**Introduction**

Generally, there are two ways to drive a single 7-segment display. One way is to connect its 8 pins directly to eight ports on the SunFounder Uno board, which we have done previously. Or you can connect the 74HC595 to three ports of the Uno board and then the 7-segment display to the 74HC595. In this experiment, we will use the latter. By this way, we can save five ports – considering the Uno board’s limited ports, this is very important. Now let's get started!

**Components**

- 8 \* Resistor (220Ω)

- 1 \* 74HC595

- 1 \* SunFounder Uno board

- 1 \* Breadboard

- 1 \* USB cable

- 1 \* 7-segment display

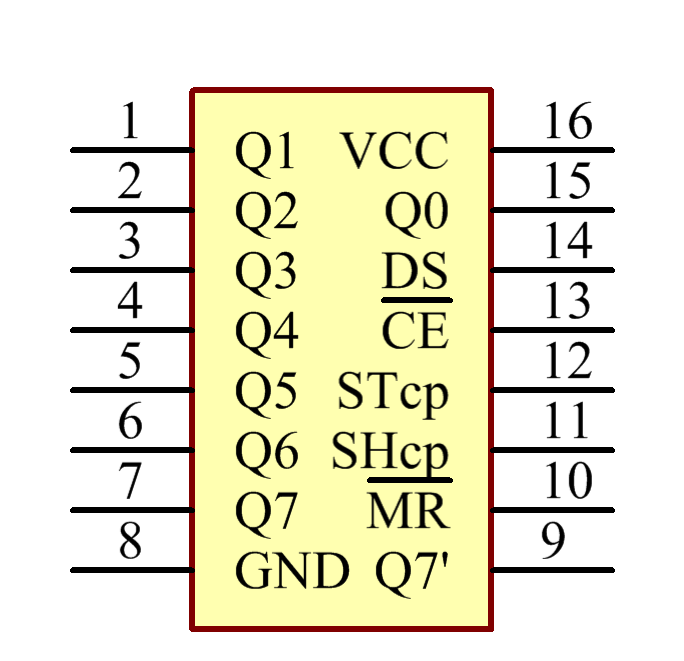
- Jumper wires

**Principle**

**74HC595**

The 74HC595 consists of an 8−bit shift register and a storage register with three−state parallel outputs. It converts serial input into parallel output so you can save IO ports of an MCU.

When MR (pin10) is high level and OE (pin13) is low level, data is input in the rising edge of SHcp and goes to the memory register through the rising edge of SHcp. If the two clocks are connected together, the shift register is always one pulse earlier than the memory register. There is a serial shift input pin (Ds), a serial output pin (Q) and an asynchronous reset button (low level) in the memory register. The memory register outputs a Bus with a parallel 8-bit and in three states. When OE is enabled (low level), the data in memory register is output to the bus.



**Pins of 74HC595 and their functions**:

**Q0-Q7**: 8-bit parallel data output pins, able to control 8 LEDs or 8 pins of 7-segment display directly.

**Q7’**: Series output pin, connected to DS of another 74HC595 to connect multiple 74HC595s in series

**MR**: Reset pin, active at low level; here it is directly connected to 5V.

**SHcp**: Time sequence input of shift register. On the rising edge, the data in shift register moves successively one bit, i.e. data in Q1 moves to Q2, and so forth. While on the falling edge, the data in shift register remain unchanged.

**STcp**: Time sequence input of storage register. On the rising edge, data in the shift register moves into memory register.

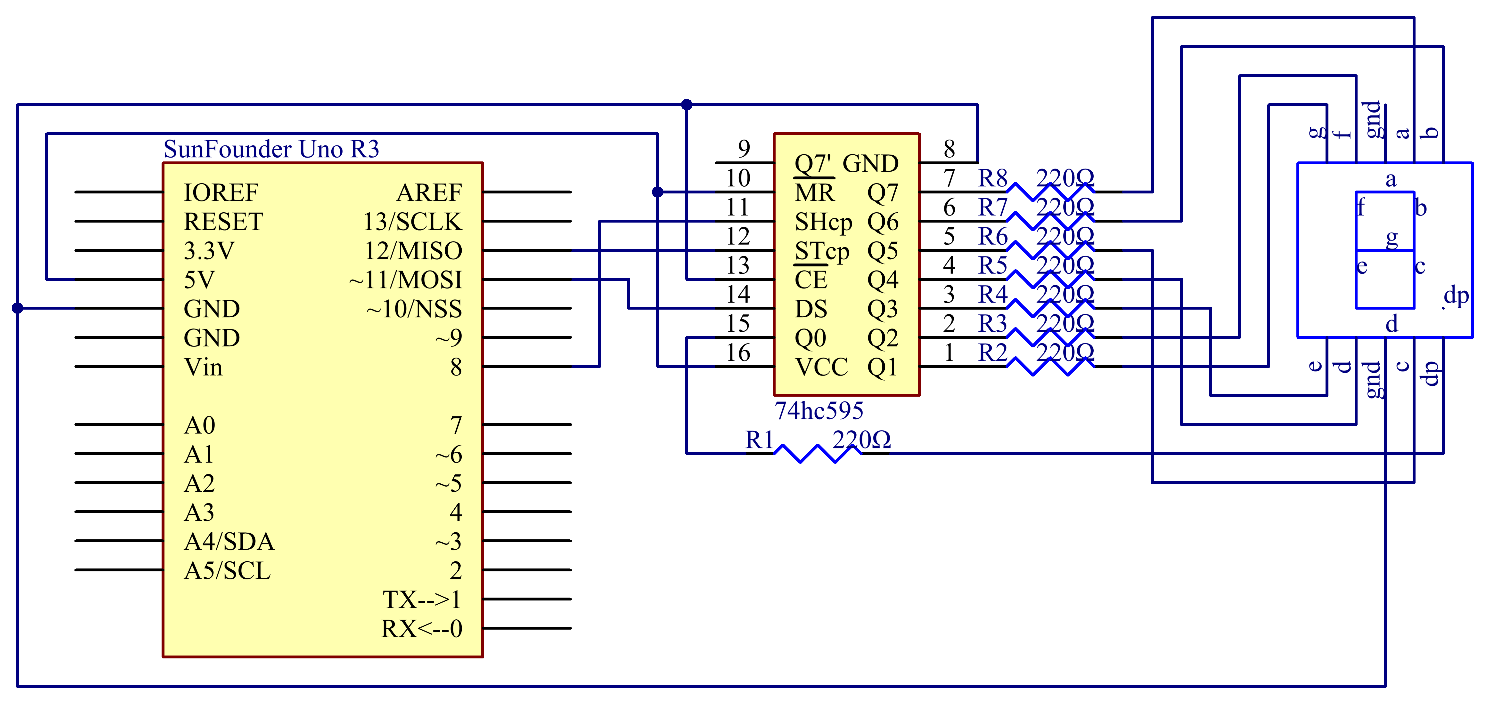
**OE**: Output enable pin, active at low level. Here connected to GND.

**DS**: Serial data input pin

**VCC**: Positive supply voltage

**GND**: Ground

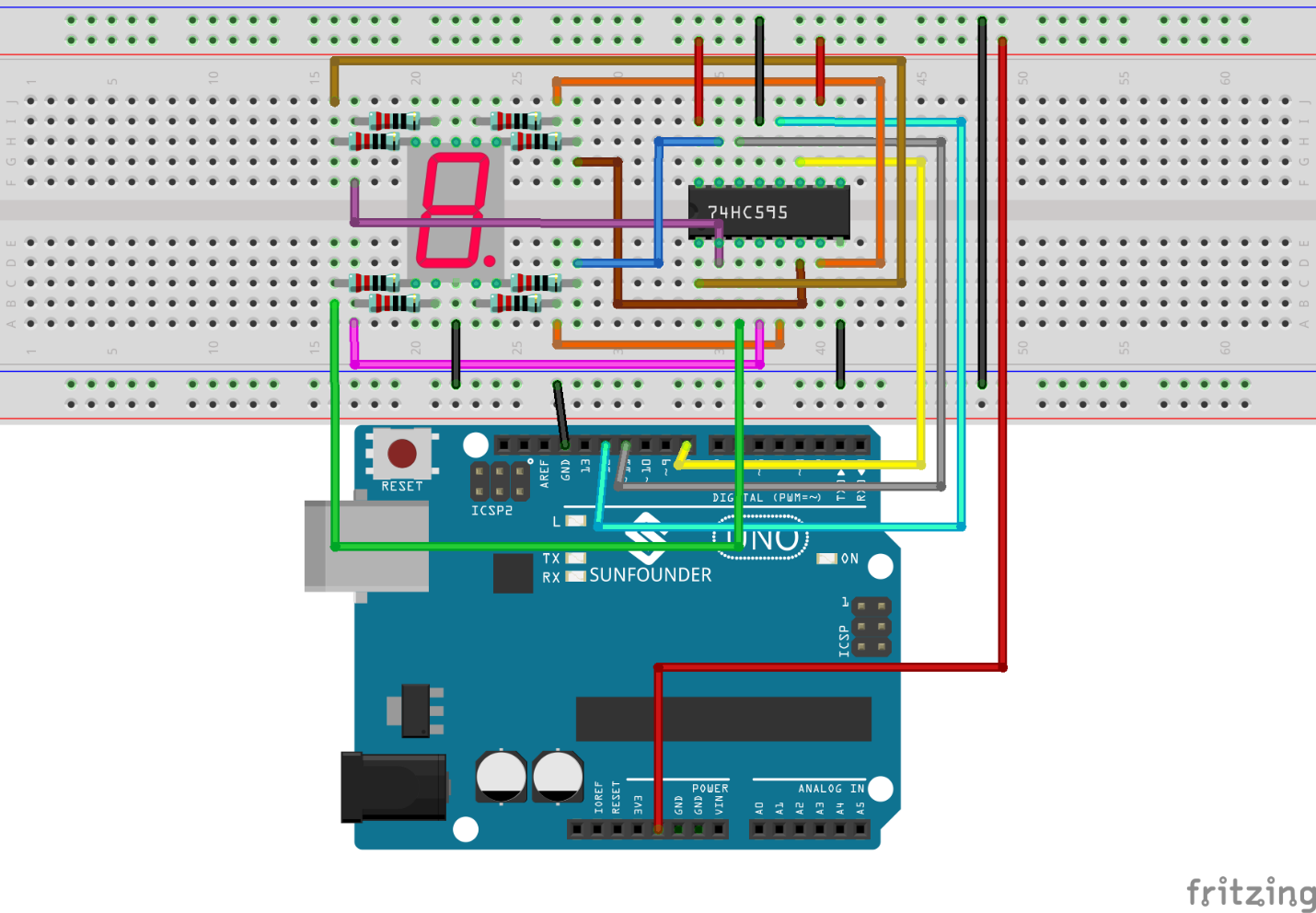
The schematic diagram



**Principle:** In the experiment MR (pin10) is connected to 5V (HIGH Level) and OE (pin 13) to GND (LOW Level). Therefore, the data is input into the rising edge of SHcp and enters the memory register through the rising edge. We use the shiftout() function to output a 8-bit data to the shift register through DS. In the rising edge of the SHcp, the data in the shift register moves successively one bit in one time, i.e. data in Q1 moves to Q2, and so forth. In the rising edge of STcp, data in the shift register moves into the memory register. All data will be moved to the memory register after 8 times. Then the data in the memory register is output to the bus (Q0-Q7). So the 16 characters are displayed in the 7-segment in turn.

**Experimental Procedures**

**Step 1:**Build the circuit (pay attention to the direction of the chip by the concave on it)

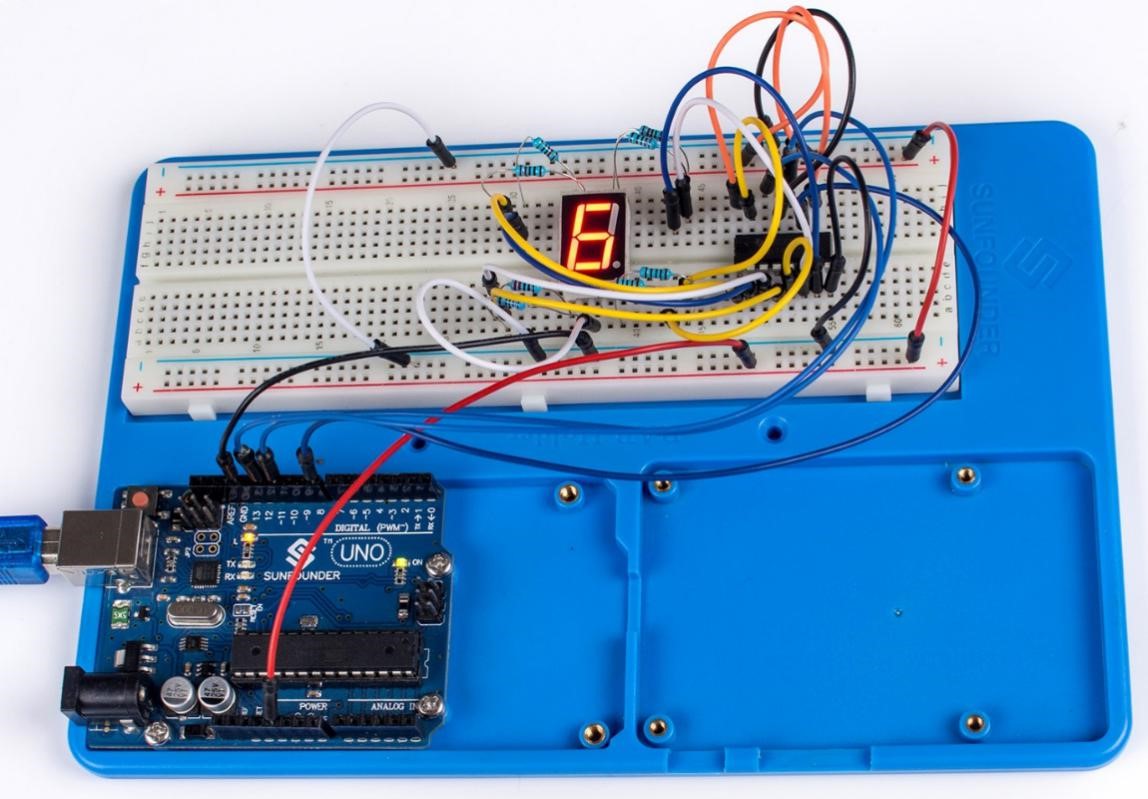


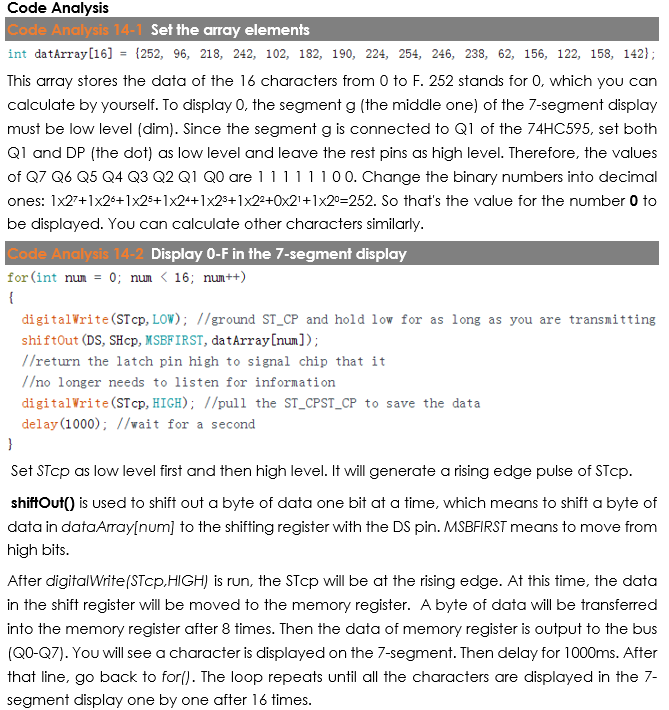
**Step 2**: Open the code file

**Step 3:** Select correct Board and Port

**Step 4:** Upload the sketch to the SunFounder Uno board

You should now see the 7-segment display from 0 to 9 and A to F.





**Code**

|  |
| --- |
| //74HC595  //You should now be able to see the 7-segment display cycle from 0 to F  //Email:support@sunfounder.com  //Website:www.sunfounder.com  //2015.5.7  const int STcp = 12;//Pin connected to ST\_CP of 74HC595  const int SHcp = 8;//Pin connected to SH\_CP of 74HC595  const int DS = 11; //Pin connected to DS of 74HC595  //display 0,1,2,3,4,5,6,7,8,9,A,b,C,d,E,F  int datArray[16] = {252, 96, 218, 242, 102, 182, 190, 224, 254, 246, 238, 62, 156, 122, 158, 142};  void setup ()  {  //set pins to output  pinMode(STcp,OUTPUT);  pinMode(SHcp,OUTPUT);  pinMode(DS,OUTPUT);  }  void loop()  {  for(int num = 0; num < 16; num++)  {  digitalWrite(STcp,LOW); //ground ST\_CP and hold low for as long as you are transmitting  shiftOut(DS,SHcp,MSBFIRST,datArray[num]);  //return the latch pin high to signal chip that it  //no longer needs to listen for information  digitalWrite(STcp,HIGH); //pull the ST\_CPST\_CP to save the data  delay(1000); //wait for a second  }  } |