World Transactions on Engineering and Technology Education © 2007 UICEE
Vol.6, No.1, 2007

Digitally controlled resistors for the remote laboratory *NetLab*

Aaron Mohtar, Zorica Nedic & Jan Machotka

University of South Australia
Adelaide, Australia

ABSTRACT: The remote laboratory *NetLab* at the University of South Australia (UniSA), based in Adelaide, Australia, allows students to conduct experiments on real equipment over the Internet. It also allows users to remotely wire electrical circuits using a set of discrete components, such as resistors, capacitors, inductors, etc. In this article, the authors present the development of hardware and software of digitally controlled variable resistors that allows *NetLab* users to vary the circuit parameters in a similar way as if they were working in a real laboratory.

INTRODUCTION

NetLab is a remote online laboratory developed at the University of South Australia (UniSA) in Adelaide, Australia, to allow students to conduct experiments using real equipment over the Internet. It also allows users to remotely wire electrical circuits using a set of discrete components such as resistors, capacitors and inductors.

In the real laboratory, students often use typical resistor boxes, as shown in Figure 1, to manually modify the resistance values.

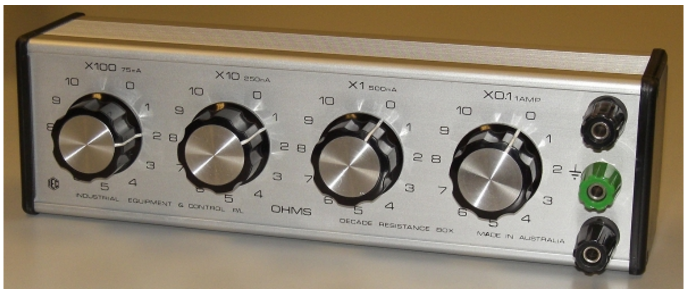


Figure 1: A typical resistor box used in laboratories [1].

In this article, the authors describe the design of microcontroller-based hardware that allows *NetLab* users to remotely modify the values of circuit components by interacting with their Graphical User Interfaces (GUIs).

This new development follows the main *NetLab* concept to keep experiments and components as real as possible by using photographic images of instruments and components for their GUIs. This allows students to interact with instruments and circuit components in the same way that they usually would in the real laboratory.

SYSTEM ARCHITECTURE

The variable resistor system consists of two separate modules. The first module is the actual variable resistor module whose resistance can be set digitally. The second module is the communication module, which is responsible for maintaining the communication between a computer and the multiple variable resistors that might be connected to the system as shown in Figure 2. Currently, the communication between the computer and the communication module utilises the RS232 protocol, while the common communication bus, between the communication module and multiple variable resistors, uses the Inter-Integrated Circuit (I^2C) protocol. I^2C is a high-speed serial communication protocol invented by Philips. It supports 7 and 10-bit addressing, has inbuilt feedback and can support data transfer rates of up to 1MHz.

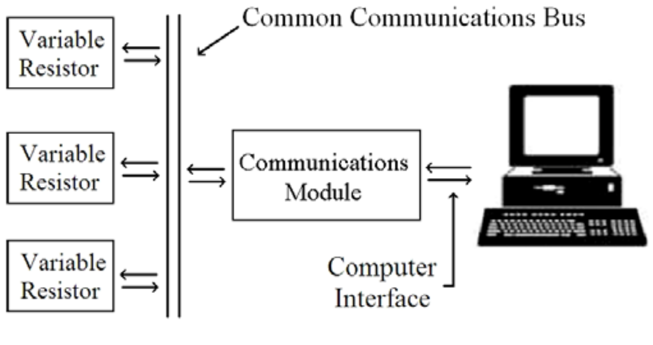


Figure 2: The system architecture.

Variable Resistor Module

As indicated, this module is a microcontroller-based hardware that allows resistance to be set digitally. This module accepts 4 bytes of data from the I^2C bus and then processes this data to control the output resistance of the module.

The module is meant to closely resemble a typical four-decade resistor box like that shown in Figure 1. As shown in Figure 3, each knob on the box is used to place N resistors in series where N is an integer from 0 to 10 to which the knob points. An arrangement of four of these sub-circuits in series results in a four-decade variable resistor box. The motion and operation of this rotation switch (knob) is emulated using relays, as shown in Figure 4.

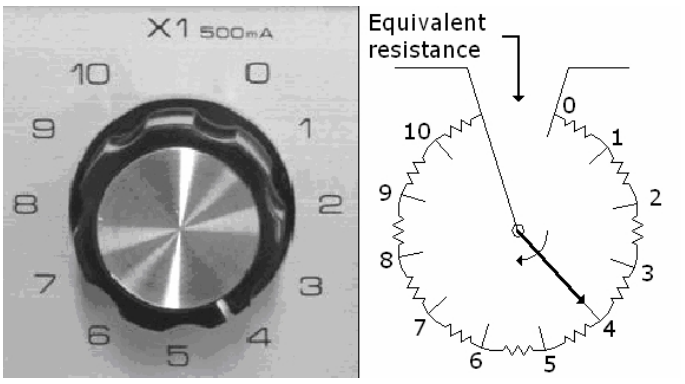


Figure 3: Knob and its operation [1].

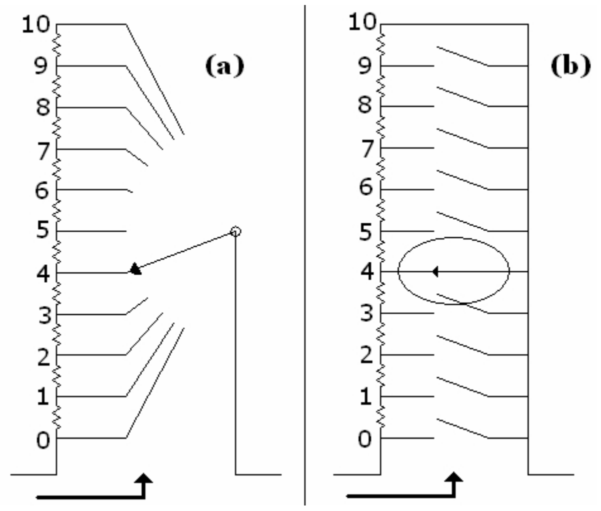


Figure 4: The operation of each knob (left) and its equivalent using mechanical relays (right) [1].

As the value of the resistance is not to be changed frequently, the relays are to stay in the same state (*on* or *off*) for extended periods of time; the latching type of relay, shown in Figure 5, was used to reduce the power consumption. This type of relay consumes power only when switching is required, after which the power can be cut off while the switch remains (latches) in place.

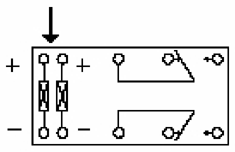


Figure 5: Double Pole Double Through (DPDT) latching relay with dual coil control [2].

So far, the circuit includes four knob-equivalent sub-circuits, and thus the components involved include: 4 x 10 resistors (rated at 1W) and 4 x 10 dual coil latching relays. To control the relays, driver circuits are required to drive the coils. This is accomplished by the use of the Micrel's MIC5842 driver IC. This IC can drive 8 coils, ie 4 dual coil relays and accepts serial input [3]. To control 80 coils (for the 40 relays), 10 Driver ICs are needed.

A microcontroller is used to determine which relays are to be switched *on* or *off* in order to achieve the required resistance value. The role of this microcontroller is to read 4 bytes of data off the I^2C bus, find the required states of the relays and then send a serial stream of data to the driver ICs to correctly set relays. The microcontroller selected for this role is Microchip's Pic16F819 [4]. Details of the operation of the microcontroller are discussed in the next section. Figure 6 shows the complete resistor board equivalent to a four-decade resistor box with the main components of the system labelled.

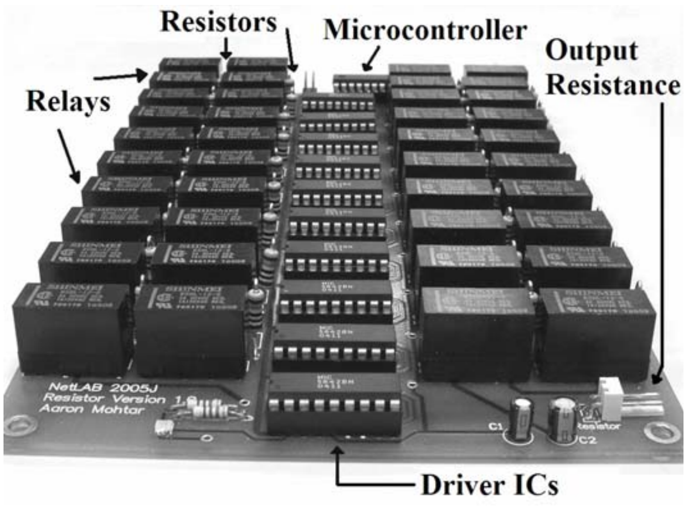


Figure 6: Completed four-decade variable resistor board.

Each variable resistor module includes one PIC16F819 microcontroller that has the role of monitoring the I^2C bus. Each resistor module has a unique address assigned to it. The microcontroller waits until it has detected its address and then reads 4 bytes of data off the bus. These 4 bytes represent the values of the four knobs of a typical resistor box. Once all data has been received, the microcontroller determines the right combination of switching that needs to take place in order to set the required resistance. Once this has happened, the microcontroller sends the data serially to the drivers ICs. If all the control data is sent successfully, the microcontroller enables the drivers outputs for a short period of time (50ms), so switching of the relays can take place with minimum power used. As the relays used are latching, they will maintain their position after the power has been disconnected. Figure 7 illustrates and summarises the flowchart of the digital four-decade variable resistor firmware.

Communication Module

The communication module was solely developed to isolate and encapsulate the communication between the computer and the variable resistor modules. This module accepts RS232 commands from a computer, and then sends them through to the variable resistors using the I^2C bus.

The hardware of this module mainly consists of a 40-pin microcontroller; the PIC16F877. In addition to this, a MAX232 IC is used to convert the voltage of the RS232 port to TTL compatible levels. This module also has some additional relays to control the power delivered to the variable resistors. Thus, when no data is to be transferred, the resistor board power is cut-off, leaving only the communication module powered in order to further reduce the power consumption of the system.

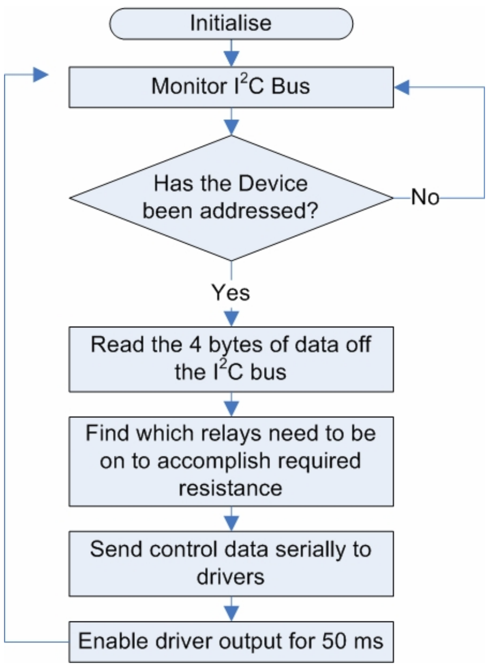

Initialise
Monitor I^{2}C Bus
Has the Device been addressed?
No
Yes
Read the 4 bytes of data off the I^{2}C bus
Find which relays need to be on to accomplish required resistance
Send control data serially to drivers
Enable driver output for 50 ms

Figure 7: Flowchart of the PIC16F819 firmware.

The communication module's function is mainly based on the firmware programmed into the PIC16F877 microcontroller. The microcontroller monitors its RS232 port and then places data on the I^{2}C bus. The RS232 communication bus, between the microcontroller and the host computer, transfers data at a rate of 57.6kbps, while the I^{2}C bus is running at a speed of 100 KHz and uses 7-bit addressing.

Figure 8 shows a flowchart of the communication module. The microcontroller monitors and waits until a byte of data has been sent through the RS232 port. Once that has happened, it waits and reads another 4 bytes off the port, ie 5 bytes in total. The first byte is the address of the variable resistor whose values is to be set, while the other 4 bytes are the numbers that represent the values of the knobs of a typical resistor box. As soon as all the bytes have been received, the controller sends the first byte (address byte) to the I^{2}C bus and waits for a response. If no response is received, ie a device with such an address does not exist, the controller sends back to the host computer feedback to indicate that the device is not connected. If a positive response has been received, the controller continues to send the remaining 4 bytes (checking acknowledgement after each byte) and sends *write successful* feedback to the computer. However, if any of the 4 bytes is not acknowledged, sending is aborted and the message *write unsuccessful* is sent to the computer.

Host Computer Software

The remaining link in this system is the host computer, which is required to communicate with the communication module through RS232. This process involves configuring the serial port with the correct settings and then sending the data in the suitable sequence. This has been accomplished in two different programs, which are detailed below.

The first program uses the *LabVIEW* graphical programming language developed by National Instruments [5]. A VI (Virtual Instrument) has been developed for the variable resistor system to allow users who are familiar with *LabVIEW* to easily customise the user interface of this device.

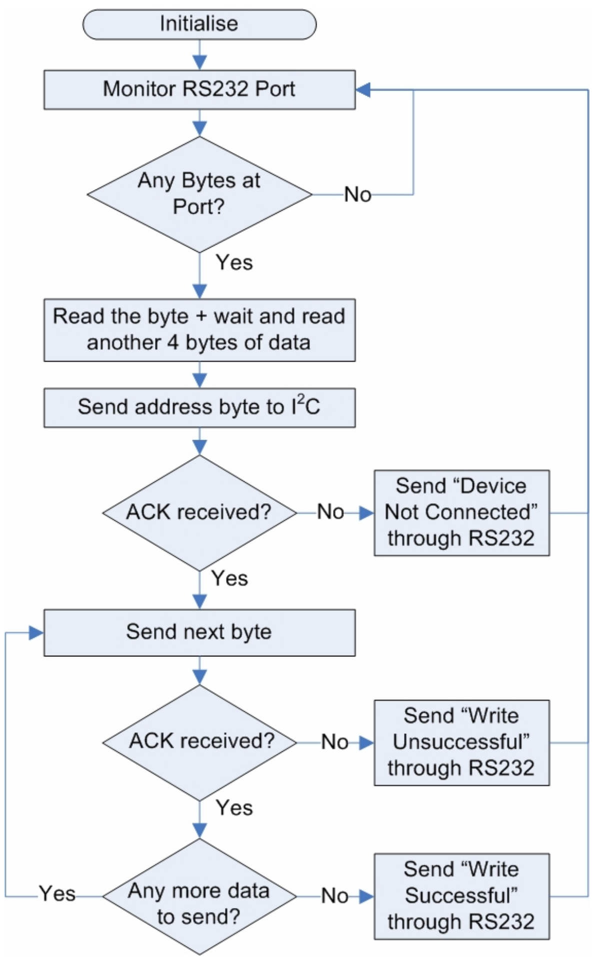
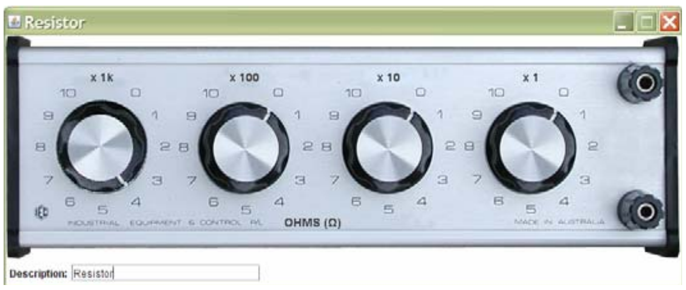

Initialise
Monitor RS232 Port
Any Bytes at Port?
No
Yes
Read the byte + wait and read another 4 bytes of data
Send address byte to I^{2}C
ACK received?
No
Send "Device Not Connected" through RS232
Yes
Send next byte
ACK received?
No
Send "Write Unsuccessful" through RS232
Yes
Yes
Any more data to send?
No
Send "Write Successful" through RS232

Figure 8: Flowchart of the PIC16F877.

The second program is written in Java and its role is to communicate directly with the serial port. Experienced Java users are able to easily customise this solution to suit their application needs. Figure 9 shows the GUI that was developed using Java. It can be noted that this GUI closely resembles the typical resistor box that is shown in Figure 1.



Description: Resistor

Figure 9: The GUI resembling the typical resistor box.

Figure 10 shows the *NetLab* GUI used by students to perform an experiment. The top left-hand side corner shows how a simple circuit can be wired using the Circuit Builder software developed for the remote wiring of real components into circuits in the *NetLab* environment [6]. Circuit Builder provides an option to use fixed resistors or variable resistors. Fixed value resistors are distinguished by their red labels from the variable resistors which have black labels. A double-click on a variable resistor activates resistor GUI which is also shown in Figure 10.

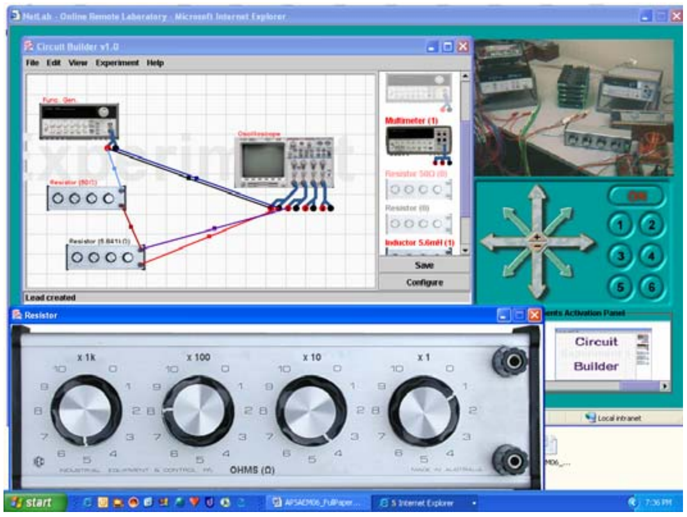


Figure 10: *NetLab* with Circuit Builder and variable resistor GUI.

The photographic image of a four-decade resistor box is used to give students a sense of reality when setting the resistor value. Students use a mouse to turn around animated knobs in order to set the resistance to a required value. Once set, the resistance value appears in the label of the resistor box within the Circuit Builder. Students can then configure the circuit and perform measurements on real equipment in a distance laboratory.

CONCLUSION

Four resistor boards of different ranges have been developed that cover resistance values between 1Ω and 10MΩ. The system has been built in a way to allow flexibility and easy upgradeability. The proposed architecture of the system isolates all communication with the host computer from the individual variable components. Consequently, if needed, a change of the computer interface will only affect the communication module.

The system is now in the process of being extended to include variable capacitors and inductors. Also, the interface change from RS232 to USB is progressing in order to enhance the communication speed with variable devices.

However, this proposed design has far more flexibility and can allow the addition of many electrical and electronic components, eg adjustable gain amplifiers, adjustable supplies, etc. With minor modifications, the resistor board can also be used as a 2-layer 4 x 10 matrix switch. This flexibility allows the system to be used as a base for various applications in the field of digital and remote control.

REFERENCES

1. Mohtar, A., Nedic, Z. and Machotka, J., The latest developments for remote laboratory *NetLab*. *Proc. 9th UICEE Annual Conf. on Engng. Educ.*, Muscat, Oman, 165-168 (2006).
2. Shinmei, RSB Relay Data Sheet (2004), http://info.tactnet.co.jp/shinmei/e/product/pdf/E_RSB.pdf
3. Micrel, MIC5841/5842, 8-Bit Serial-Input Latched Drivers (1998), http://www.micrel.com/_PDF/mic5841.pdf
4. Microchip, PIC16F818/819 Data Sheet, 18/20-Pin Enhanced Flash Microcontrollers with nanoWatt Technology. Chandler: Microchip Technology (2004), http://ww1.microchip.com/downloads/en/DeviceDoc/39598e.pdf
5. National Instruments, *LabVIEW*: 20 Years of Innovation (2005), http://www.ni.com/labview/
6. Nedic, Z., Machotka, J., Sprok, A., Ruud, L.O. and Carr, S., The circuit builder for *NetLab*. *Proc. UICEE 8th Annual Conf. on Engng. Educ.*, Kingston, Jamaica, 239-242 (2004).