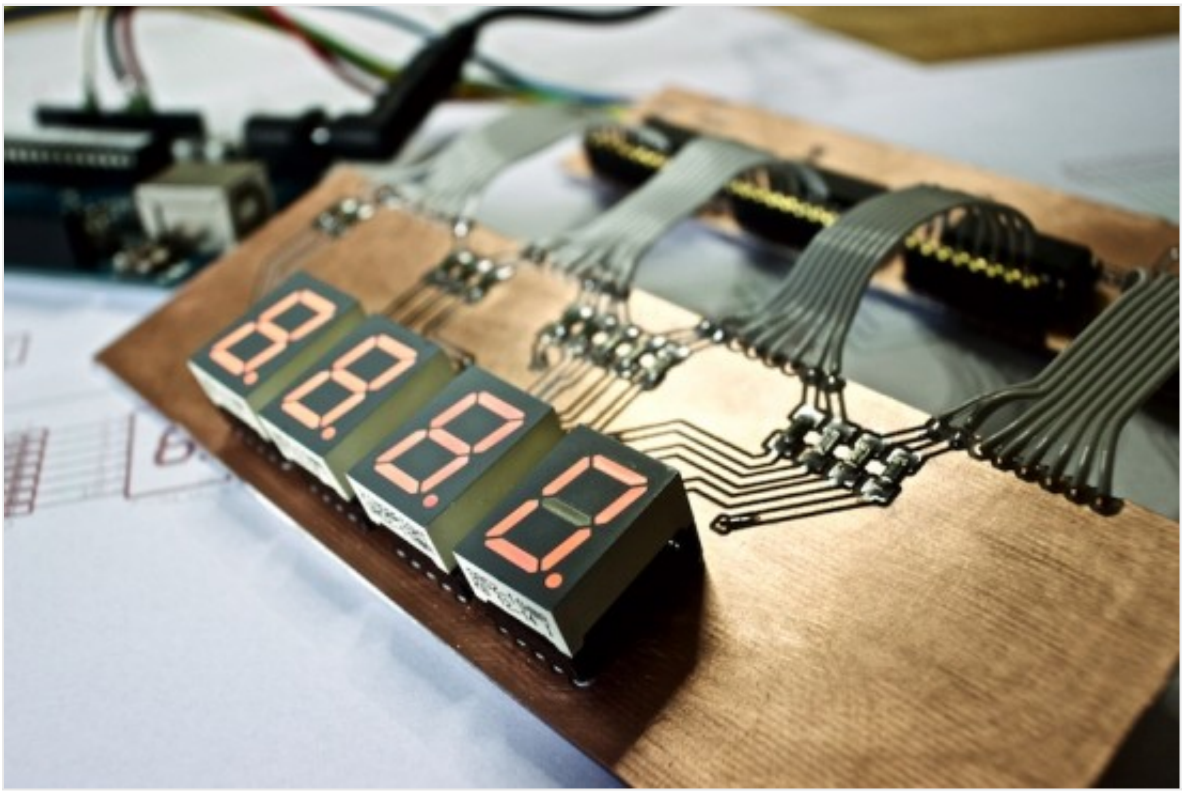
Multiple Shift-out Registers on Arduino – part 1

Shift-out: again? Yes, shift-out registers again, now daisy-chained! But let me write down a short foreword.



As probably occurs for most of the subjects, especially those related to the Arduino board, shift registers have already been discussed several times on the Internet; Blogs, Sites, Tutorials, Instructables, the same Arduino.cc site and many more. The reason why we talking again about them is that in this series of articles we are following a sort of plan: The aim is to give – all in the same place – suggestions which propose to the users projects and circuits not only compatible with Arduino so to avoid, among other things, an excessive simplification.

The other goal of these projects is providing the users with the option to see how electronic circuits can be done in a simple, even though efficient way. The same applies to the Arduino sketches related to the circuits. Last but not least, these projects can be used with other microcontroller boards with the same behavior described in the Arduino sketches.

Introducing Daisy Chain

The most clear and generic definition of this term can be the following:

A daisy chain is an interconnection of computer devices, peripherals, or network nodes in series, one after another. It is the computer equivalent of a series electrical circuit. In personal computing, examples of “daisy-chainable” interfaces include Small Computer System Interface (SCSI) and FireWire, which allow computers to communicate with peripheral hardware such as disk drives, tape drives, CD-ROM drives, printers, and scanners faster and more flexibly than previous interfaces.

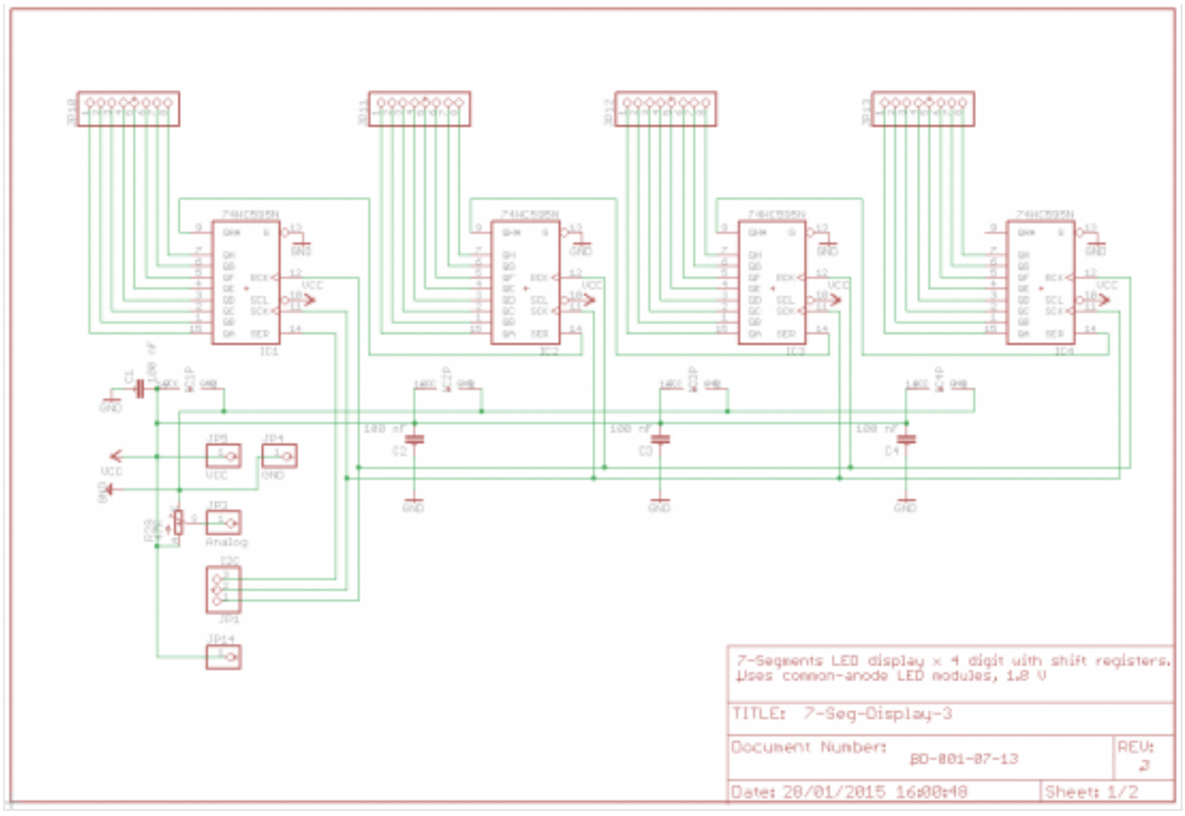
Daisy chain in electrical engineering is a wiring scheme in which multiple devices are wired together in sequence or in a ring. Daisy chains may be used for power, analog signals, digital data, or a combination thereof. (source: **en.wikipedia.org**)

As a matter of fact there are many integrated circuits, such as **74HC595** shift-out register supporting this feature: it is **daisy-chainable**. If you take a look to the circuit scheme published in the **Alphanumeric LCD with Shift Register on Arduino** you can see that there is the pin number 9 that is NC (not connected). In the shift register datasheet this pin is mentioned as **Q7 – Serial data output**.

This means that there is not only one serial input in the register but also… an exit! This is the pin we will use to daisy chain many 74HC595 together.

The mechanism is simple: the *clock* and *latch* pins are shared between all the daisy-chained devices while every Q7 PIN of the previous chip in the chain is used as serial input for the following chip. This up to the last device.

Looking at the circuit scheme below, the last chip in the chain (i.e. the fourth to the right) is the one with the Q7 pin NC.


74HC595N
IC1
IC2
IC3
IC4
GND
VCC
Analog
7-Segments LED display x 4 digit with shift registers.
Uses common-anode LED modules, 1.8 V
TITLE: 7-Seg-Display-3
Document Number: BD-001-07-13
REV: 2
Date: 28/01/2015 16:00:48
Sheet: 1/2

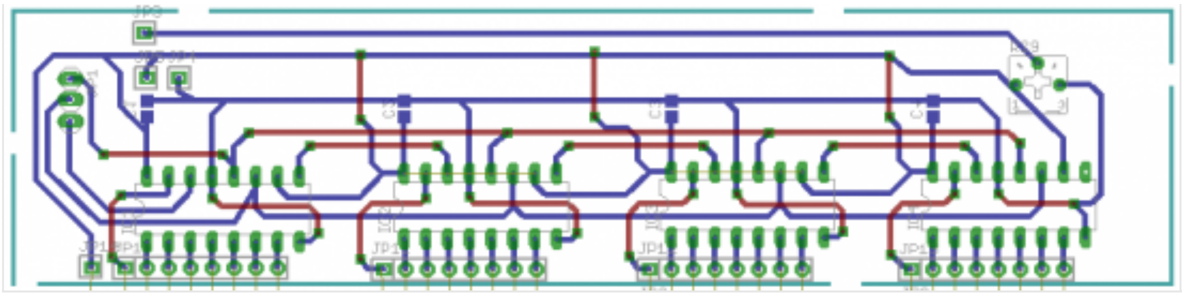
circuit schematic

By a practical point of view the other two signals clock and latch act as a bus that we will use to serially send data synchronously from the microcontroller to the series of daisy-chained shift registers.

In which order should we interpret the chained devices 8-bit sequence? The four daisy-chained shift registers operate like a single *32-bit register* where the first device in the chain (the one connected to the microcontroller) is the Most Significant Byte, so it is the last to be filled while the farthest, i.e. the fourth in our case, is the Least Significant Byte, that is, the first to be filled by the microcontroller serial data set.

The circuit in depth

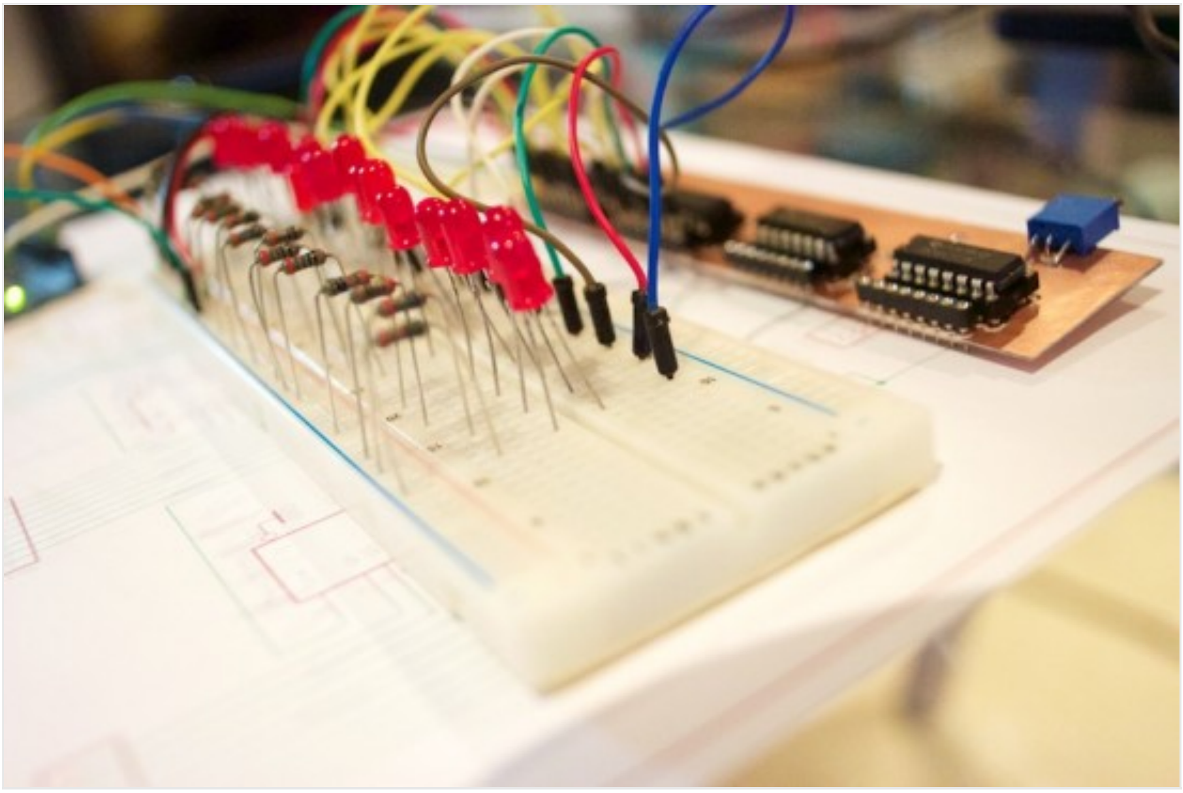
As usual, we have tried to keep the circuit as simple as possible even though stable. This circuit can be considered as a general purpose daisy-chained parallel shift-out register group for a total of 32 output ports that we can easily manage with our Arduino board (or similar microcontroller boards), using *only three I/O pin*. The four capacitors between the GND pin of every device and the GND signal are designed to give more stability to the system, as suggested in the device data sheet.



Then you will find a some how strange detail: a *variable capacitor* that can be optionally connected to an analog input of the Arduino. This part of the circuit is not directly involved with the logic of the shift registers, but as we consider to use this board as general purpose, it will be frequent for an analog value to be sent to the microcontroller at least for testing purposes.

How to control four daisy-chained shift registers

First of all, consider the following trick: by the the microcontrollers point of view, a set of daisy-chained shift registers equals to only one; the difference is that four bytes should be sent instead of only one.



The Arduino code for sending a byte to a shift-out register is not so complex, as the **shiftOut()** method is a standard part of the Arduino IDE command set. A suggested reading is the ShiftOut() function source code on the **Arduino Playground**.

So, to send eight bits to a shift register, the code is similar to the example below:

```
// Assign the Arduino pins to shift register
```

```
int latchPin = 2;   // pin  12 on the 74hc595
int dataPin = 3;   // pin 14 on the 74hc595
int clockPin = 4;  // pin 11 on the 74hc595

// Initialisation function
void setup() {
  // Set the ShiftRegister PINs as output
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  // other stuff ...
}
```

Then, in the loop function, we should send the serial 8 bits using a call to *shiftOut()*

```
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, MSBFIRST, B11111111);
   digitalWrite(latchPin, HIGH);
```

Note that before calling the shift operation the latch pin is set to low, to inform the device that the next bits should not be processed immediately. Then when the shiftOut() call returns, the latch pin signal is set to high, to store the 8 bits in the shift register. This is the same as setting the corresponding values to the 8 output pins of the shift register.

To simplify, *Alexander Brevig* (the guy who also contributed to other Arduino libraries such as the Wires library) has created the **ShiftRegister595 library** optimizing the process mentioned above, with a single call to the library instance as shown in the usage example of the library.

```
void loop() {
  //count up routine
  for (int j = 0; j < 256; j++) {
    shiftRegister595.write(i);
    delay(1000);
  }
}
```

Using the *ShiftRegister595* library, you can optimize your code, give more readability to the sketches, and simplify the calls to the output pins using one single instruction. But the library only manages one shift register, while we want to use more, daisy-chained devices. In our case four devices.

For your information: don't forget that when more than one shift register is daisy-chained, their logical behavior is a **longer register** (32 bits) able to manage several groups of 8 bits. This means that four shift registers will be considered like a 4-bytes latch where the *Most Significant Byte* is the first connected to the microcontoller data pin and the *Least Significant Byte* is the farthest in the circuit design.

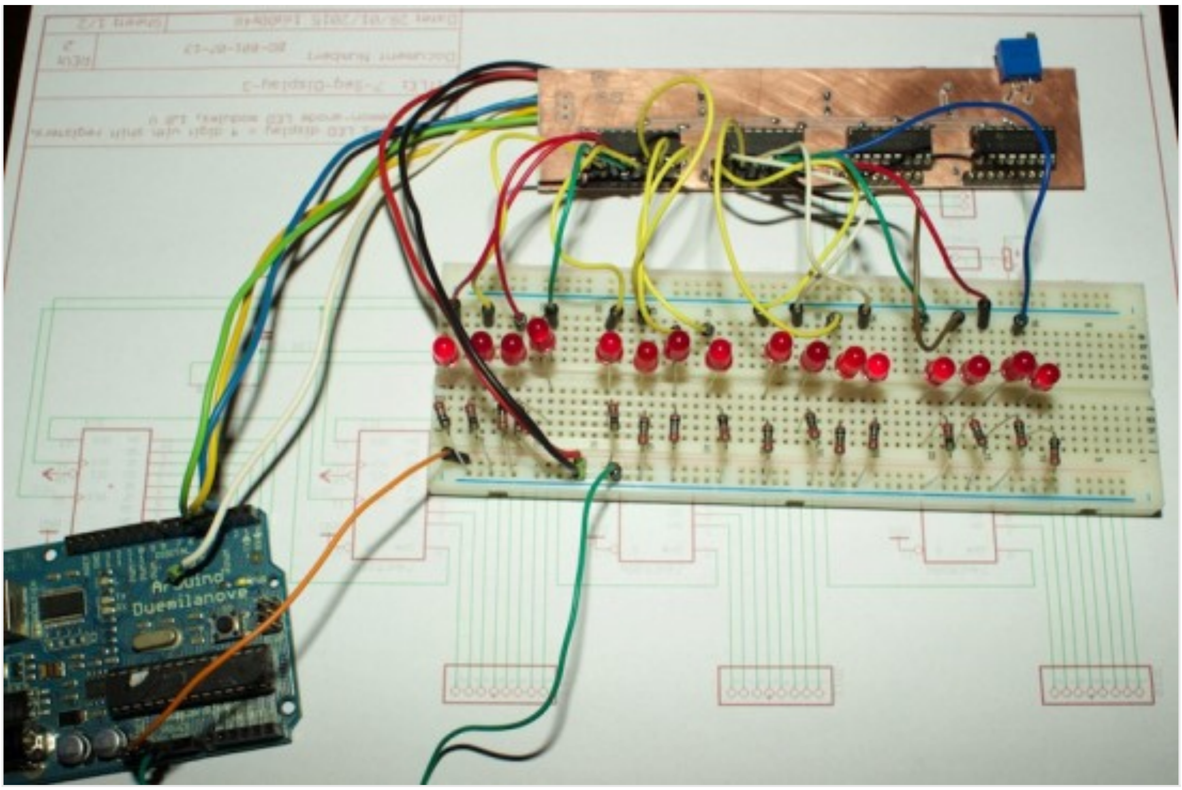
The four daisy-chained shift registers circuit can be considered as an *extension* of the standard PINs of the Arduino board (don't forget that it can be used on any other microcontroller e.g. PIC, any Arduino compatible board, and so on).

The ShiftOutX library

To better manage the circuit, we should use the **ShiftOutX library**, developed by *Juan Hernandez*. Here we need to use only the shift-out registers, so the download version of the library is 1.0 but there is also a version 1.1 of the same library supporting also special pins i.e. PWM and SPI. More information on the library and a good example (included in the library package) to be applied to our circuit can be found on the **Arduino Playground**.

Assembly testing

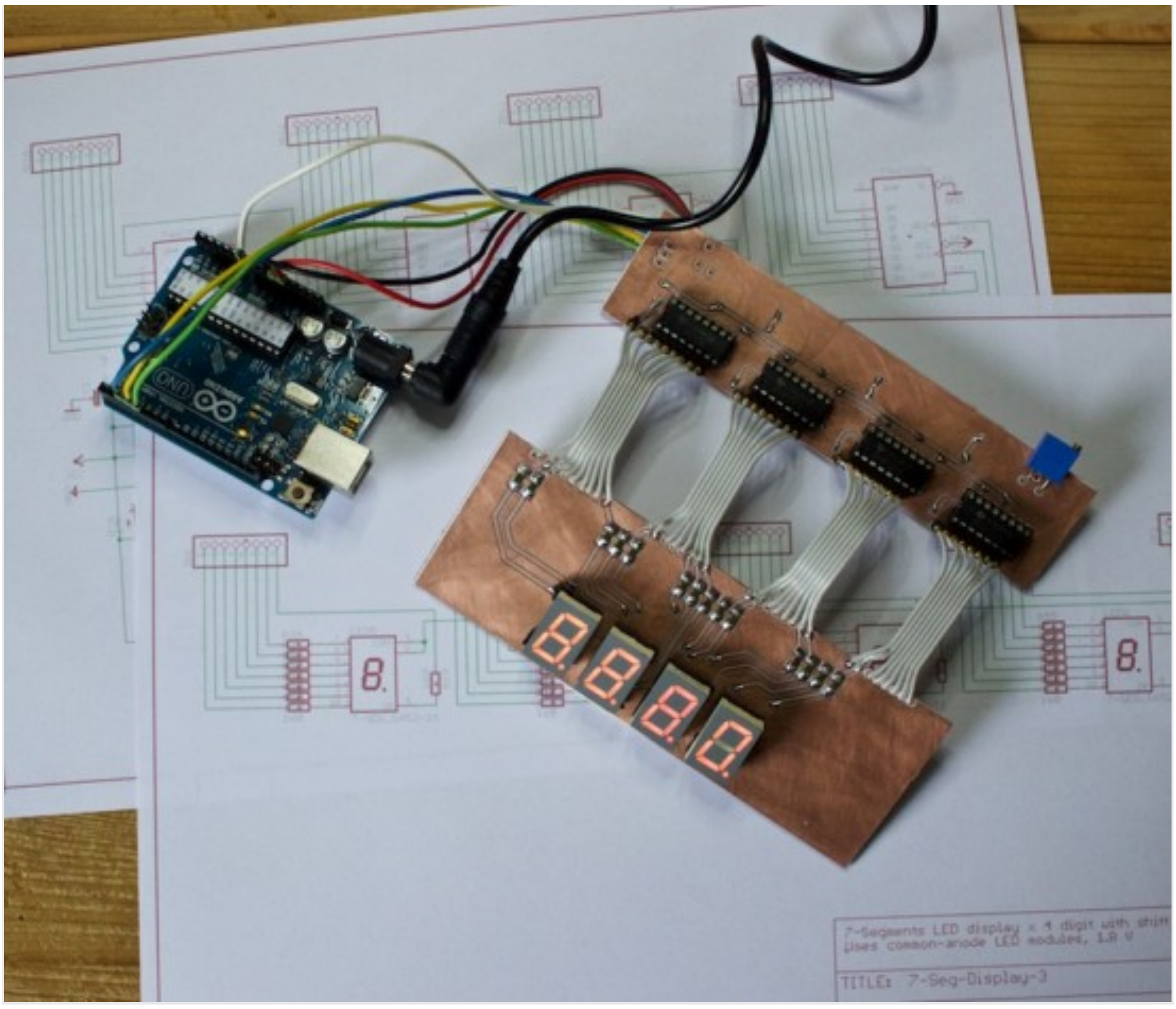
The better way to test the circuit is to use a very simple method, in order to avoid any other possible interferences. As shown in the following image, we use one led and one resistor (10 or 20 K maybe ok) connected to the ground for every output pin we want to test. This is sufficient to show us the state of every bit of the shift registers. In our example images we have used only 16 leds for simplicity, connected to the first two 8-bit groups.



Using the board to control such devices as motors or relays, the leds will have the advantage to independently monitor every signal of the register.

Example sketch

To test the shift-out registers board we can use the *ShiftOutX* library example, already provided to manage 4 daisy-chained shift-out registers. As a matter of fact the library is able support up to 8 daisy-chained 74HC595 shift registers.



A more complex application

In the next articles we will use this board in some applications requiring many pins.

The image above is a preview of the first application discussed in the next article.

You can find a pre-assembled daisy-chained four shift registers board or the PCB only at **BalearicDynamics**. Don't forget to use the discount code **ELEKTRO1008** to get 20% off discount, the special price for Electroschematics.com users.