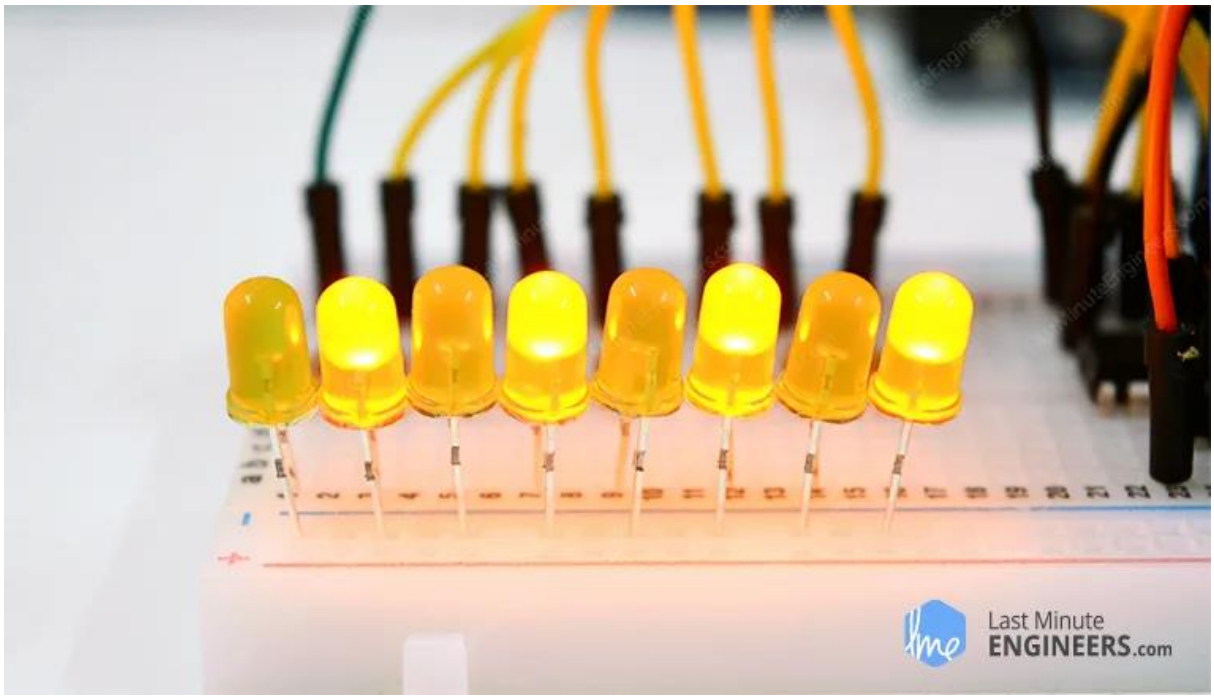
74HC595



One of the wonderful things about the Arduino is that it comes with a good number of I/O pins for your projects. You can connect various components like buttons, sensors, and motors to these pins. However, as your project grows more complex, you'll quickly find yourself running out of pins.

Imagine you're creating a project that needs to control 16 individual LEDs. Normally, you would need 16 separate pins on your Arduino to do this. The problem is, most Arduino boards don't have that many available pins.

Fortunately, there's a clever solution to this problem! You can use something called a shift register, which lets you add more I/O pins to your Arduino (or any microcontroller). The most popular and widely used shift register is the 74HC595, often simply referred to as the "595".

What makes the 74HC595 so useful is that it can control eight different output pins while only requiring three input pins from your Arduino. Even more impressive is that if you need more than 8 additional pins, you can connect multiple shift registers together in what's called a "daisy chain". This allows you to create as many I/O pins as you need.

For our example with 16 LEDs, you could connect two shift registers in series and control all 16 LEDs using just three pins from your Arduino. That's a tremendous saving of resources!

But how does the 74HC595 manage to do this? It uses a technique called "bit-shifting". Let's learn more about how this works!

HOW DOES THE 74HC595 SHIFT REGISTER WORK?

Internally, the 74HC595 chip contains two 8-bit registers:

The shift register – This register's job is to receive data input one bit at a time (serially).

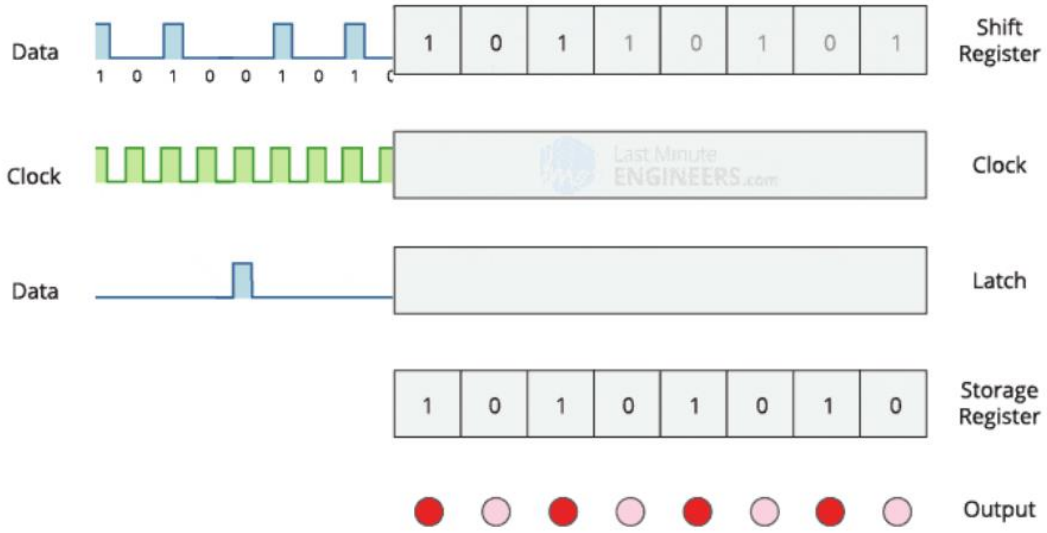
The storage register – This register holds the data and then sends it all at once (in parallel) to the output pins.

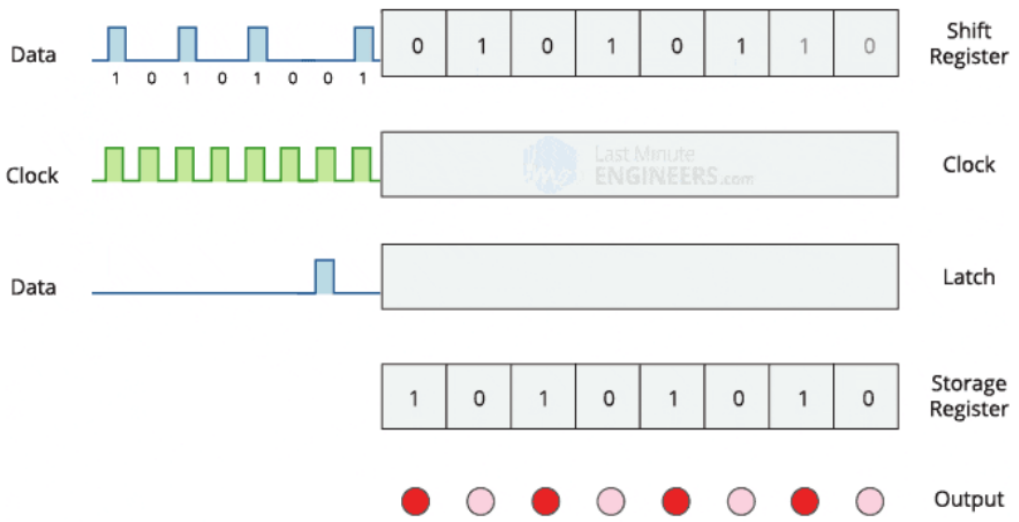
When you want to send information to the chip, you send the data to the SER pin (Serial Data Input), one bit at a time, while simultaneously pulsing the SRCLK pin (Shift Register Clock).

Each time you pulse the SRCLK pin, all the bits inside move forward by one position, and allow a new bit to enter. After 8 pulses, you'll have loaded all 8 bits into the shift register.

But here's the important part – even though the data is now inside the shift register, it doesn't immediately show up on the output pins. This is because the shift register and the storage register are kept separate on purpose. This separation allows you to load new data without changing the outputs right away.

When you're ready to update the outputs, you pulse the RCLK (Register Clock or Latch Pin). This copies all the data from the shift register into the storage register, which then updates all the output pins (labeled QA through QH) at the same time. This simultaneous updating ensures that the outputs stay stable while you're loading new data.





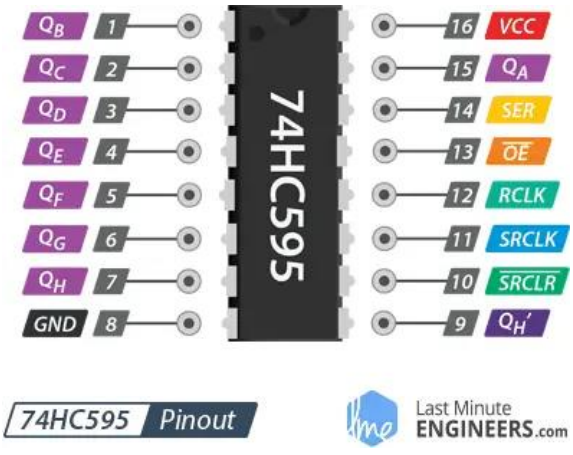
The chip also has an OE (Output Enable) pin that acts like an on/off switch for all outputs. This pin is "active-low," which means when you connect it to ground (set it "low"), the outputs will show whatever is in the storage register. If you set this pin "high", all outputs shut off, regardless of what's in the registers.

Finally, there's a SRCLR (Shift Register Clear) pin that lets you reset the shift register. When activated, it changes all stored bits to 0, effectively erasing all the data.

74HC595 SHIFT REGISTER PINOUT

Before we dive into the details, it's important to know that the 74HC595 chip is made by several different companies. In this guide, we'll focus on the SN74HC595N from Texas Instruments, which is one of the most common versions. If you have a different version, you should check its datasheet to see if there are any differences.

Now, let's look at what each pin on this chip does:



GND is the ground pin and should be connected to the ground of your circuit.

VCC provides power to the 74HC595 and should be connected to 5V.

SER (Serial Input) is where you send your data (bits) into the shift register, one bit at a time. The data starts with the least significant bit (LSB) and moves in with each clock pulse.

SRCLK (Shift Register Clock) controls how data moves into the shift register. It is positive-edge triggered, which means the data shifts when the clock signal changes from LOW to HIGH.

RCLK (Register Clock / Latch Pin) controls when the data from the shift register is transferred to the storage register. When the signal changes from LOW to HIGH on this pin, the shift register data is copied to the storage register and appears on the output pins (QA–QH).

SRCLR (Shift Register Clear) is used to reset the shift register by clearing all bits to 0. This pin is Active-Low, which means you need to connect it to ground (set it LOW) to perform the reset. For normal operation, it should be kept HIGH (connected to power).

OE (Output Enable) acts like an on/off switch for all outputs. This pin is also Active-Low, which means when you connect it to ground (set it LOW), the outputs (QA–QH) will show whatever is in the storage register. If you set this pin HIGH, all outputs shut off, regardless of what's stored in the registers.

QA–QH (Parallel Outputs) are the eight output pins that you can connect to LEDs, motors, or other devices.

QH' (Serial Output) outputs the most significant bit (MSB) of the shift register. It's mainly used for connecting multiple 74HC595 chips together. If you connect QH' of one chip to the SER pin of another and share the same clock signals, the two chips will work together as a single unit with 16 outputs. You can keep connecting more chips this way to get even more outputs!

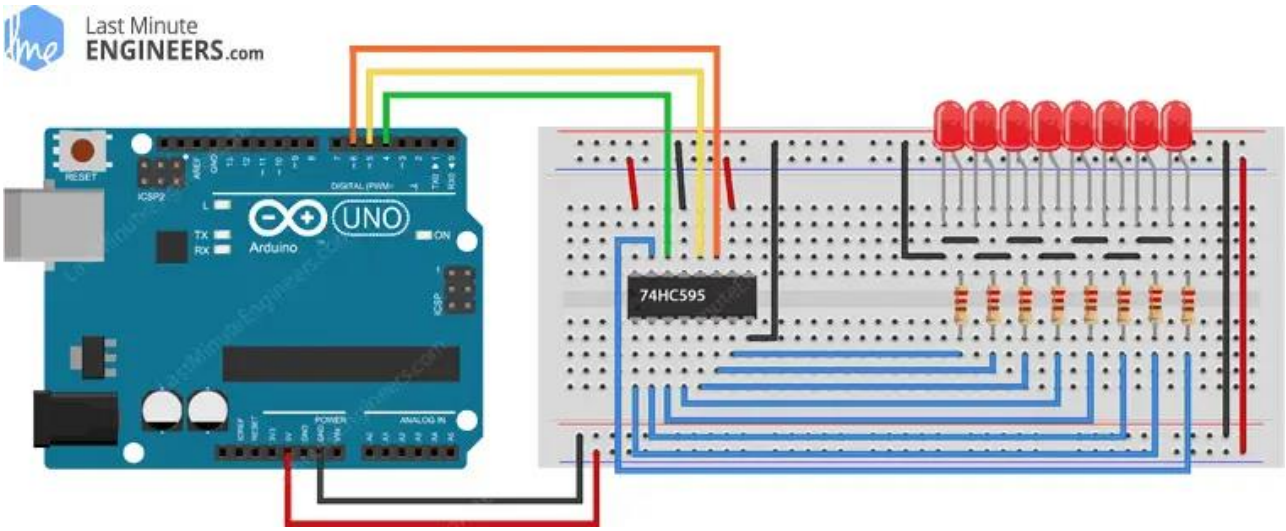
WIRING A 74HC595 SHIFT REGISTER TO AN ARDUINO

Now that we understand how the 74HC595 works, let's connect it to an Arduino and start using it!

1. Placing the Shift Register on the Breadboard
2. Power and Ground Connections
3. Connecting the Control Pins
4. Connecting the LEDs

Quick Reference

The following diagram will help you wire everything correctly:



CODE

Here's a simple program that gradually turns on each LED one by one until all are lit. Once all LEDs are on, they will turn off together, and the cycle will repeat.

First, upload the code to your Arduino and observe how the LEDs light up. Then, we'll break down how it works.

2nd_Arduino_Final | Arduino IDE 2.3.6

File Edit Sketch Tools Help

Arduino Uno

2nd_Arduino_Final.ino

```
1  int latchPin = 5;  // Latch pin of 74HC595 is connected to Digital pin 5
2  int clockPin = 6;  // Clock pin of 74HC595 is connected to Digital pin 6
3  int dataPin = 4;   // Data pin of 74HC595 is connected to Digital pin 4
4
5  byte leds = 0;  // Variable to hold the pattern of which LEDs are currently turned on or off
6
7  void setup() {
8    // Set all the pins of 74HC595 as OUTPUT
9    pinMode(latchPin, OUTPUT);
10   pinMode(dataPin, OUTPUT);
11   pinMode(clockPin, OUTPUT);
12 }
13
14 void loop() {
15   leds = 0;  // Initially turns all the LEDs off, by giving the variable 'leds' the value 0
16   updateShiftRegister();
17   delay(500);
18   for (int i = 0; i < 8; i++)  // Turn all the LEDs ON one by one.
19   {
20     bitSet(leds, i);  // Set the bit that controls that LED in the variable 'leds'
21     updateShiftRegister();
22     delay(500);
23   }
```

```
24  }
25
26  /*
27   This function sets the latchPin to low, then calls the Arduino function 'shiftOut' to shift
28   out contents of variable 'leds' in the shift register before putting the 'latchPin' high again.
29   */
30  void updateShiftRegister() {
31    digitalWrite(latchPin, LOW);
32    shiftOut(dataPin, clockPin, LSBFIRST, leds);
33    digitalWrite(latchPin, HIGH);
34  }
```

REFERENCE:

https://lastminuteengineers.com/74hc595-shift-register-arduino-tutorial/