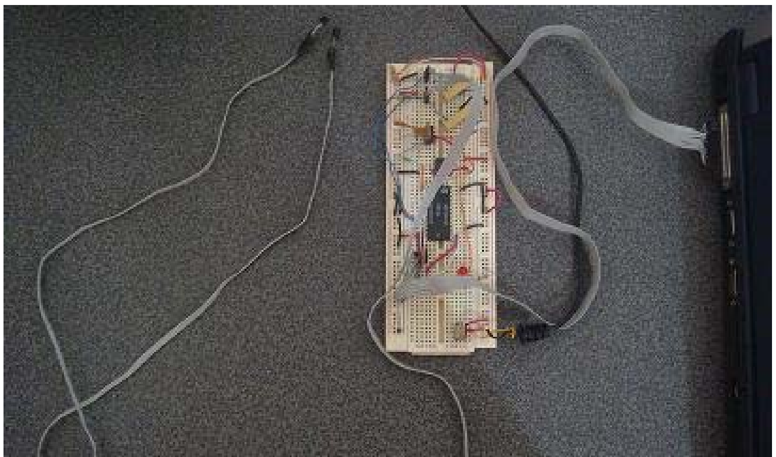


Fig. 3. The circuit implementation on a proto board. At left side a couple of sensors, at the center the multiplexer, voltage regulator and ADC converter, at right side the parallel cable

2.2 The application program

Figure 4, shows the class diagram for the proposed application. The core class of the application is the parallel port class which provides the methods to read or write from the parallel port, this class is used by the Main Class at Server side, which instantiates the methods to read from the parallel port information, and decodes data coming from sensors to be displayed in GUI as Centigrade or Fahrenheit grades. This class can be used to send data for the control of data reading process, as well as to control remote devices (i.e. alarm, fan or cooler).

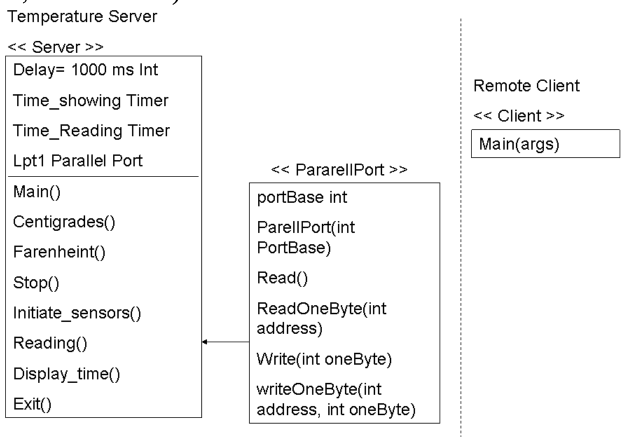


Fig. 4. The Class Diagram

2.2.1 The Java code

As we stated in previous section, the application consists of a Server, a Client and the parallel port class. First, we will analyze the parallel port class.

The Parallel Port class

This class is used in several applications due its simplicity and usability regarding Java libraries [6, 8]; it was used to control five axis arms for robots, remote control of cars, and to interface controllers and PC's [2]. This class is available to work in Windows and Linux environments. The code and libraries required are free to download from Internet. Bellow we show the class we used to read and write information from and to the parallel port.

The parallel port class from Del Cid Portillo [2]. Parallel Print Port Access Through Java

```
public class ParallelPort {
/** The port base address (e.g. 0x378 is base address for
LPT1) */
   private int portBase;
   /** To construct a ParallelPort object,
     * you need the port base address   */
   public ParallelPort (int portBase){
      this.portBase = portBase;
   }
/** Reads one byte from the STATUS pins of the parallel
port. The byte read contains 5 valid bits, corresponding
to 5 pins of input from the STATUS pins of the parallel
port (the STATUS is located at "portBase + 1", e.g. the
STATUS address for LPT1 is 0x379)
     *
     * This diagram shows the content of the byte:
     *
     *  Bit | Pin # | Printer Status   | Inverted
     * -----+-------+-----------------+-----------
     *   7  |  ~11  | Busy             |   Yes
     *   6  |   10  | Acknowledge      |
     *   5  |   12  | Out of paper     |
     *   4  |   13  | Selected         |
     *   3  |   15  | I/O error        |
     *
* Note that Pin 11 is inverted, this means that "Hi"
input on pin means 0 on bit 7, "Low" input on pin means 1
on bit 7.*/
   public int read ()
   {
      return ParallelPort.readOneByte(this.portBase+1);
   }

   /** Writes one byte to the DATA pins of parallel port.
located at the base address of the port (e.g. DATA
address for LPT1 is 0x378).
     *
     * This diagram shows how the byte is written:
     *
     *  Bit | Pin # | Printer DATA
     * -----+-------+--------------
     *   7  |   9   |   DATA 7
     *   6  |   8   |   DATA 6
```

```
      *   5  |   7   |    DATA 5
      *   4  |   6   |    DATA 4
      *   3  |   5   |    DATA 3
      *   2  |   4   |    DATA 2
      *   1  |   3   |    DATA 1
      *   0  |   2   |    DATA 0
    */
    public void write (int oneByte)
    {
       ParallelPort.writeOneByte (this.portBase, oneByte);
    }
    /** Reads one byte from the specified address.
      * (normally the address is the STATUS pins of the
 port)
      */
    public static native int readOneByte (int address);
    /** Writes one byte to the specified address
      * (normally the address is the DATA pins of the
 port)   */
    public static native void writeOneByte (int address,
 int oneByte);

    static
    {
       System.loadLibrary("parport");

    }
 }
```

Now we will show the core code for the Server application. This code uses the parallel port class, and creates the GUI to interact with the user; this GUI contains the timers that control remote monitoring of temperatures. For simplicity, we will show just the fragment of code where is possible to view the interaction of the interface with the parallel port class.

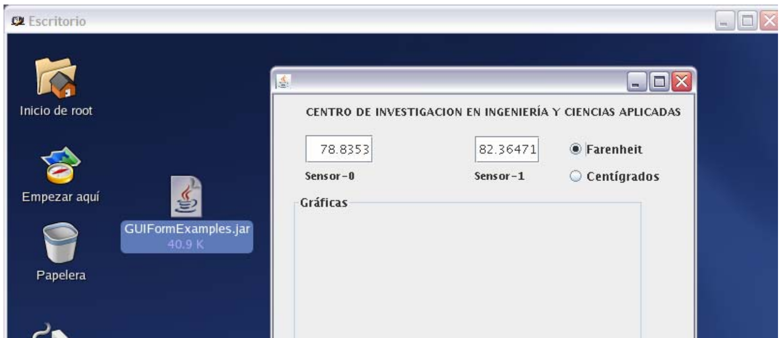
Fragment of the Server class focussing on the monitoring and control of temperature sensors implemented on Java.

```
ActionListener lecturaSensores = new ActionListener() {
public void actionPerformed(ActionEvent evt) {
//...Perform a task...
      lee_Sensor(0, jTextField1);
      lee_Sensor(1, jTextField2);

};
  // Reading of sensor
  // parameters int
  public void lee_Sensor(int iSensor, JTextField Control){
      int i, k, cuentas;
      double Voltaje, Temperatura;
      //DecimalFortmat DF = new DecimalFormat();
/*       ParallelPort lpt1 = new ParallelPort(0x378); 0x378 is
normally the base address for the LPT1 port
*/
    lpt1.writeOneByte(888, 1 + 4 * iSensor); //Actives the
converter
    lpt1.writeOneByte(888, 0 + 4 * iSensor);
    lpt1.writeOneByte(888, 2 + 4 * iSensor);// Clock or the
converter
    lpt1.writeOneByte(888, 0 + 4 * iSensor);
    cuentas = 0;
    for(i=0; i<=7; i++){
        lpt1.writeOneByte(888, 2 + 4 * iSensor);//Clock
        lpt1.writeOneByte(888, 0 + 4 * iSensor);//Error on i <-
isensor
        //k = (lpt1.readOneByte(889) & 64) / 64; // Read byte, but
only bit from S6
        // Read from higher bit to lower bit
        k = (lpt1.read() & 64) / 64; // Read byte, but only bit
from S6
        cuentas = cuentas + k * (int) Math.pow( 2, 7-i);
    }
    Voltaje = cuentas * ( 2.5 / 255);
    Temperatura = (Voltaje + 2.5) * 100 - 273; //Conversion to
Celcius degrees
    if (!jRadioButton2.isSelected())
        Temperatura = 1.8 * Temperatura + 32; // Calculation of
Fahrenheit degrees
Control.setText(String.valueOf((float)Temperatura));
  }//--- End of function Reading of sensor
```

2.2.2 The implementation

To implement the Client-Server System we used a recycled laptop Pentium II, with 196 MB in RAM Memory, 5 GB in HD, network interface card 100 Mbps and Modem ASDL for Broad Band Internet connection. We installed Linux Red Hat 9.0 as Server and enabled the httpd and ssh servers (web and secure shell servers) to enable web pages and remote desktops, we also installed the Java Virtual Machine 6.1 for Linux, the parport class and its respective library for linux, finally we installed the resulting .JAR file containing the GUI generated in Netscape 5.5.1.



CENTRO DE INVESTIGACION EN INGENIERÍA Y CIENCIAS APLICADAS

78.8353 Sensor-0

82.36471 Sensor-1

[✓ Farenheit] Centígrados

Gráficas

Fig. 5. The application running at remote client side

To achieve results shown on previous figure, first the client must open a session in Server side since the Client side, by means of SSH and Xming software to emulate a graphic terminal. The client will be asked for an specific user and password, once is introduced this information, the Client must open the desktop and execute the .JAR application by means of double click over the cup of coffee icon, or by typing the command java –jar GUIFormExamples.jar. The application will start the reading of temperatures at remote Server; the above example shows the information of two sensors expressed on Fahrenheit degrees or Celsius if selected.

2.2.3 Details for a successful implementation

We installed the software in Windows and Linux environments successfully. For Windows environments is mandatory to install the library –included on the download package from internet- parport.dll on the System32 directory. For Windows NT, Windows 2000 and XP, is necessary to install the useport.dll library [10] on the System32/drivers directory. In other hand, on Linux environment is mandatory to install the libpartport.so on the /lib directory as the root user. Finally, is necessary to calibrate each sensor individually by means a device considered like a standard (i.e. other temperature sensor from a well known supplier).

3 Preliminary results and discussion

We verify the performance of the application at local area network and remote locations and the application performance is OK, regarding the works of [2, 3, 7] our system is better, fist because is possible to read sixteen different temperature signals, second provides a multiplatform environment (runs perfectly well in windows based systems as well as Linux based systems); third, we reuse old computers contributing to diminish pollution of environment; four, we use gnu software which provides a low cost solution; five, is possible to control remote process, and finally we provide an open architecture that can be modified for a myriad of applications.

Of course we are aware of the system's limitations like the requirement of the internet, but the question is where is not available a kind of internet connection today? Another limitation is the maximum distance of the sensors from the parallel port, but we can solve this problem by using a different standard.

3.1 Further work

We are working on the improvement of the graphic user interface (GUI), we are considering the use of wireless sensors for monitoring temperature, relative humidity, and luminescence, and we would like to use our system in real world applications like green house monitoring and control, and the generation of expert system with sensing capabilities for the improving of diagnosis for plant diseases.

4 Conclusions

The proposed technology is useful, provides an open architecture to deal with different problems of remote monitoring and control, besides provides a low cost alternative thanks the use of gnu software and multiplatform code generated on Java, and, maybe more important, contributes to diminish earth pollution by recycling e-waste from computers.

Acknowledgments. We want to thank to CONACYT due part of this research is supported by project 47220.

References

1. Barba-Gutiérrez, Y.. Adenso-Díaz, B. and Hopp, M. An analysis of some environmental consequences of European electrical and electronic waste regulation. Resources, Conservation and Recycling, Volume 52, Issue 3, January 2008, Pages 481-495 (2008)
2. Del Cid Portillo, J.G. and Blank; J. Parallel Printer Port Access through Java. http://www.geocities.com/Juanga69/parport/ (2008)
3. Gold, M. Remote Sensing and Remote Control over the Internet with GP-3 Board, January16, 2004. Consulted in http://www.c-sharpcorner.com/UploadFile/mgold/RemoteSensing08292005100245AM/RemoteSensing.aspx (2004)
4. Hilty, L.M. Electronic waste—an emerging risk? Environmental Impact Assessment Review, Volume 25, Issue 5, July 2005, Pages 431-435 (2005)
5. Hischier, R., Wäger, P. and Gauglhofer, J. Does WEEE recycling make sense from an environmental perspective?: The environmental impacts of the Swiss take-back and recycling systems for waste electrical and electronic equipment (WEEE). Environmental Impact Assessment Review, Volume 25, Issue 5, July 2005, Pages 525-539 (2005)

6. Java Communications API http://java.sun.com/products/javacomm/index.jsp (2008)
7. Lapinskas, S. Remote Temperature Monitoring with Linux. February 23rd, 2006 Consulted in http://www.linuxjournal.com/article/8780 (2006)
8. New to Java Technology Archive - programming the parallel port. http://forum.java.sun.com/thread.jspa?threadID=519878&messageID=2483612 (2007)
9. Short, N. Remote Sensing Tutorial. http://rst.gsfc.nasa.gov/ (2008)
10. Userport http://www.embeddedtronics.com/public/Electronics/minidaq/userport/ (2007)