VOL. 11, NO. 19, OCTOBER 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2016 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

AUTOMATIC WIRING SYSTEM APPLIED TO THE TRAINING MODULE M2CI

Diego F. Sendoya-Losada[1], PedroTorres Silva[2] and Harold Pérez Waltero[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 is being implemented. As a part of this project, an automatic wiring system should be designed. In the market, equipment that allow the connection of multiple points automatically can be found; however, the high cost of these devices is a disadvantage. This article presents the design of hardware and software for a system that allows making and keeping the M2CI connections by using synchronous serial communication. The hardware and software of the project is based on Arduino, which makes it economical in comparison with the existing ones in the market. This contribution serves as a reference for a future development that will allow the automatic connection via internet.

Keywords: automatic wiring, 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 means to travel to these cities or donot 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 is implementing 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 order to remotely monitor and control the M2CI, it should be designed a system that allows automatically wiring the different components. The M2CI has about 256 independent connections, which can be carried out depending on the number of instruments, sensors and actuators that are wanted to work simultaneously. In the market, equipment that allow the connection of 256 points automatically can be found; however, the high cost of these devices is a disadvantage [5].

This article presents the design of hardware and software for a system that allows making and keeping the 256 M2CI connections, depending on the needs of the individual, by using synchronous serial communication. The hardware and software of the project is based on Arduino, which makes it economical in comparison with the existing ones in the market. This contribution serves as a reference for a future development that will allow the automatic connection via internet.

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 serial 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 32 packets of 8 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 32 packets of 8 bits. The user enters information by using the serial monitor of the Arduino software. On the same monitor, a response of confirmation, detailing which bits were activated and which bits were deactivated, can be seen. The communication with the serial monitor is done by using pins 0 and 1 of the Arduino UNO board (Figure-2).

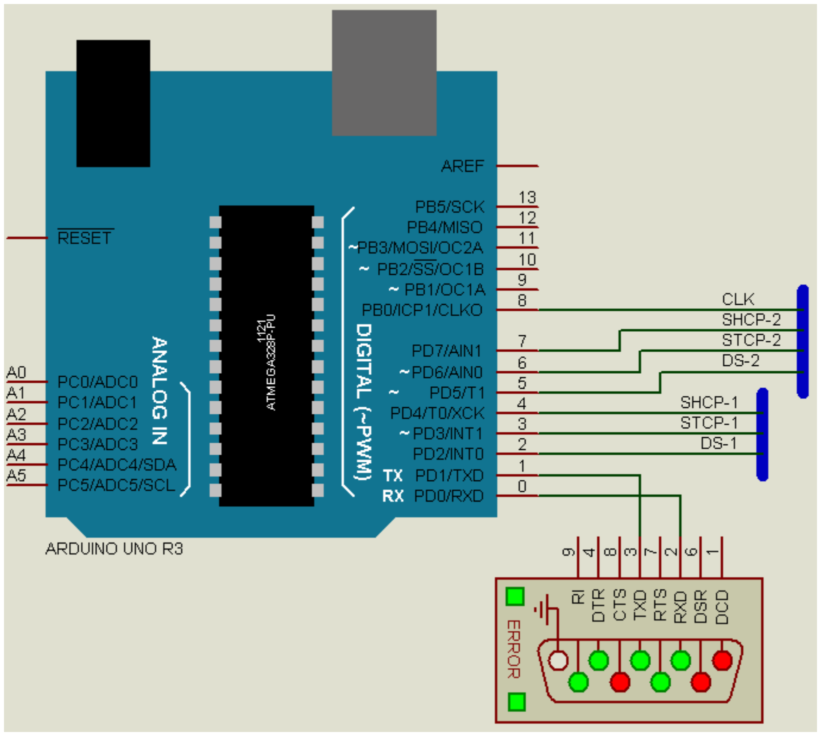


Figure-2. Serial communication with Arduino UNO board.

2.2 Shift registers

Once the Arduino UNO processes the information received from the serial monitor, the 32 packets of 8bits 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].

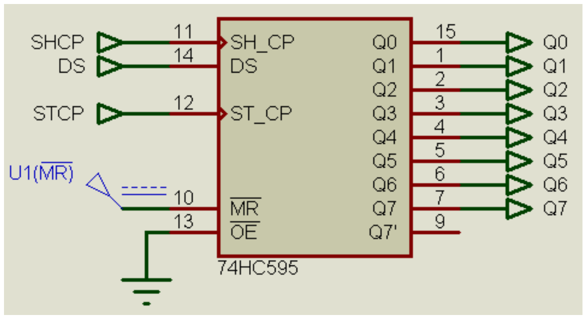


Figure-3. 8 bits serial-in parallel-out shift register.

In Table-1, the pin description of 74HC595 is shown. In addition, datasheet also provides other important information: The timing diagram (Figure-4) and the function Table (Table-2).

Table-1. Pin description of 74HC595.

Symbol	Pin	Description
Q1	1	parallel data output 1
Q2	2	parallel data output 2
Q3	3	parallel data output 3
Q4	4	parallel data output 4
Q5	5	parallel data output 5
Q6	6	parallel data output 6
Q7	7	parallel data output 7
GND	8	ground (0 V)
Q7S	9	serial data output
$\overline{\text{MR}}$	10	master reset (active LOW)
SHCP	11	shift register clock input
STCP	12	storage register clock input
$\overline{\text{OE}}$	13	output enable input (active LOW)
DS	14	serial data input
Q0	15	parallel data output 0
V_{CC}	16	supply voltage

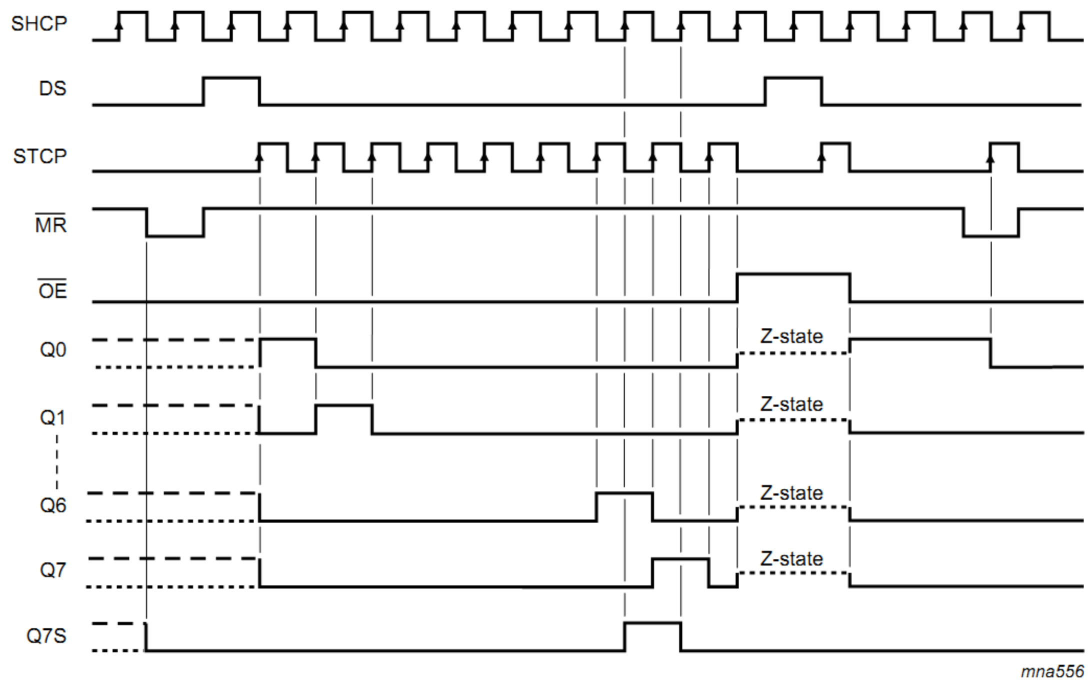


Figure-4. Timing diagram of 74HC595.

Table-2. Function table of 74HC595.

Control				Input	Output		Function
SHCP	STCP	$\overline{OE}$	$\overline{MR}$	DS	Q7S	Qn	
X	X	L	L	X	L	NC	a LOW-level on $\overline{MR}$ only affects the shift registers
X	↑	L	L	X	L	L	empty shift register loaded into storage register
X	X	H	L	X	L	Z	shift register clear; parallel outputs in high-impedance OFF-state
↑	X	L	H	H	Q6S	NC	logic HIGH-level shifted into shift register stage 0. Contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6S) appears on the serial output (Q7S).
X	↑	L	H	X	NC	QnS	contents of shift register stages (internal QnS) are transferred to the storage register and parallel output stages
↑	↑	L	H	X	Q6S	QnS	contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages

The function table 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, 8bits 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.

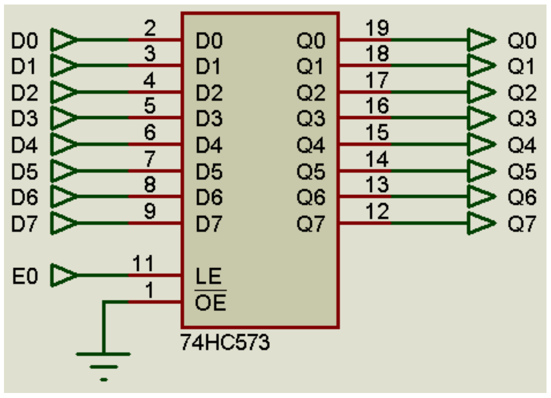


Figure-5. Octal D-type latch

The pin diagram and the function table of 74HC573 is presented in Table-3 and Table-4.

Table-3. Pin description of 74HC573.

Symbol	Pin	Description
$\overline{OE}$	1	3-state output enable input (active LOW)
D[0:7]	2, 3, 4, 5, 6, 7, 8, 9	data input
GND	10	ground (0 V)
LE	11	latch enable input (active HIGH)
Q[0:7]	19, 18, 17, 16, 15, 14, 13, 12	3-state latch output
V_{CC}	20	supply voltage

Table-4. Function table of 74HC573.

Operating mode	Control		Input	Internal latches	Output
	OE	LE	Dn		Qn
Enable and read register (transparent mode)	L	H	L	L	L
			H	H	H
Latch and read register	L	L	l	L	L
			h	H	H
Latch register and disable outputs	H	L	l	L	Z
			h	H	Z

It can be observed that 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

To select the 32 D-type latches, two 4-to-16 lines decoder/demultiplexer are used, 74HC154 [10]. The pin diagram and the function table of 74HC154 are presented in Table-5 and Table-6.

Table-5. Pin description of 74HC154.

Symbol	Pin	Description
$\overline{Y0}$	1	data output (active LOW)
$\overline{Y1}$	2	data output (active LOW)
$\overline{Y2}$	3	data output (active LOW)
$\overline{Y3}$	4	data output (active LOW)
$\overline{Y4}$	5	data output (active LOW)
$\overline{Y5}$	6	data output (active LOW)
$\overline{Y6}$	7	data output (active LOW)
$\overline{Y7}$	8	data output (active LOW)
$\overline{Y8}$	9	data output (active LOW)
$\overline{Y9}$	10	data output (active LOW)
$\overline{Y10}$	11	data output (active LOW)
GND	12	ground (0 V)
$\overline{Y11}$	13	data output (active LOW)
$\overline{Y12}$	14	data output (active LOW)
$\overline{Y13}$	15	data output (active LOW)
$\overline{Y14}$	16	data output (active LOW)
$\overline{Y15}$	17	data output (active LOW)
$\overline{E0}$	18	enable input (active LOW)
$\overline{E1}$	19	enable input (active LOW)
A3	20	address input
A2	21	address input
A1	22	address input
A0	23	address input
V_{CC}	24	supply voltage

Table-6. Function table of 74HC154.

Input						Output															
$\overline{E0}$	$\overline{E1}$	A0	A1	A2	A3	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$	$\overline{Y4}$	$\overline{Y5}$	$\overline{Y6}$	$\overline{Y7}$	$\overline{Y8}$	$\overline{Y9}$	$\overline{Y10}$	$\overline{Y11}$	$\overline{Y12}$	$\overline{Y13}$	$\overline{Y14}$	$\overline{Y15}$
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
		H	L	L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
		L	H	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
		H	H	L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
		L	L	H	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
		H	L	H	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
		L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
		H	H	H	L	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
		L	L	L	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
		H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
		L	H	L	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
		H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
		L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
		H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
		L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
		H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L

By using a binary code of 4bits 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 again 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.

The serial-parallel conversion is presented in Figure-6. Digital outputs 2, 3 and 4 of the Arduino UNO board are used to send 32 packets of 8bits via serial to the first register 74HC595, labeled in the figure as 74HC595-LD.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 registergoes 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 subcircuit labeled as 74HC154, is detailed in Figure-7.

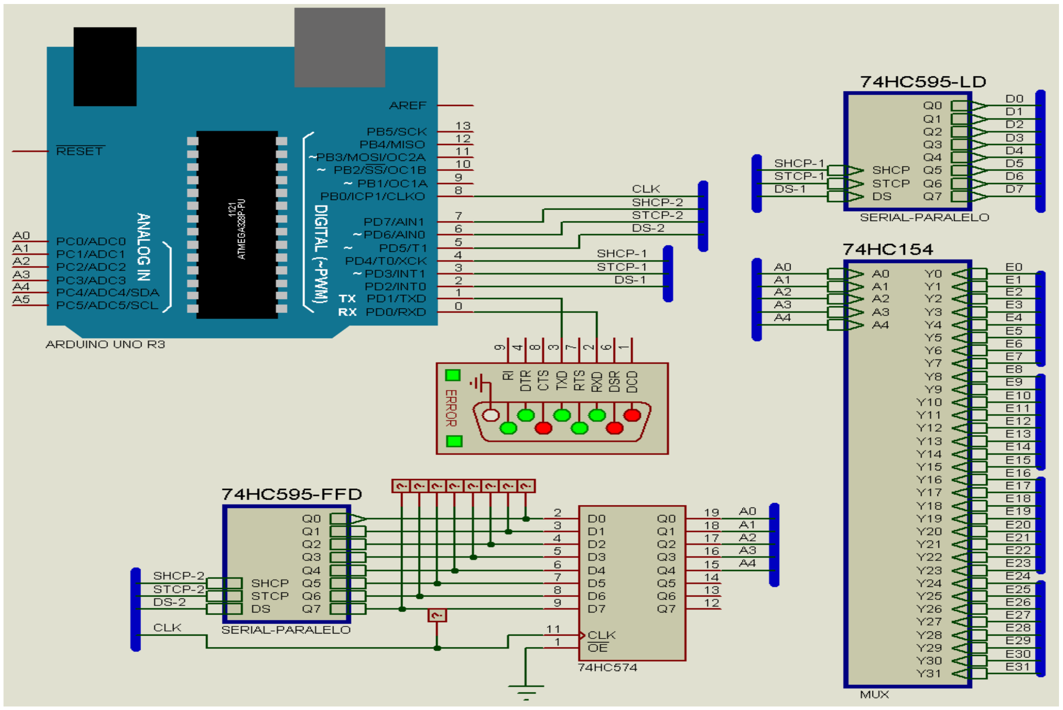


Figure-6. Serial-parallel conversion.

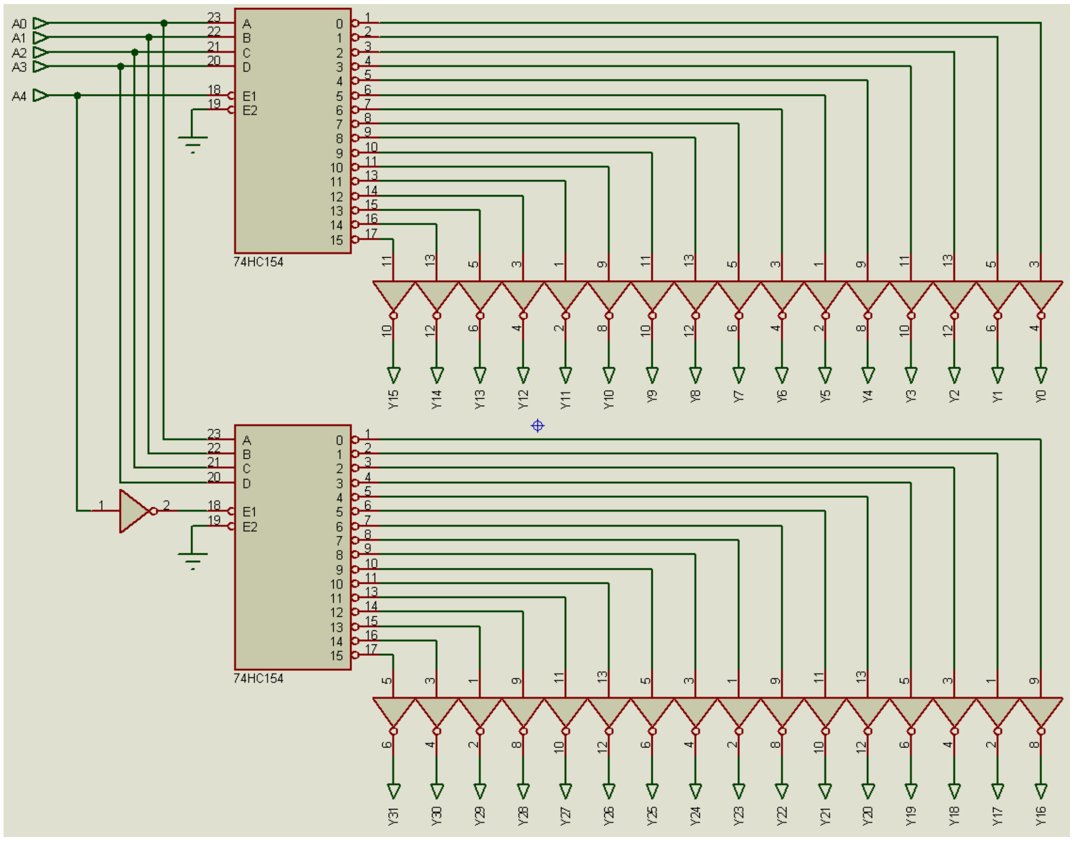


Figure-7. Decoding/demultiplexing.

The outputs of chips 74HC154 are responsible of enabling or disabling the 32 D-type latches by using LE pin. As can be seen in Figure-8, the data from the first register 74HC595 are arriving 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. In Figure-9, the subcircuit 74HC573 can be seen in detail.

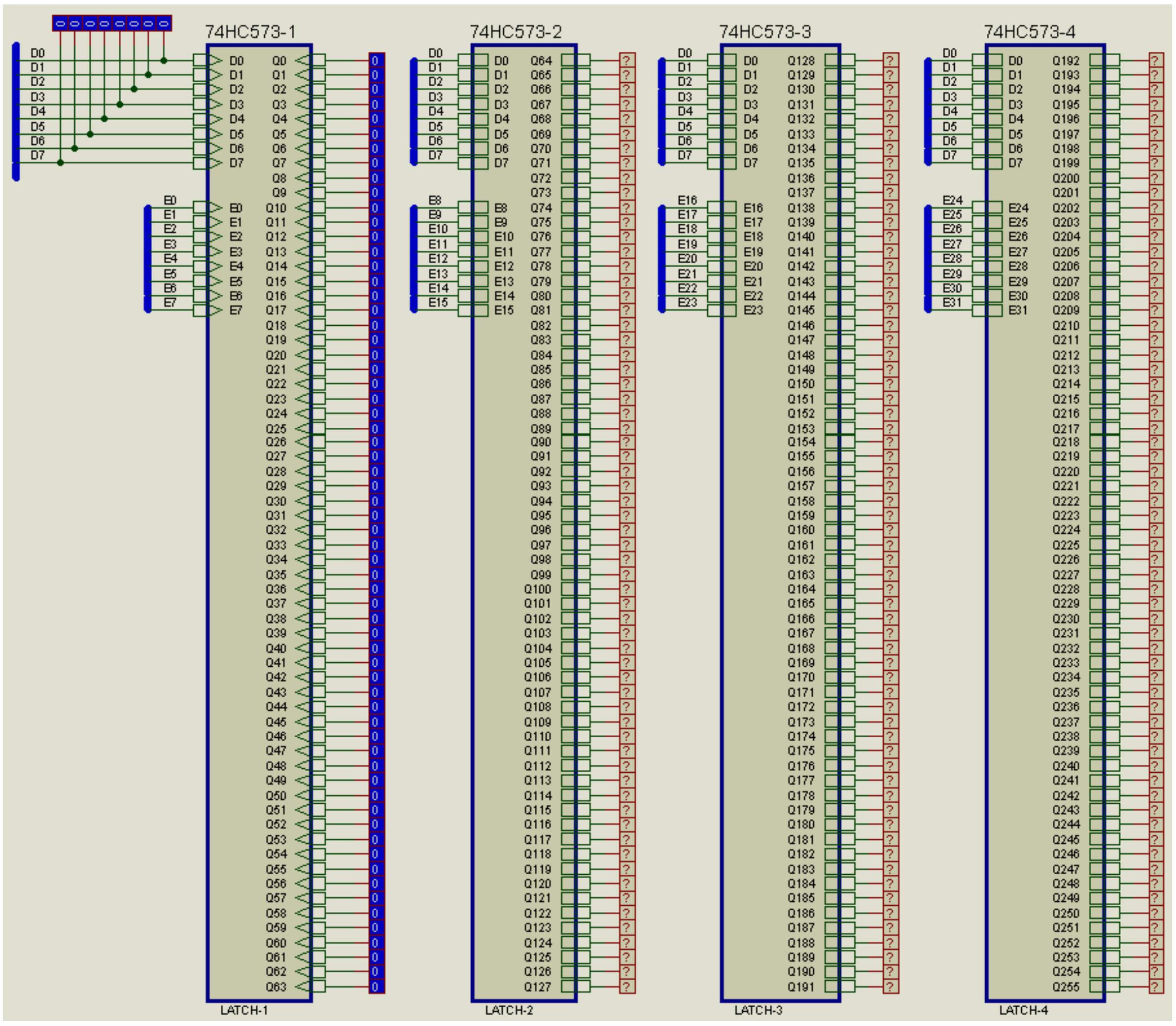


Figure-8. Subcircuits to latch information.

VOL. 11, NO. 19, OCTOBER 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2016 Asian Research Publishing Network (ARPN). All rights reserved.

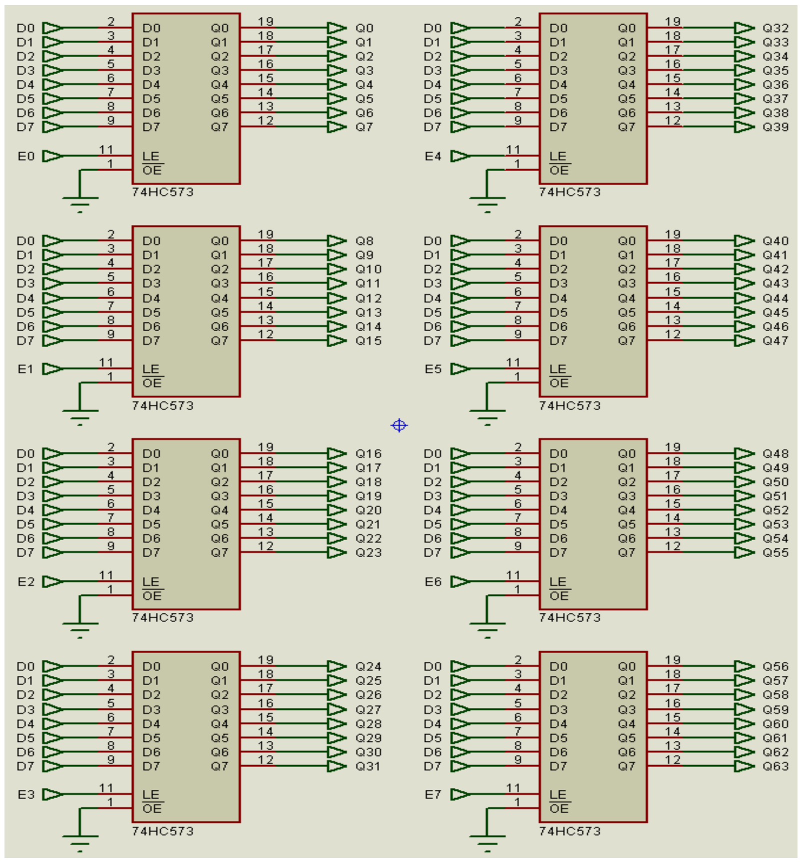


Figure-9. Subcircuit with D-type latches in detail.

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-10).

Figure-10. 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 user enters 32 packets of 8 bits by using the serial monitor. By means of serial communication and multiplexing, 256 digital outputs can be controlled only with 7pins of the Arduino UNO board - from2 to 8. The Arduino code is presented below:

```
// Digital output connected to serial data (DS) input of
// 74HC595-LD
const byte dataPin1 = 2;
// Digital output connected to latch pin (ST_CP) of
// 74HC595-LD
const byte latchPin1 = 3;
// Digital output connected to clock pin (SH_CP) of
// 74HC595-LD
const byte clockPin1 = 4;
// Digital output connected to serial data (SD) input of
// 74HC595-FFD
const byte dataPin2 = 5;
// Digital output connected to latch pin (ST_CP) of
// 74HC595-FFD
const byte latchPin2 = 6;
// Digital output connected to clock pin (SH_CP) of
// 74HC595-FFD
const byte clockPin2 = 7;
// Digital output connected to CLK pin of 74HC574 for
// synchronization
const byte clkpin = 8;
// Variable that takes the count of the data packets
byte sel = 0;

void setup() {
```

VOL. 11, NO. 19, OCTOBER 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2016 Asian Research Publishing Network (ARPN). All rights reserved.
www.arpnjournals.com

```
  Serial.begin(9600);
  for (byte i = 2; i <= 8; i++) {
    pinMode(i, OUTPUT);
  }
}

void loop() {
  while (Serial.available() > 0) {
    byte data = Serial.parseInt();
    data = constrain(data, 0, 255);
    digitalWrite(latchPin2, LOW);
    shiftOut(dataPin2, clockPin2, MSBFIRST, sel);
    digitalWrite(latchPin2, HIGH);
    // A clock pulse is generated to synchronize the process
    digitalWrite(clkpin, LOW);
    digitalWrite(clkpin, HIGH);
    digitalWrite(clkpin, LOW);
    digitalWrite(latchPin1, LOW);
    shiftOut(dataPin1, clockPin1, MSBFIRST, data);
    digitalWrite(latchPin1, HIGH);
    Serial.println(data, BIN);
  }
  sel++;
  // If 32 packets are received, the packet counter
  // restarts
  if (sel == 32) {
    sel = 0;
  }
}
```

4. RESULTS AND DISCUSSIONS

In order to verify the operation of the designed system a test is performed. The user enters three data packets of 8bits each one, by using the serial monitor. The data entered are "1", "20" and "240".

The first data entered is "1", which is "00000001" in binary. This first data is addressed to register 0 by using the decoder/demultiplexer (Figure-11).

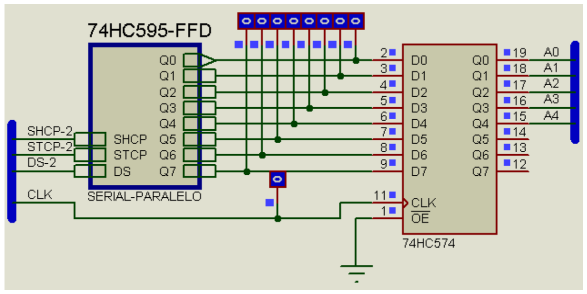
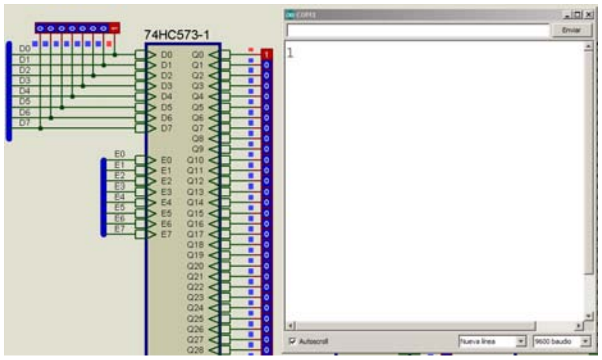


Figure-11. Addressing to register 0.

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-12, none other output suffers modification.



```
1
```

Figure-12. Test with data 00000001.

The second data introduced by using the serial monitor is "20", which is equivalent to "00010100" in binary. This data is addressed to the register 1, by using the decoder/demultiplexer (Figure-13).

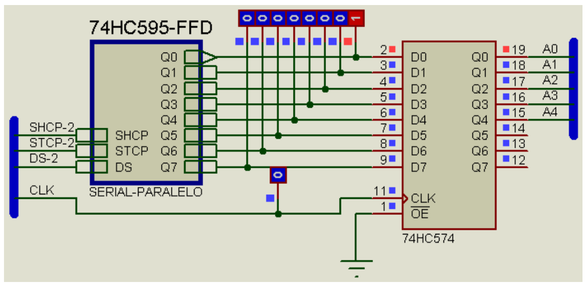
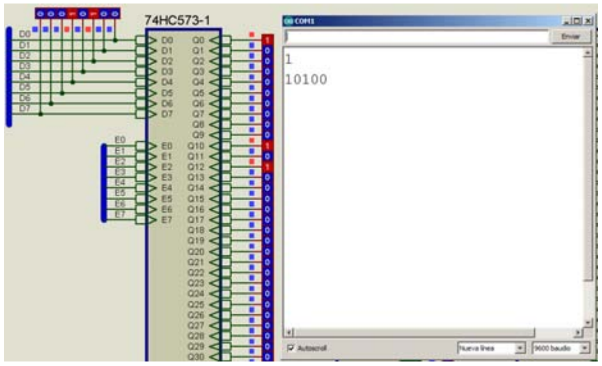


Figure-13. Addressing to register 1.

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-14, the first group of 8 bits is not affected when the transference of information is made, only the second group of outputs is modified.



```
1
10100
```

Figure-14. 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-15).

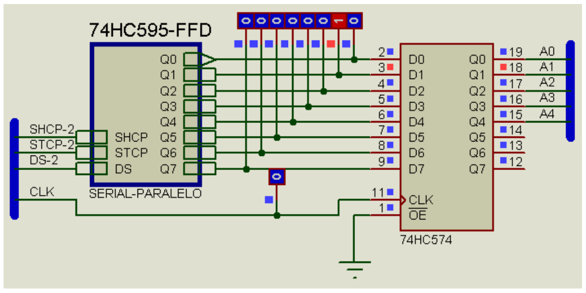
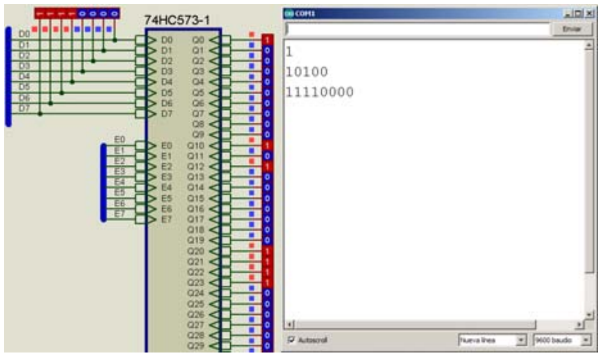


Figure-15. Addressing to register 2.

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-16, 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.



```
1
10100
11110000
```

Figure-16. Test with data 11110000.

When the user continues typing 8 bits data, these are transferred to the next group of outputs, while the previous groups of outputs do not change. By completing the transfer of the 32 packets of 8 bits, the system is enabled to receive another set of 256 bits, modifying all outputs previously attached.

In this way, the design of an economic system that allows, by using simple hardware and software, automatically to establish 256 M2CI connections, has been achieved. Future work will allow that the 256 bits are not sent by using the serial monitor, but via internet. With this approach, a remote access to training module M2CI could be done.

REFERENCES

[1] Santana I., Ferre M., Izaguirre E., Aracil R. and 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-PapersOnLine. 48(29): 49-58.

[3] Kaluz M., Garcia-Zubia J., Fikar M. and 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. and 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] E8460A 256-Channel Relay Multiplexer. (n.d.). Retrieved May 06, 2016, from http://www.keysight.com/en/pd-1000003000:epsg:pro-pn-E8460A/256-channel-high-density-reed-relay-multiplexer?cc=CO

[6] Arduino - ArduinoBoardUno. (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 - ShiftOut. (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.