Registro desplazamiento 74HC595

Ya sabemos que Arduino UNO dispone de 14 pines digitales que podríamos usar para entrada y salida y que además disponemos de 6 puertas de entrada analógicas de A0 hasta A5. Lo que no suele ser muy conocido es que las puertas analógicas también se pueden usar como pines digitales en caso necesario. Podemos leer y escribir valores digitales exactamente como lo hacemos con los pines digitales normales:

pinMode (A1, OUTPUT) ;

Y después escribir tranquilamente:

digitalWrite(A1, HIGH) ;

Nuestras puertas analógicas se comportarán gentilmente como puertas digitales (*Si, también podemos hacer lo mismo para la lectura*).

Así pues, en realidad, disponemos de 20 puertas digitales para nuestras cosas. Pero en la vida de toda persona, siempre hay un momento en que esto no es suficiente.

Podemos pasarnos a un **Arduino Mega** que con 54 pines digitales más 16 puertas analógicas hacen el impresionante número de 60 puertas disponibles. Pero al final, la combinación entre la ley de Murphy y la segunda ley de la termodinámica (La de que todo tiende al caos) garantizan que aparezca alguien que quiere un cuadro de luces con 64 LEDs (o 128 ya puestos) y la catástrofe nos acecha porque no hay **Arduinos** con más pines.

Afortunadamente la industria electrónica nos provee con una forma sencilla de aumentar el número de salidas digitales de nuestros Arduinos sin demasiada complicación. Unos pequeños chips llamados **Shift Registers** fáciles de encontrar como el **74HC595**.

El **74HC595** es un **Shift Register** de 8 bits serial-in, parallel-out, pertenece a una familia de chips que aceptan una entrada de bits en serie y los sacan en 8 pines paralelos. Solo sirve para escribir señales digitales y no para leerlas.

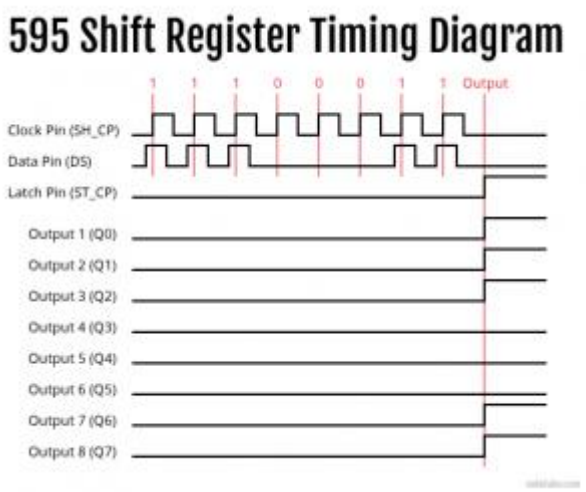
- *Si lo que necesitas es aumentar los pines digitales de entrada, prueba a usar 74HC165 que es un shift register de entrada*
- *Si lo que necesitas es aumentar los pines analógicos puedes usar un multiplexor / demultiplexor como el 74HC4051.*

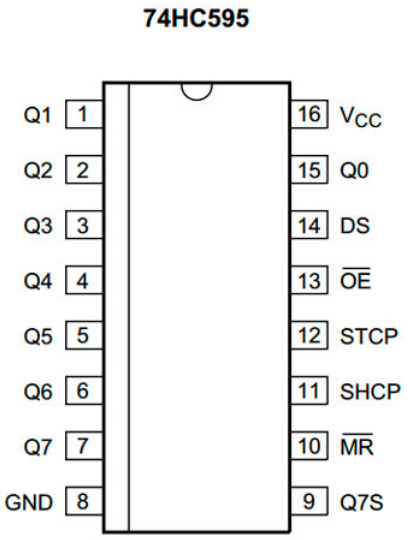
Aunque ahora todo parezca un poco confuso (no me extraña) son bastante sencillos de manejar una vez que entiendes lo que hacen y son sorprendentemente útiles en cantidad de situaciones.

Cómo funciona un Shift Register

Un Shift Register funciona mediante la comunicación **serie síncrona**. Es decir que usamos un pin para enviar los bits en serie (el **Data pin**) y usamos un segundo pin (el **Clock pin**) para indicar cuando hay que leer el bit.

Cuando los 8 bits se han leído en el registro un tercer pin (Latch pin) escribe estos bits en los pines de salida del chip y los mantiene hasta que
se reciban nuevos datos.



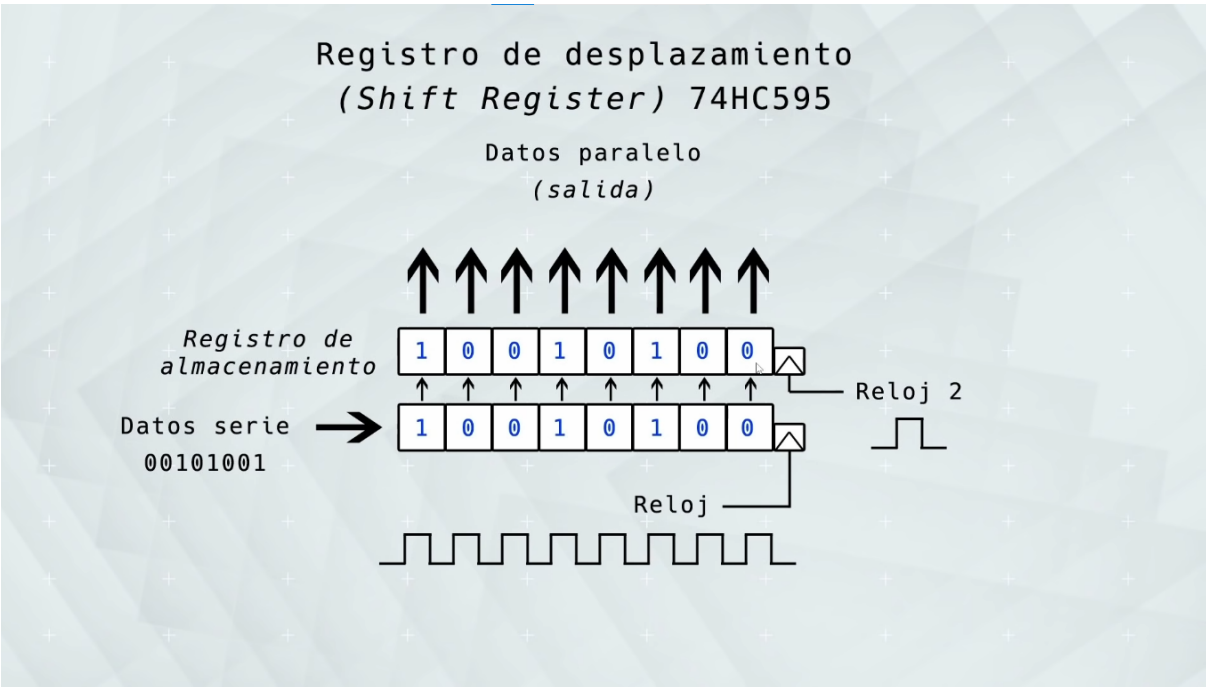


Fíjate en el gráfico superior. Nuestro **Arduino** envía la señal de **Clock** Pin de modo regular para indicar cuando hay que leer (Se lee en el flanco de subida, cuando **Clock** sube de 0 a 1).

A medida que **Arduino** va poniendo valores en el **Data** Pin(DS), el chip los va leyendo en el flanco de subida del **Clock** pin y por eso va a ir leyendo 1 1 1 0 0 0 1 1 sucesivamente.

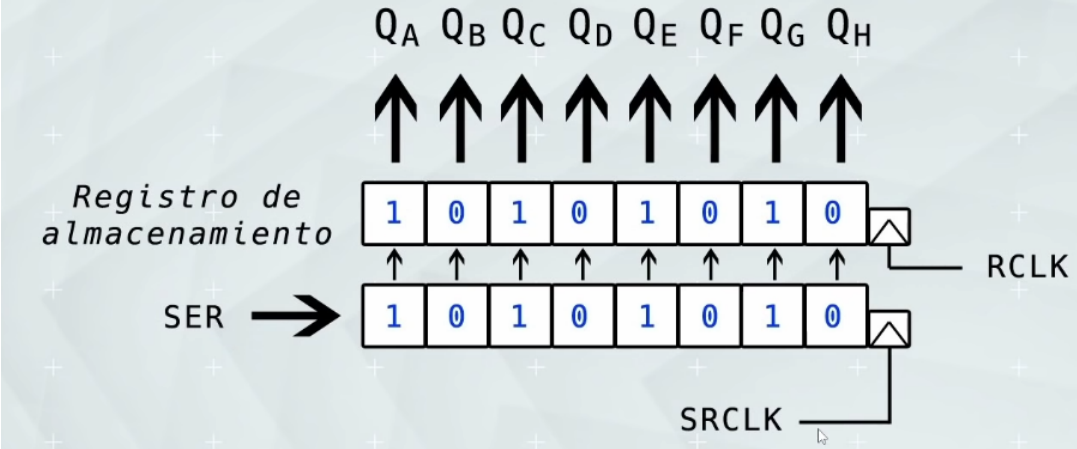
Cada uno de estos valores se van pasando en orden a los pines de salida de Q0 hasta Q7, pero aún no se activan. Cuando el Latch Pin se activa (también por flanco de subida) los valores pasan a los pines de salida y se memorizan.

Registro de desplazamiento *(Shift Register)* 74HC595

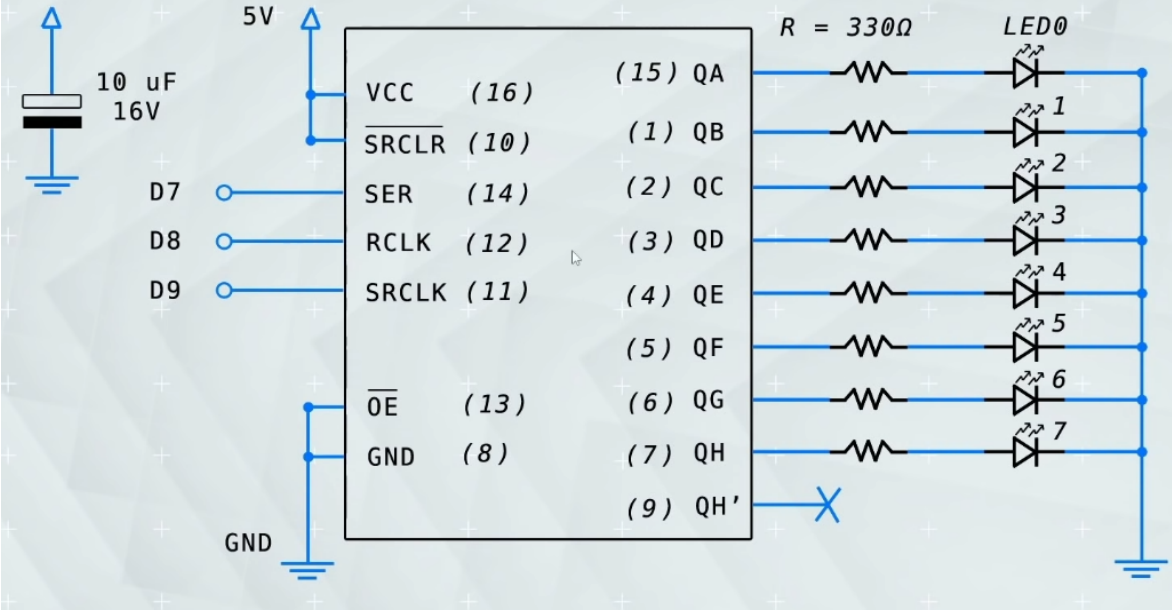


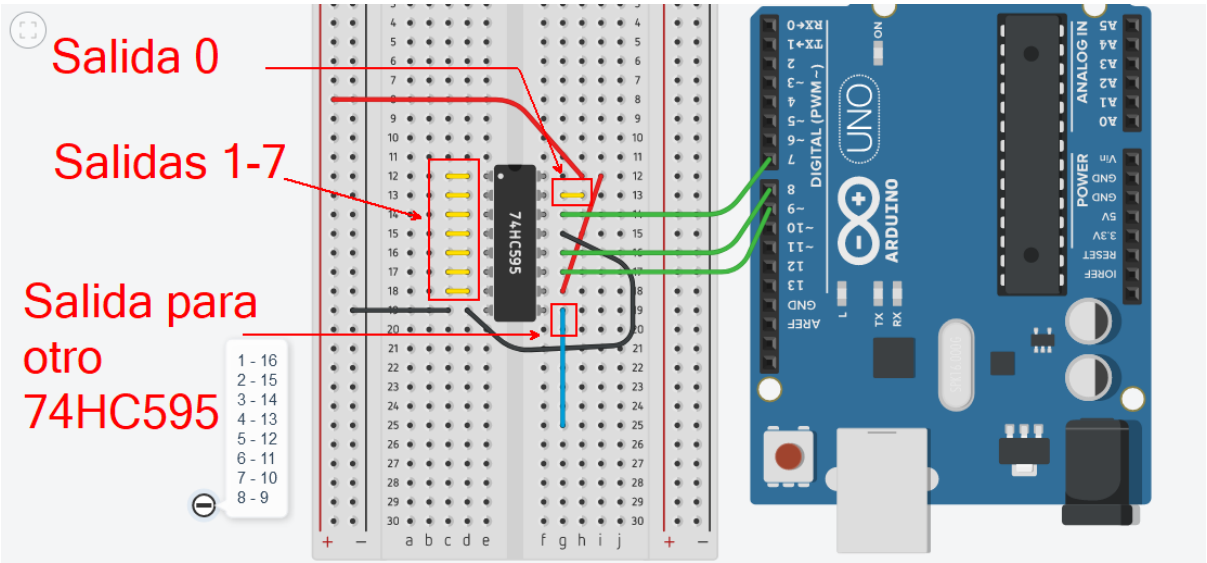
LED	7	6	5	4	3	2	1	0
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
en Decimal	128	64	32	16	8	4	2	1

Registro de desplazamiento *(Shift Register)* 74HC595



74HC595
Circuito de conexión *(esquemático)*




Salida 0
Salidas 1-7
Salida para otro 74HC595
1 - 16
2 - 15
3 - 14
4 - 13
5 - 12
6 - 11
7 - 10
8 - 9
74HC595
UNO
ARDUINO
DIGITAL (PWM~)
ANALOG IN
POWER