VOL. 13, NO. 5, MARCH 2018 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

REMOTE LABORATORIES USING THE TRAINING MODULEM2CI

Diego F. Sendoya-Losada[1], Pedro Torres Silva[2] and Fabián Bolívar Marín[2]
[1]Department of Electronic Engineering, Faculty of Engineering, Surcolombiana University, Neiva, Huila, Colombia
[2]School of Basic Sciences, Technology and Engineering, National Open and Distance University, Bogotá, Colombia
E-Mail: diego.sendoya@usco.edu.co

ABSTRACT

Training module M2CI is a system that allows undergraduate students to acquire control engineering skills and competencies related to the automation of processes. The M2CI has several sensors and actuators for interacting with temperature, position and liquid level plants. However, many times the students do not have the economical means to travel or do not have time to use the equipment on the schedules in which the university attends the individuals, causing an underutilization of the M2CI. In order to make better use of the M2CI, a system to control and monitor the plants and instruments remotely has been implemented. This article presents the design of hardware and software to perform laboratories using the training module M2CI remotely via internet. The hardware and software of the project is based on Arduino, with the purpose of obtaining an economic solution.

Keywords: arduino, E-learning, engineering education, remote laboratories.

1. INTRODUCTION

Training module M2CI (Figure-1) is a system that allows both undergraduate students and professionals to acquire control engineering skills and competencies related to the automation of processes. The M2CI has several sensors and actuators for interacting with temperature, position and liquid level plants. Currently the National Open and Distance University (UNAD, for its acronym in Spanish), has three training modules M2CI located in Bogotá, Bucaramanga and Neiva cities. Engineering students who wish to practice with this system should be addressed to any of these three cities. However, many times the students do not have the economic means to travel to these cities or do not have time to use these equipment on the schedules in which the UNAD attends the individuals, causing an underutilization of the M2CI.

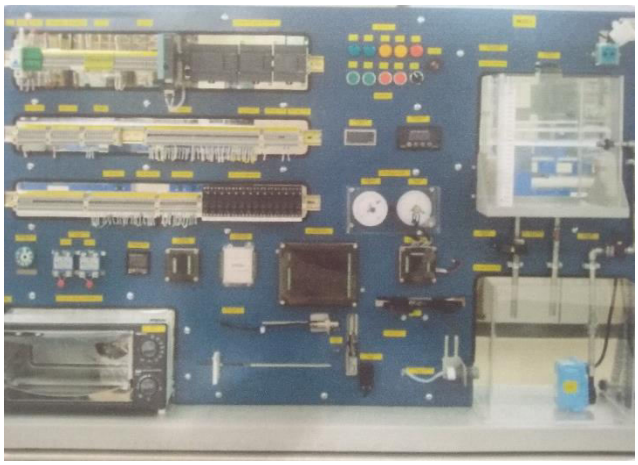

Figure-1. Training module M2CI.

In order to make better use of the M2CI, UNAD has implemented systems to control and monitor the different plants and instruments remotely, via internet. Globally, there are different evidences showing the increase that the use of remote laboratories has had in traditional higher education and distance learning [1 – 4]. In a previous contribution, a system for automatically wiring the M2CI module using synchronous serial communication was developed [5]. This system has served as a basis for wiring, monitoring and control of the M2CI via internet. The hardware and software of the project is based on Arduino, which makes it economical.

2. HARDWARE DESIGN

The Arduino UNO board has 14 pins, which can be programmed as digital inputs/outputs [6]. However, for this project it is required the control of 256 digital outputs, which should give a value of 0 or 1, depending on the information provided via internet to the Ethernet shield and then to the Arduino board. Below are the different sections of hardware.

2.1 Arduino UNO board

The Arduino UNO board is responsible for receiving, processing and sending 256 bits, corresponding to the different M2CI connections. Information is entered as packets of 256 bits. The "high" state of each bit represents the respective connection that should be activated. Moreover, the "low" state represents the indicated connection that should be disabled. Once the 256 connections are established, these remain without change until the user sends another packet of 256 bits. The information is entered using an interface designed in LabVIEW and is received by the Ethernet shield. The communication with the Ethernet shield is done by using pins 10, 11, 12 and 13 of the Arduino UNO board.

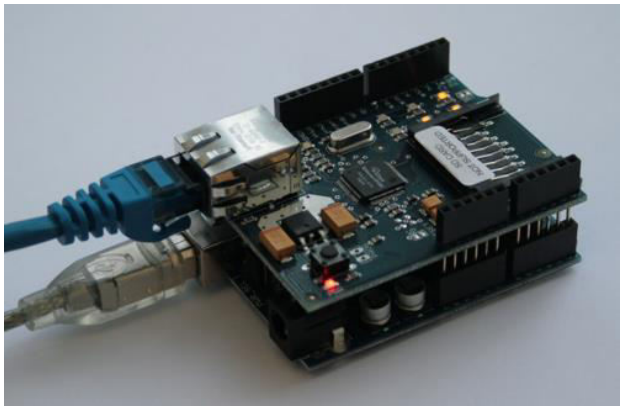

Figure-2. Arduino UNO and ethernet shield.

2.2 Shift registers

Once the Arduino UNO processes the information received from the Ethernet shield, the 256 bits are divided into 32 packets of 8 bits, which are placed on the output pins 2, 3 and 4. To achieve this, the chip 74HC595 is used (Figure-3). The datasheet refers to the 74HC595 as an "8 bits serial-in, serial or parallel-out shift register with output latches; 3-state" [7]. In other words, it can be used to control 8 outputs at the same time while only a few pins on Arduino UNO board are taken [8].

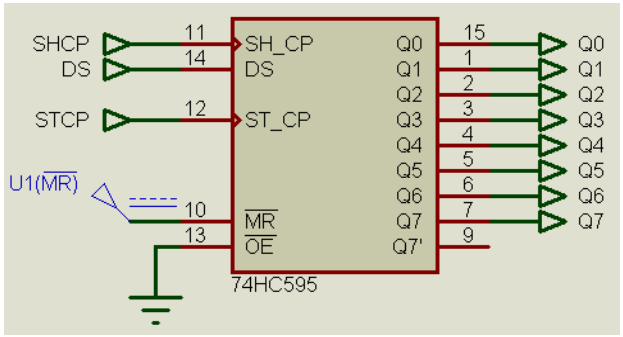

Figure-3. 8 bits serial-in parallel-out shift register.

The function table of 74HC595 indicates everything of significance that happens on a rising edge. When *clockPin* goes from *LOW* to *HIGH*, the shift register reads the status of the data pin. The displaced data is stored in a register of internal memory. When *latchPin* goes from *LOW* to *HIGH*, the sent data moves from the aforementioned memory register to the output pins.

2.3 D-Type latches

It can be seen that using a smaller amount of output pins of the Arduino UNO board, 8 bits data can be sent. If D-type latches are used as, for example, 74HC573 (Figure-5), then Arduino UNO board can send 32 packets of 8 bits serially and each packet can be routed to the 74HC573 of interest [9]. Thus, the output of each 74HC573 will keep the information even if the set of input bits changes.

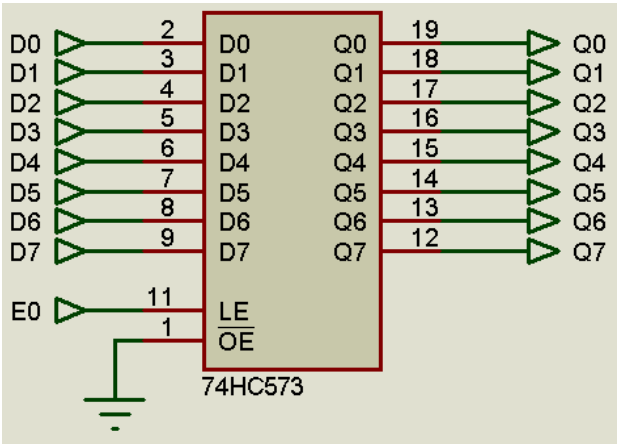

Figure-4. Octal D-type latch.

The LE pin of 74HC573 allows data transfer from the input to the output. Therefore, if a 1 is placed on this pin, the 8 bits data from the register 74HC595 is transferred to the output of the 74HC573. If a 0 is placed on the LE pin, the output is not changed, even if the input is changed. In this way, the Arduino UNO board can send, by using serial communication, 32 packets of 8 bits to the register 74HC595, and the output of this chip is simultaneously connected to the input of 32 latches 74HC573. The LE pin of each 74HC573 is used to transfer only the 8 bits that corresponds to them.

2.4 Decoding/Demultiplexing

In order to select the 32 D-type latches, two 4-to-16 lines decoder/demultiplexer are used, 74HC154 [10]. By using a binary code of 4 bits in the inputs of 74HC154, one of its 16 outputs is selected (2^4 = 16). Therefore, a binary code of 5 bits and two chips 74HC154 are required to activate 32 outputs (2^5 = 32), which are connected to the LE pins of the latches 74HC573. However, instead of using 5 Arduino's digital outputs, the data are sent, using serial communication via another register 74HC595, whereby the number of pins used to route each packet of 8 bits is reduced from 5 to 3.

Digital outputs 2, 3 and 4 of the Arduino UNO board are used to send 32 packets of 8 bits via serial to the first register 74HC595. The output data are simultaneously placed on the inputs of the 32 D-type latches. Digital outputs 5, 6 and 7 of the Arduino UNO board, send serially 5 bits to the second register 74HC595 in order to address the 32 D-type latches. The output of this second register goes to a D-type Flip-Flop, 74HC574 [11], which allows addressing the corresponding D-type latch, once the appropriate data packet is received. To perform this function, the pin 8 of the Arduino UNO board is used to generate the clock signal that allows this synchronization. Once the 5 bits of addressing appear on the inputs of the chips 74HC154, they activate the corresponding output and enable the appropriate D-type latch. As the output pins of the 74HC154 are active-low, and the enable inputs of the D-type latches operate with a high level, each output of the 74HC154 must be passed through an inverter that allows adjusting these logic levels [12].

The outputs of chips 74HC154 are responsible of enabling or disabling the 32 D-type latches by using LE pin. The data from the first register 74HC595 arrive simultaneously to all D-type latches. The four subcircuits labeled as 74HC573-1, 74HC573-2, 74HC573-3 and 74HC573-4 contain the same hardware, and their only difference is that the LE signal comes from a distinct output of the subcircuit 74HC154.

2.5 Output relays

Once the information is present on the output of the D-type latches 74HC595, it is transferred to 32 modules, each one of 8 relays, which are compatible with Arduino (Figure-5).

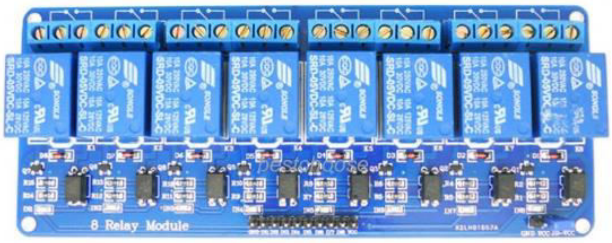

Figure-5. 8-channel relay module.

Each relay is connected to one of the 256 M2CI connections. In this way, the relays are responsible for connecting or disconnecting the connections present in the M2CI. Some of these connections carry logic levels of 0 or 5 volts, others are responsible for the supply of sensors and actuators (12 volts), and others carry signals of alternating current (120 VAC).

3. SOFTWARE DESIGN

As mentioned in the previous section, the information is entered using an interface designed in LabVIEW and is received by the Ethernet shield via internet. By means of serial communication and multiplexing, 256 digital outputs can be controlled only with 7 pins of the Arduino UNO board - from 2 to 8.

The interface developed in LabVIEW is presented below.

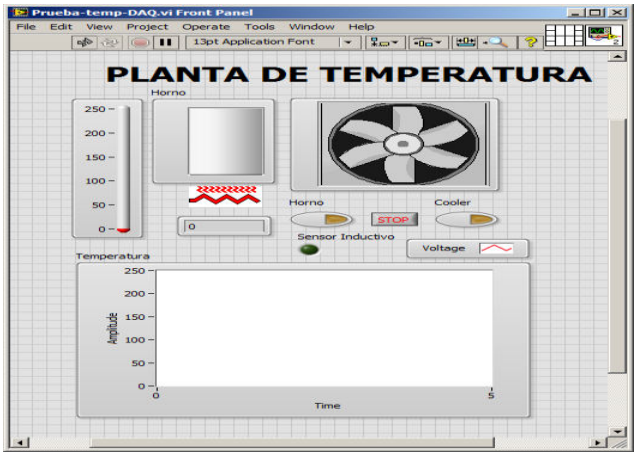

Figure-6. LabVIEW interface for temperature plant.

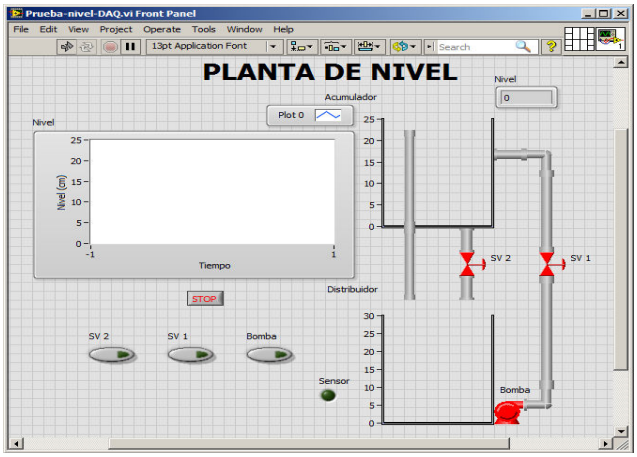

Figure-7. LabVIEW interface for level plant.

Figure-8. LabVIEW interface for position plant.

The Arduino code is presented below:

```
#include<SPI.h>
#include<Ethernet.h>
// Enter a MAC address and IP address
for your controller below.
// The IP address will be dependent on
your local network:
bytemac[]={0xDE,0xAD,0xBE,0xEF,0xFE,0xE
D};
IPAddressip(192,168,1,200);
// Initialize the Ethernet server
library
// with the IP address and port you
want to use
// (port 8000 is default for
Shoutcast):
EthernetServerserver(8000);
//Pin connected to latch pin (ST_CP) of
74HC595
constintlatchPin1=3;
//Pin connected to clock pin (SH_CP) of
74HC595
constintclockPin1=4;
```

```
////Pin connected to Data in (DS) of
74HC595
constintdataPin1=2;
//Pin connected to latch pin (ST_CP) of
74HC595
constintlatchPin2=6;
//Pin connected to clock pin (SH_CP) of
74HC595
constintclockPin2=7;
////Pin connected to Data in (DS) of
74HC595
constintdataPin2=5;
////Pin connected to CLK of 74HC574
constintclkpin=8;
constbytenumDatos=32;
bytedato[numDatos];
voidsetup(){
//set pins to output because they are
addressed in the main loop
pinMode(latchPin1,OUTPUT);
pinMode(dataPin1,OUTPUT);
pinMode(clockPin1,OUTPUT);
pinMode(latchPin2,OUTPUT);
pinMode(dataPin2,OUTPUT);
pinMode(clockPin2,OUTPUT);
pinMode(clkpin,OUTPUT);
// Open serial communications:
Serial.begin(9600);
// start the Ethernet connection and
the server:
Ethernet.begin(mac,ip);
server.begin();
Serial.print("server is at ");
Serial.println(Ethernet.localIP());
}
voidloop(){
// listen for incoming clients
EthernetClientclient=server.available()
;
if(client)
{
Serial.println("new client");
while(client.connected())
{
if(client.available())
{
for(inti=0;i<numDatos;i++)
{
dato[i]=client.parseInt();
digitalWrite(latchPin2,LOW);
shiftOut(dataPin2,clockPin2,MSBFIRST,i)
;
digitalWrite(latchPin2,HIGH);
digitalWrite(clkpin,LOW);
digitalWrite(clkpin,HIGH);
digitalWrite(clkpin,LOW);
digitalWrite(latchPin1,LOW);
shiftOut(dataPin1,clockPin1,MSBFIRST,da
to[i]);
digitalWrite(latchPin1,HIGH);
Serial.println(dato[i],BIN);
}
}
}
client.stop();
Serial.println("client disconnected");
}
}
```

4. RESULTS AND DISCUSSIONS

In order to verify the operation of the designed system a test is performed. The user uses the LabVIEW interface to remotely wire the module connections. In this case the connections made correspond to the data "1", "20" and "240".

The first data received by Ethernet shield is "1", which is "00000001" in binary. This first data is addressed to register 0 by using the decoder/demultiplexer (Figure-9).

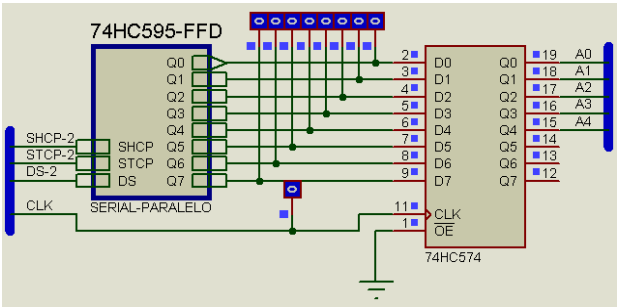

Figure-9. Addressing to register 0.

This addressing allows that "00000001" is transferred to the first group of outputs Q0-Q7, where Q7 in this case corresponds to the MSB and Q0 is the LSB. As shown in Figure-10, none other output suffers modification.

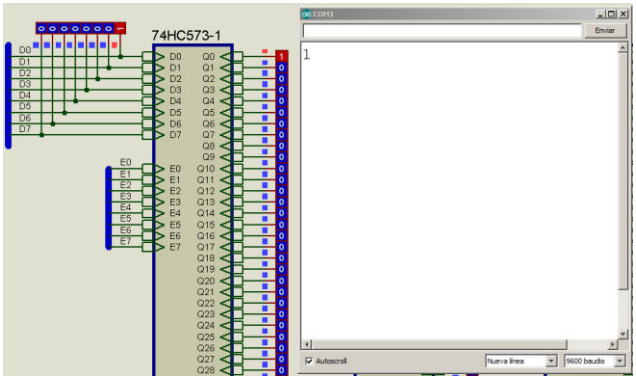

Figure-10. Test with data 00000001.

The second data received by Ethernet shield is "20", which is equivalent to "00010100" in binary. This data is addressed to the register 1, by using the decoder/demultiplexer (Figure-11).

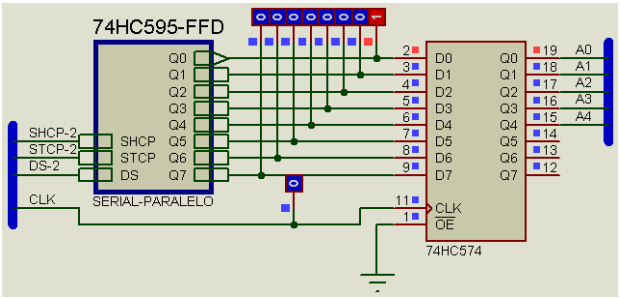

Figure-11. Addressing to register 1.

This addressing allows that "00010100" is transferred to the second group of outputs Q8-Q15, where Q15 corresponds to the MSB and Q8 is the LSB. As shown in Figure-12, the first group of 8 bits is not affected when the transference of information is made, only the second group of outputs is modified.

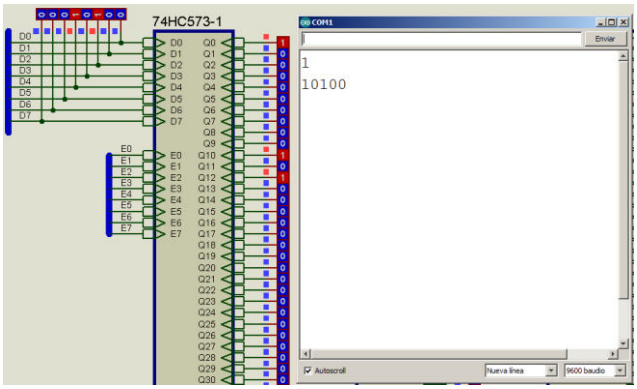

Figure-12. Test with data 00010100.

The last data used for the test is "240", which is equivalent to "11110000" in binary. This data is addressed to register 2 by using the decoder/demultiplexer (Figure-13).

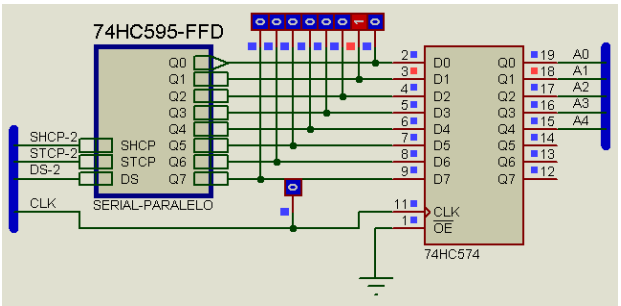

Figure-13. Addressing to register 2.

This addressing allows to "11110000" to be transferred to the third group of outputs Q16-Q23, where Q23 corresponds to the MSB and Q16 is the LSB. As shown in Figure-14, the two first groups of 8 bits are not altered when the transference of information is made, only the third group of outputs is modified.

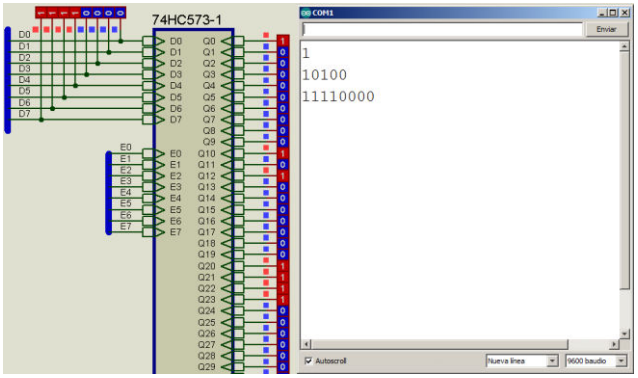

Figure-14. Test with data 11110000.

When the user continues making connections in LabVIEW, these are converted to 8-bit data that is transferred to the next group of outputs, while the previous output groups do not change. Completing the transfer of 32 packets of 8 bits, the system is enabled to receive another set of 256 bits, modifying all the previously connected outputs. In this way, the design of an economical remote laboratory system has been achieved which allows, through the use of simple hardware and software, to automatically establish the 256 M2CI connections.

REFERENCES

[1] Santana I., Ferre M., Izaguirre E., Aracil R. & Hernandez L. 2013. Remote laboratories for education and research purposes in automatic control systems. Industrial Informatics, IEEE Transactions on. 9(1): 547-556.

[2] Esquembre F. 2015. Facilitating the Creation of Virtual and Remote Laboratories for Science and Engineering Education. IFAC-Papers On Line. 48(29): 49-58.

[3] Kaluz M., Garcia-Zubia J., Fikar M. & Cirka, L. 2015. A flexible and configurable architecture for automatic control remote laboratories. Learning Technologies, IEEE Transactions on. 8(3): 299-310.

[4] Menacho A., Castro M. & Gil R. 2016, February. Competency-based learning management systems: Practices using remote laboratories to improve the use of the subjects and get required competences. In: 2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV) (pp. 109-111). IEEE.

[5] Sendoya-Losada D. F., Torres Silva P. and Perez Waltero H. 2016. Automatic wiring system applied to the training module M2CI, ARPN Journal of

Engineering and Applied Sciences. 11(19): 11503-11513.

[6] Arduino - Arduino Board Uno. (n.d.). Retrieved May 06, 2016, from https://www.arduino.cc/en/main/arduinoBoardUno.

[7] 74HC_HCT595 [Pdf]. (2016, February 25). NXP Semiconductors.

[8] Arduino - Shift Out. (n.d.). Retrieved May 06, 2016, from https://www.arduino.cc/en/Tutorial/ShiftOut

[9] 74HC_HCT573 [Pdf]. (2016, March 4). NXP Semiconductors.

[10] 74HC_HCT154 [Pdf]. (2016, February 29). NXP Semiconductors.

[11] 74HC_HCT574 [Pdf]. (2016, March 4).NXP Semiconductors.

[12] 74HC_HCT04 [Pdf]. (2015, November 27). NXP Semiconductors.