Lesson 6 Eight LED with 74HC595

Introduction

In this lesson, you will learn how to use the IC 74HC595 to turn on the LEDs in the order you specify.

Hardware Required

- ✓ 1 * RexQualis UNO R3
- ✓ 1 * Breadboard
- ✓ 8 * LED
- ✓ 8 * 220ohm Resistors
- ✓ 1 * 74HC595 IC
- ✓ 14 * M-M Jumper Wires

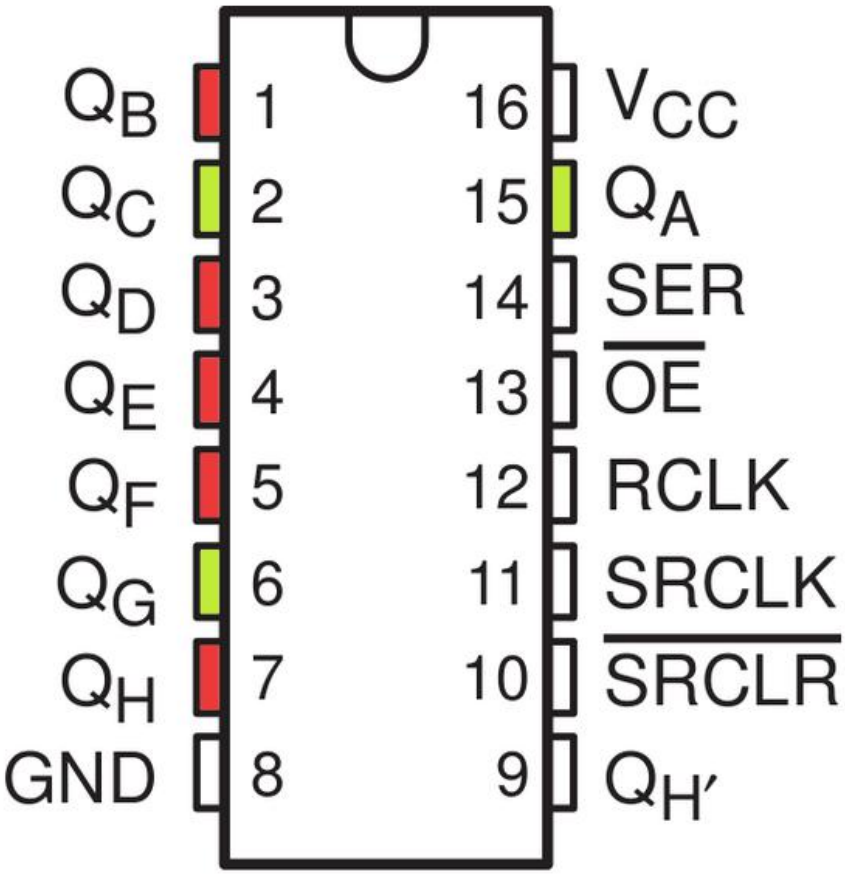
Principle

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 that you can save IO ports of an MCU. The 74HC595 is widely used to indicate multipath LEDs and drive multi-bit segment displays. "Three-state" refers to the fact that you can set the output pins as either high, low or "high impedance." With data latching, the instant output will not be affected during the shifting; with the data output, you can cascade 74HC595s more easily.

The chip contains eight pins that we can use for output, each of which is associated with a bit in the register. In the case of the 74HC595 IC, we refer to

these as QA through to QH. In order to write to these outputs via the Arduino, we have to send a binary value to the shift register, and from that number ,the shift register can figure out which outputs to use. For example, if we sent the binary value 10100010, the pins highlighted in green in the image below would be active and the ones highlighted in red would be inactive.



Code interpretation

int latchPin = 4;**//Pin 12 (RCLK) of the shift register to pin 4 on the Arduino – this will be referred to as the "latch pin"**

int clockPin = 5;**//Pin 11 (SRCLK) of the shift register to pin 5 on the Arduino – this will be referred to as the "clock pin"**

int dataPin = 2;**//Pin 14 (SER) of the shift register to pin 2 on the**

Arduino – this will be referred to as the "data pin"

```
byte leds = 0;//a variable called 'leds' is defined. This will be used to hold the pattern of which LEDs are currently turned on or off. Data of type 'byte' represents numbers using eight bits. Each bit can be either on or off, so this is perfect for keeping track of which of our eight LEDs are on or off.

int currentLED = 0;

void setup()

{

  //The 'setup' function just sets the three pins we are using to be digital outputs.

    pinMode(latchPin, OUTPUT);

    pinMode(dataPin, OUTPUT);

    pinMode(clockPin, OUTPUT);

    leds = 0;

}

void loop()

{

  //In the loop method, we clear the bits in the leds variable at the start of every iteration so that all the bits are set to 0 as we only want to light up one LED at a time. After this we increment or reset the currentLED variable so that we are lighting up the correct LED next.

    leds = 0;
```

```
    if (currentLED == 7)
    {
        currentLED = 0;
    }
    else
    {
        currentLED++;
    }
    bitSet(leds, currentLED);
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, leds);
    digitalWrite(latchPin, HIGH);
    delay(250);
}
```

Experimental Procedures

Step 1: Build the circuit

To start with let's connect pins 16 (VCC) and 10 (SRCLR) to the 5v pin on the Arduino and connect pins 8 (GND) and 13 (OE) to the Gnd pin on the Arduino. Pin 13 (OE) is used to enable the outputs, as this is an active low pin we can just connect this directly to ground.

Next we need to connect the three pins that we will control the shift register

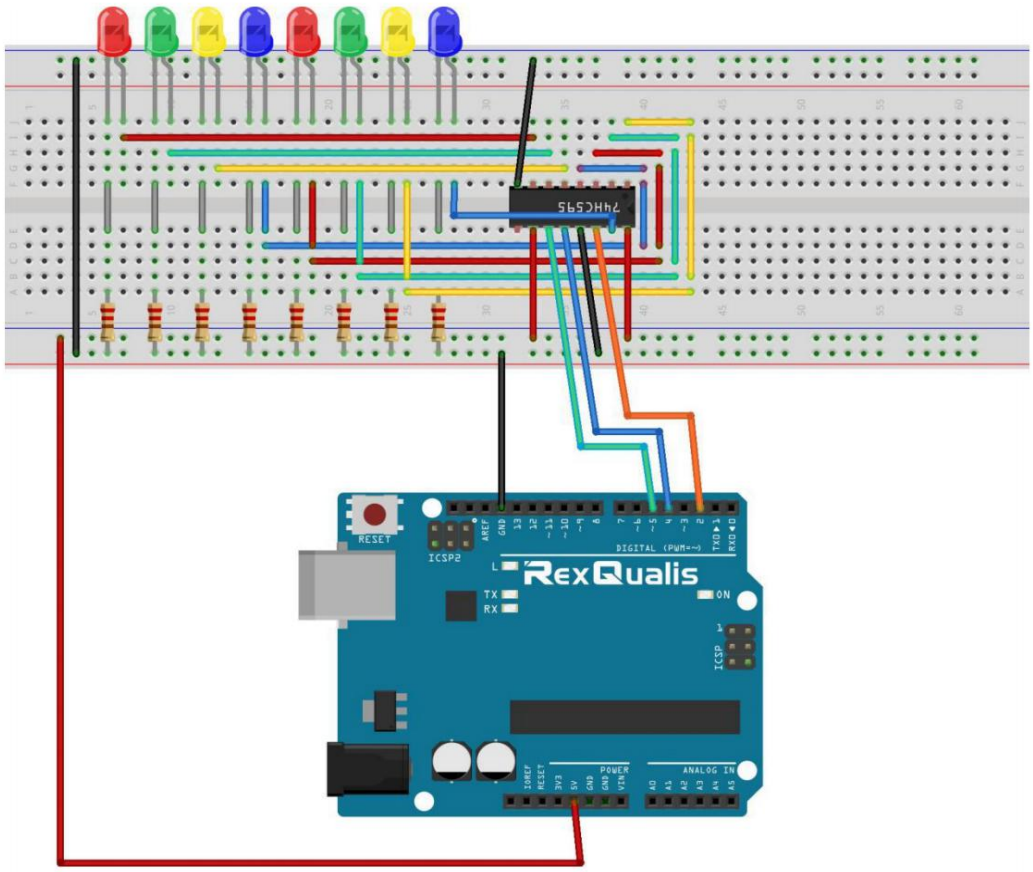
with:

Pin 11 (SRCLK) of the shift register to pin 5 on the Arduino – this will be referred to as the “clock pin”

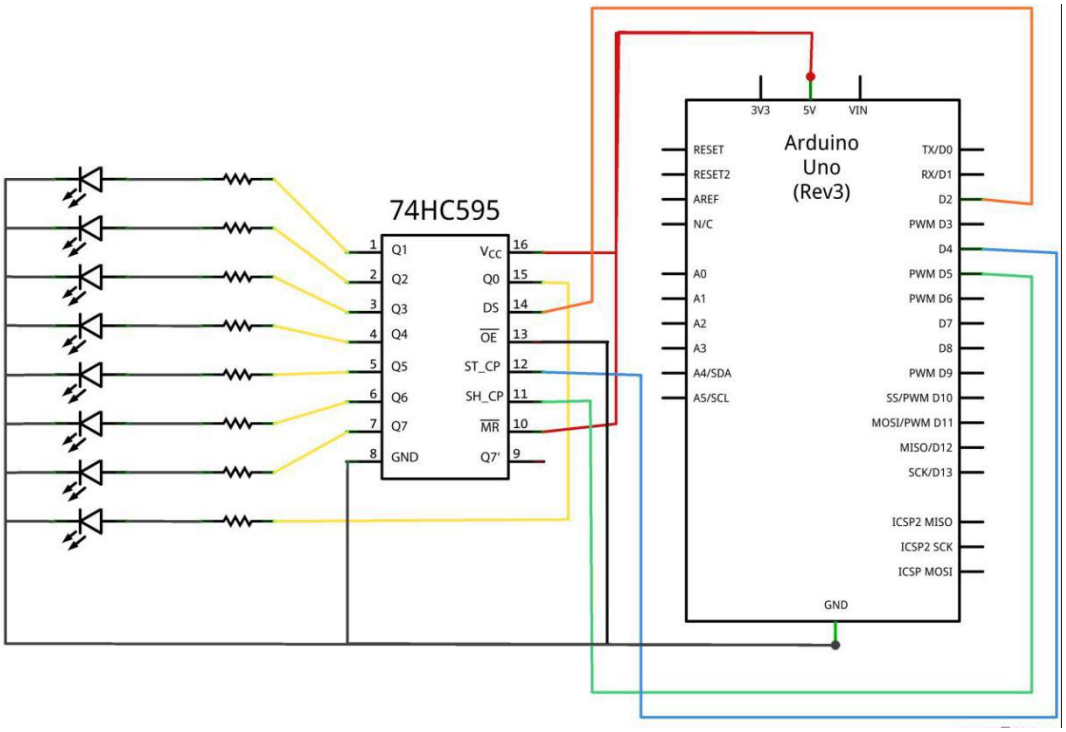
Pin 12 (RCLK) of the shift register to pin 4 on the Arduino – this will be referred to as the “latch pin”

Pin 14 (SER) of the shift register to pin 2 on the Arduino – this will be referred to as the “data pin”

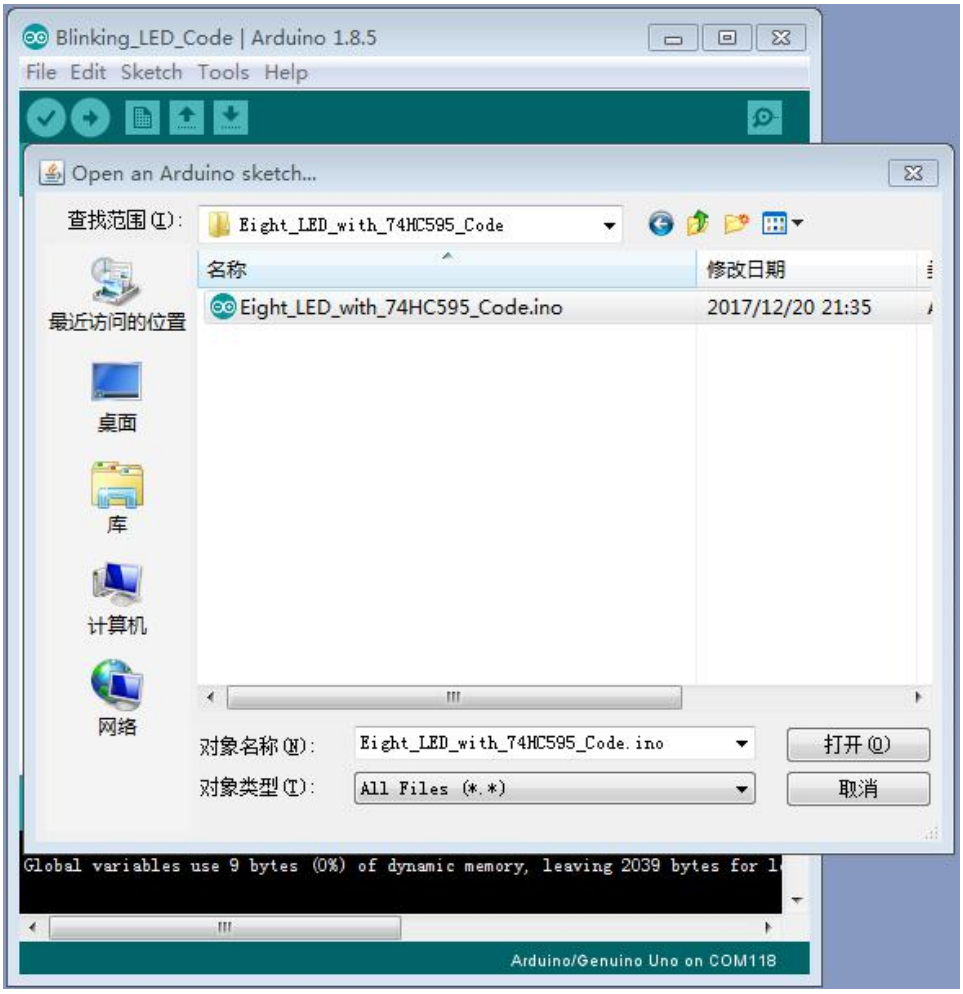
When placing the LEDs be sure that they are connected in order, so that QA is wired to the first LED, and QH is wired to the last LED, as otherwise our code is not going to light up the LEDs in the correct order!



Schematic Diagram



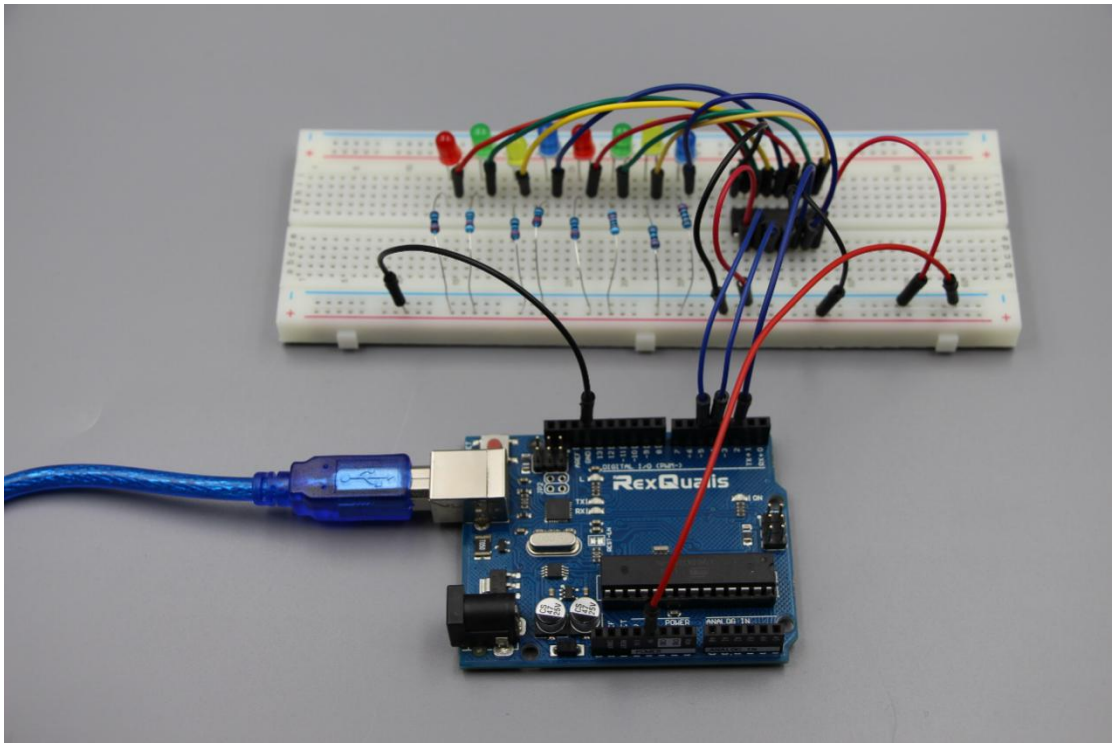
Step 2: Open the code:Eight_LED_with_74HC595_Code



Step 3: Attach Arduino UNO R3 board to your computer via USB cable and check that the 'Board Type' and 'Serial Port' are set correctly.

Step 4: Upload the code to the RexQualis UNO R3 board.

If you’ve completed all these steps correctly you should have something similar to that in the video



If it isn’t working, make sure you have assembled the circuit correctly, verified and uploaded the code to your board. For how to upload the code and install the library, check Lesson 0 Preface.