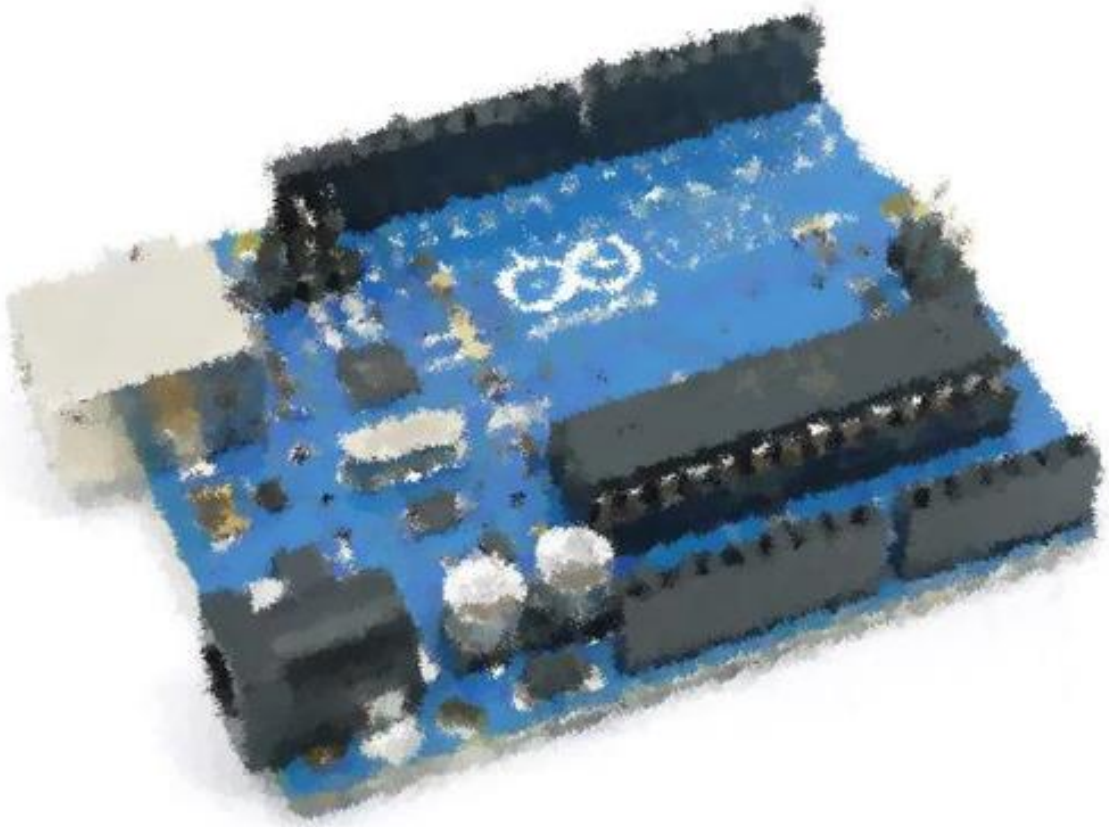
CAPÍTULO

5 nivel arduineitor

Genuino ARDUINO



Daniel Gallardo García
Profesor de Tecnología
Jerez de la Frontera

Índice

Daniel Gallardo García
Profesor de Tecnología
Jerez de la Frontera

Sensor Ultrasónico de Distancia

El **sensor de ultrasonido** se utiliza para medir distancias. El modelo comercial más común posee 4 patas: dos para alimentación (5 V y GND), una para el disparador de ultrasonido (Trigger) y otro para el receptor del eco del ultrasonido (Echo).

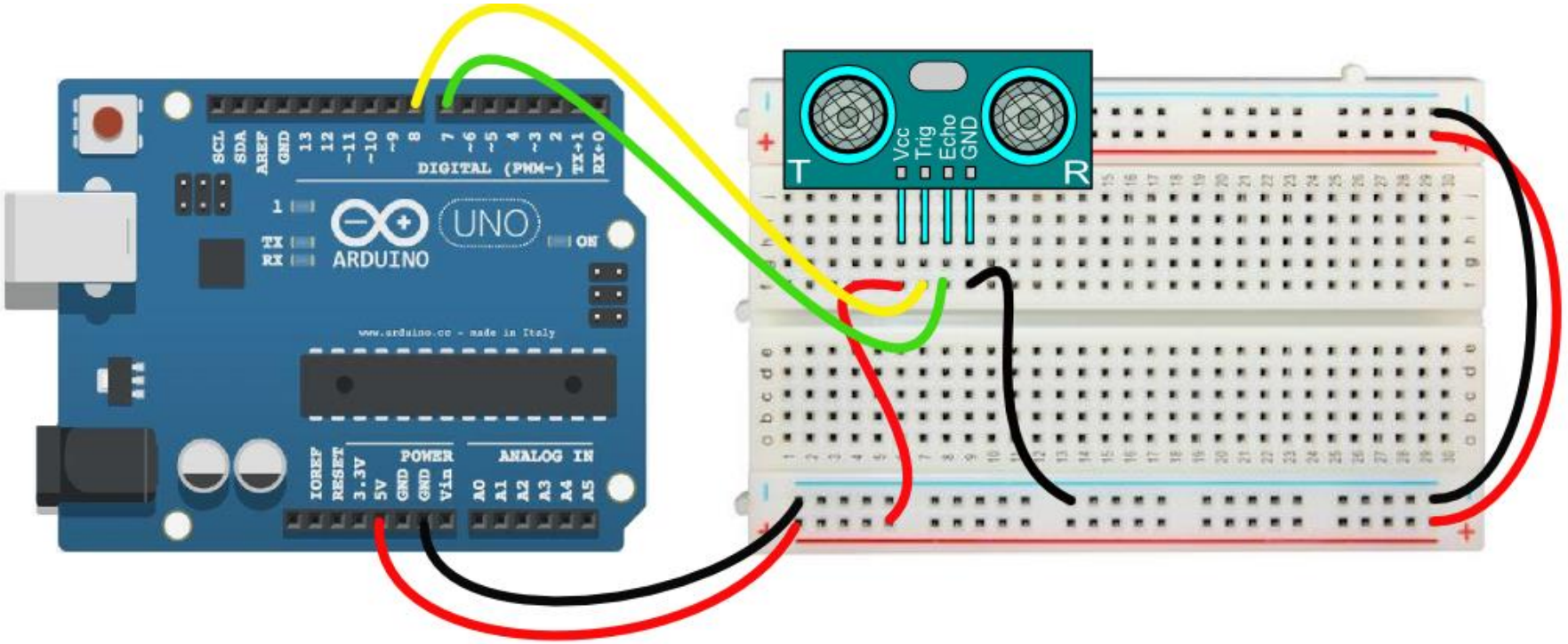
También existe otro modelo con solo tres patas: dos de alimentación y una tercera que habrá que declararla como `OUTPUT` cuando hagamos el disparo, y luego declararla como `INPUT` para recibir el rebote, y medir el tiempo a través de:

```
pulseIn(pinSensor, HIGH);
```

La idea es sencilla: con el disparador enviamos un pulso en `HIGH`, y calculamos el tiempo que tarda dicha señal en llegar rebotada al receptor. Luego hacemos una **conversión** entre los **milisegundos** que emplea la señal para llegar al receptor y los **centímetros** que habrá desde el sensor al objeto que produce el rebote (recordemos que la velocidad del sonido es de unos 340 m/s).

Ejemplo: Lectura de la distancia a un objeto

El montaje será el siguiente:



Ejemplo: Lectura de la distancia a un objeto_2

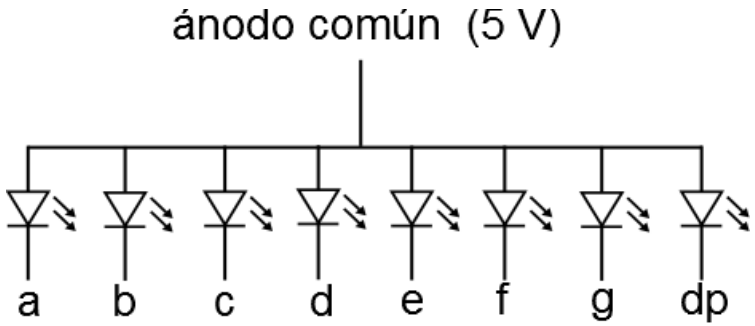
```
int disparo = 8, eco = 7;                          //pines para el sensor de distancia
long tiempoInicial, tiempoFinal, duracion;         /*variables que utilizaremos para
                                                   obtener el tiempo hasta el rebote */
int cm, senalEco;    //variables para los centímetros y para detectar el rebote

void setup() {
  Serial.begin(9600);
  pinMode(disparo, OUTPUT);
  pinMode(eco, INPUT);
}

void loop() {
         //Hago un disparo: lanzo un pulso de 5 us de duración
  digitalWrite(disparo, LOW);
  delayMicroseconds(2);
  digitalWrite(disparo, HIGH);
  delayMicroseconds(5);
  tiempoInicial = micros();          //pongo el "cronómetro" a cero
  digitalWrite(disparo, LOW);
         //Detectaré el tiempo que tarda en llegar el rebote
  senalEco = digitalRead(eco);
  while(senalEco == LOW) {
    senalEco = digitalRead(eco);     //para saber cuándo salgo del valor LOW
  }
```

Ejemplo: Lectura de la distancia a un objeto_3

```
  while(senalEco == HIGH) {
    senalEco = digitalRead(eco);
    tiempoFinal = micros();
  }
         //Calculo la duración del recorrido sensor-obstáculo-sensor
  duracion = tiempoFinal - tiempoInicial;
  cm = int(duracion / 58);  /*el sonido se desplaza a 340m/s o 29ms/cm
          y como tiene que recorrer dos veces la distancia hasta el objeto,
          deberemos dividir entre 2*29 los microsegundos transcurridos */
  Serial.print("tiempo empleado: "); Serial.print(duracion);
  Serial.print(" ms"); Serial.print('\t');
  Serial.print("distancia: "); Serial.print(cm); Serial.print(" cm");
                  /*expresaremos la distancia a través del Serial Monitor, pero se
                  podría hacer con un display de 2 dígitos */
  delay(1000);    /*una pausa es necesaria para poder leer con comodidad los
                  datos en el Serial Monitor */
}
```

Display de 7-segmentos

Un dispositivo de salida muy empleado es el display 7-segmentos, Empleado normalmente para mostrar en el un numero del 0 a 9.

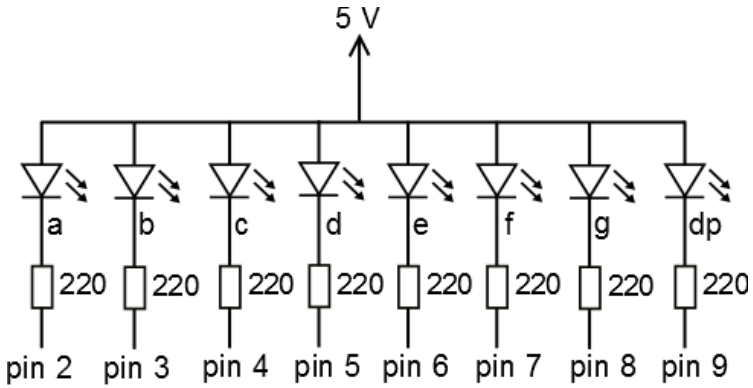


Este display posee **10 patillas**: los 7 segmentos que forman el número (a, b, c, d, e, f, g), el punto (dp), y dos pines para el ánodo común (también existen modelos de cátodo común).

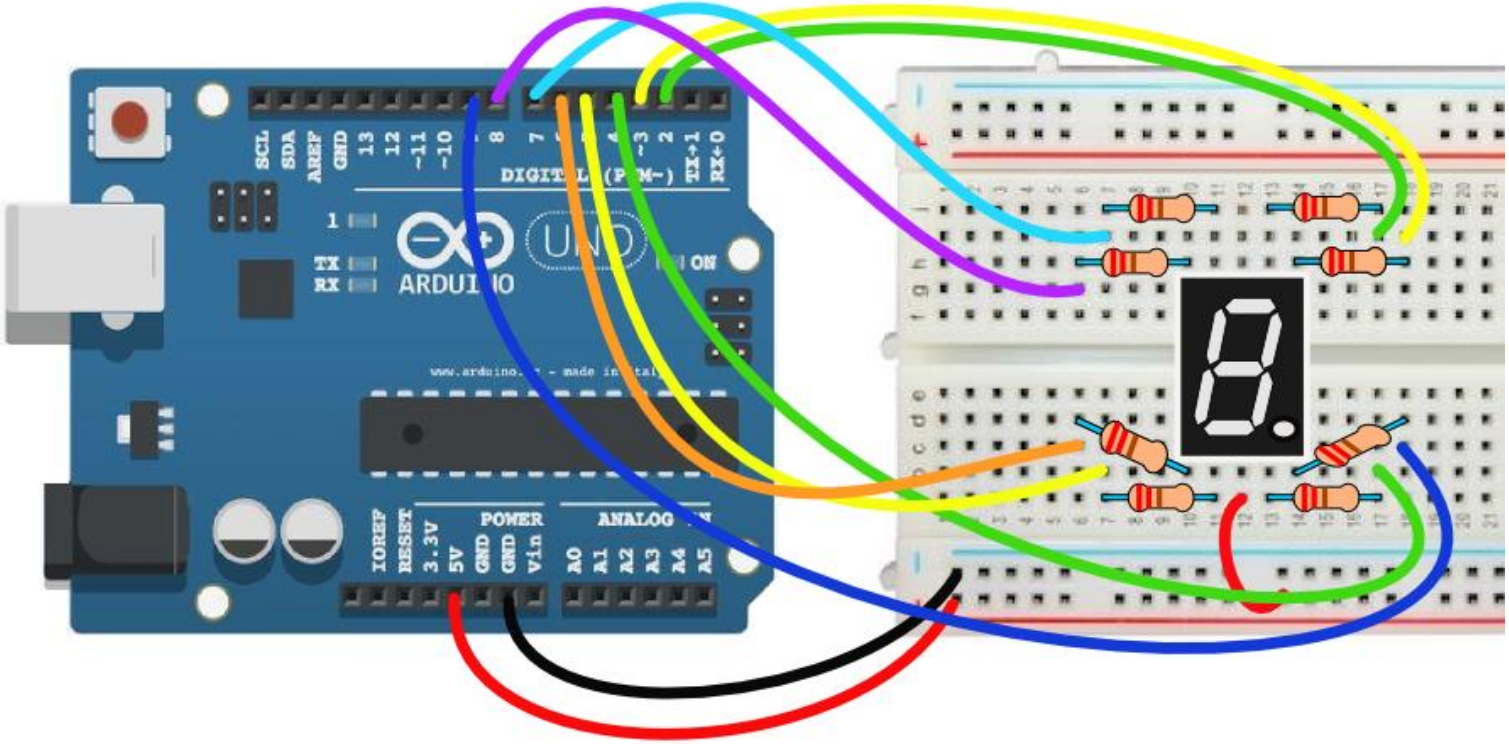
Debemos conectar nuestro display de la siguiente manera: conectaremos una de las dos patas de ánodo común a la tensión de 5 V, y cada una de las restantes patas (segmentos y el punto) a un pin de salida digital a través de una resistencia (de 220 o 470 Ω, por ejemplo).

De esta manera se encenderán los segmentos conectados a una salida digital de en nivel LOW.

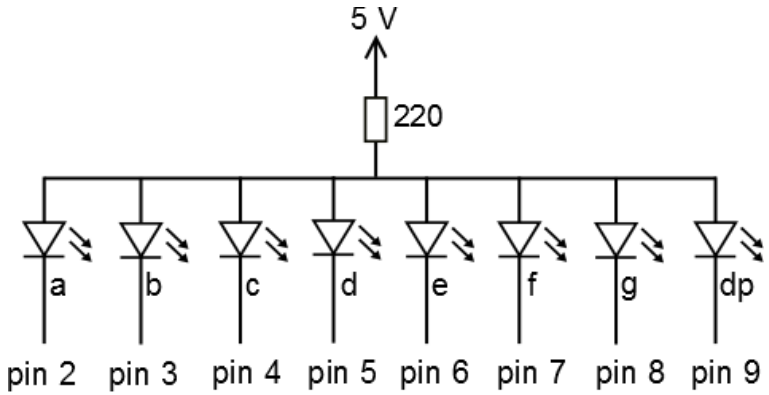
Display de 7-segmentos_2



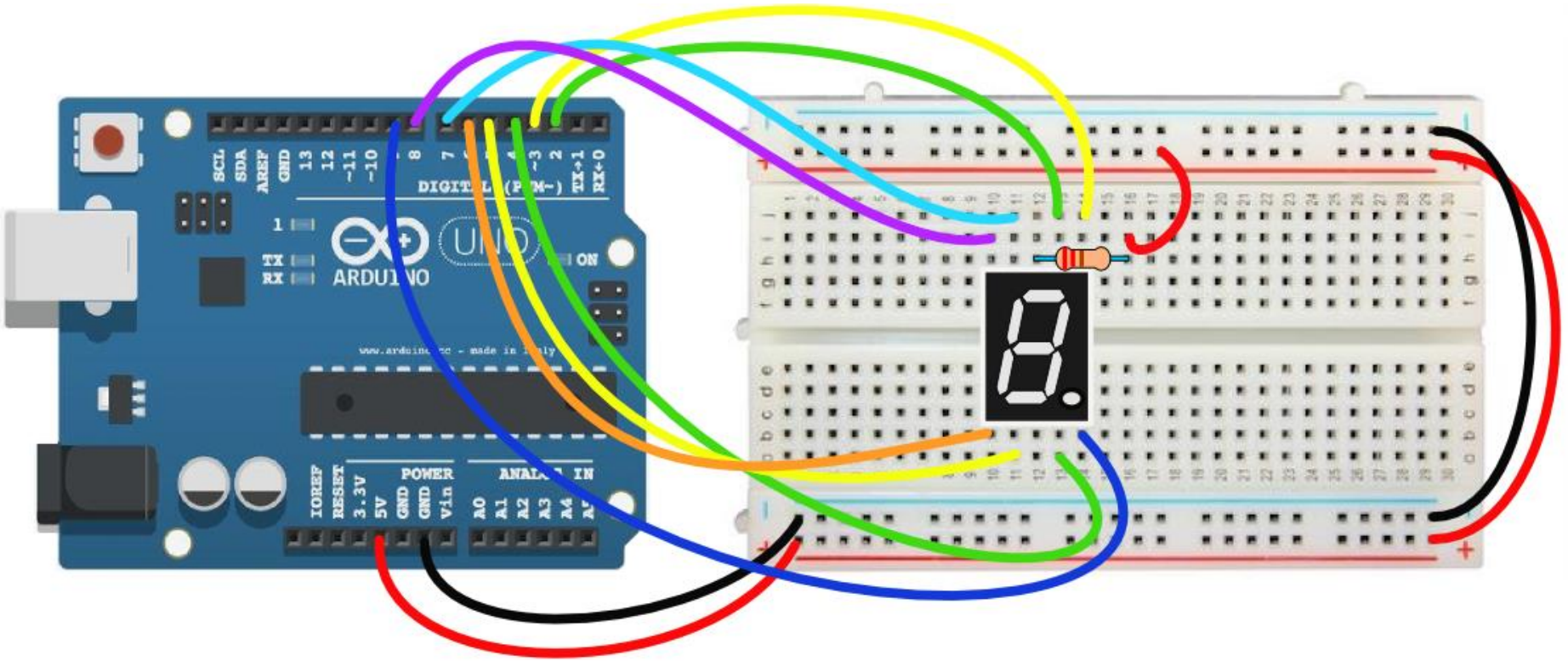
Debemos poner una resistencia para cada LED del display si no queremos que se note ninguna disminución en la intensidad de brillo en ninguno de ellos en el caso de encender más de uno a la vez.



Display de 7-segmentos_3



Si solo necesitamos encender uno a la vez, podemos realizar la siguiente configuración:

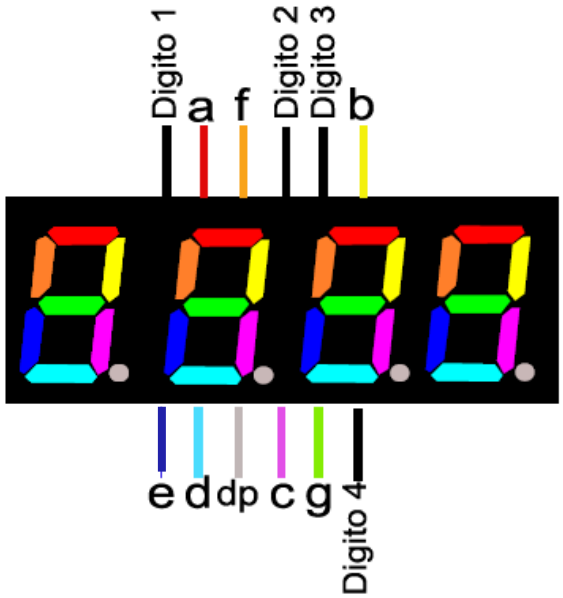


Ejemplo: La cuenta atrás

```
int pinSegmentos[] = {2, 3, 4, 5, 6, 7, 8, 9};          //(a, b, c, d, e, f, g, .)
byte segmentosNumero[10][8] = {  {1,1,1,1,1,1,0,0},     //número 0: segmentos a,b,c,d,e,f
                                 {0,1,1,0,0,0,0,0},     //número 1: segmentos b,c
                                 {1,1,0,1,1,0,1,0},     //número 2: segmentos a,b,g,e,d
                                 {1,1,1,1,0,0,1,0},     //número 3: etc…
                                 {0,1,1,0,0,1,1,0},     //número 4: etc…
                                 {1,0,1,1,0,1,1,0},     //número 5
                                 {1,0,1,1,1,1,1,0},     //número 6
                                 {1,1,1,0,0,0,0,0},     //número 7
                                 {1,1,1,1,1,1,1,0},     //número 8
                                 {1,1,1,1,0,1,1,0} };   //número 9

void setup() {
  for(int i=0; i<sizeof(pinSegmentos)/2; i++) pinMode(pinSegmentos[i], OUTPUT);
}

void loop() {
  for(int i=9; i>=0; i--) {                  //la cuenta atrás irá del 9 al 0
    for(int j=0; j<sizeof(pinSegmentos)/2; j++)
      digitalWrite(pinSegmentos[j], !segmentosNumero[i][j]);
           /*recordemos que a la hora de interpretar una señal digital 1, HIGH y TRUE
           es lo mismo. Asimismo, lo mismo ocurre con 0, LOW y FALSE.
           Como cada segmento encenderá cuando el pin esté en LOW, tal y como hemos
           definido nuestros números, debemos indicar que en el pin j escriba lo
           contrario de lo que indica: si es un 1 pues que ponga un LOW (o 0), y si es
           un 0 pues que ponga un HIGH (o 1) */
      delay(1000);
  }
}
```

Display de 4 Dígitos de 7-segmentos

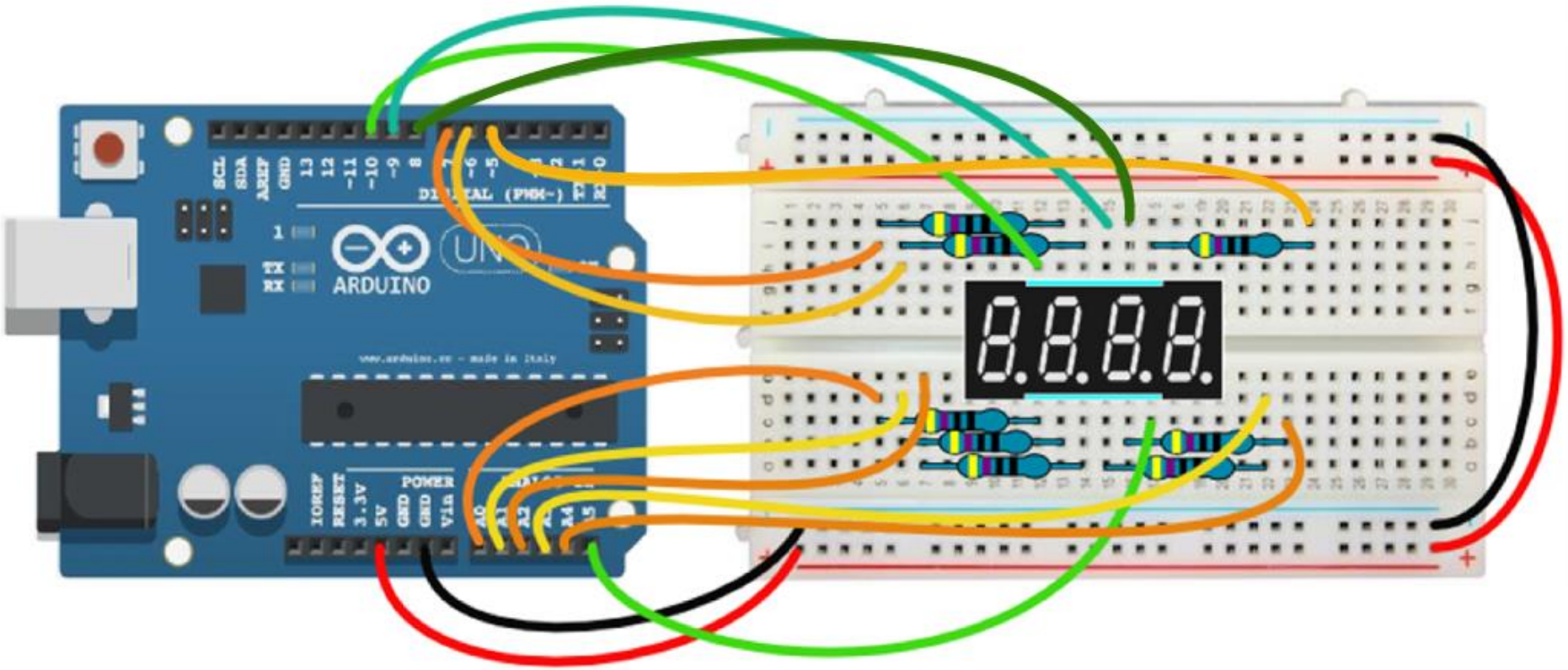
Este display se utiliza normalmente para mostrar en él números de 4 dígitos, como pudiera ser un reloj (minutos y segundos).



Los terminales `a`, `b`, `c`, `d`, `e`, `f` y `g` corresponden a los 7 segmentos (los vemos con colores), y `dp` será el punto (se encenderán con `LOW`).
Los terminales `Dígito1`, `Dígito 2`, `Dígito 3` y `Dígito 4` permitirán activar a cada uno de los dígitos (colocándolos en `HIGH`).
Será necesario, pues, ir haciendo un barrido por los cuatro dígitos (con un tiempo de unos 5 ms como mucho por dígito).

Ejemplo: Reloj

El montaje será el siguiente:



Ejemplo: Reloj_2

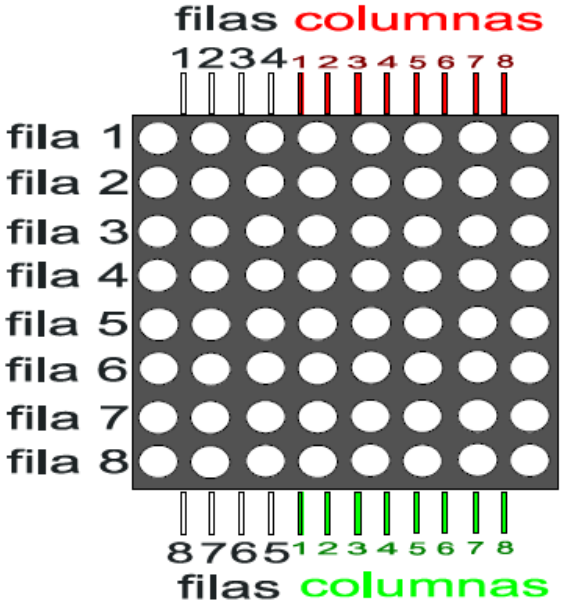
```
int segmentosPines[] = {7, 5, 17, 15, 14, 6, 18, 16};            //(a,b,c,d,e,f,g,.)
int digitosPines[] = {10, 9, 8, 19};                             //(dígito1,dígito2,dígito3,dígito4)
byte segmentosNumeros[10][8] = { {1, 1, 1, 1, 1, 1, 0, 0},       //número 0
                                 {0, 1, 1, 0, 0, 0, 0, 0},       //número 1
                                 {1, 1, 0, 1, 1, 0, 1, 0},       //número 2
                                 {1, 1, 1, 1, 0, 0, 1, 0},       //número 3
                                 {0, 1, 1, 0, 0, 1, 1, 0},       //número 4
                                 {1, 0, 1, 1, 0, 1, 1, 0},       //número 5
                                 {1, 0, 1, 1, 1, 1, 1, 0},       //número 6
                                 {1, 1, 1, 0, 0, 0, 0, 0},       //número 7
                                 {1, 1, 1, 1, 1, 1, 1, 0},       //número 8
                                 {1, 1, 1, 1, 0, 1, 1, 0} };     //número 9
int unidadesSeg = 0, decenasSeg = 0, unidadesMin = 0, decenasMin = 0;
long tiempo = 0, intervalo = 1000;          /*si pusiera intervalo=10 tendría un cronómetro
                                            con una precisión de centésimas de segundo*/

void setup() {
  for(int i=0; i<8; i++) pinMode(segmentosPines[i], OUTPUT);
  for(int j=0; j<4; j++) pinMode(digitosPines[j], OUTPUT);
}

void loop() {
  ponHora(unidadesSeg, decenasSeg, unidadesMin, decenasMin);
  if(tiempo + intervalo < millis()) {       //significaría que ha pasado 1 segundo
    unidadesSeg++;
    if(unidadesSeg == 10) {unidadesSeg = 0; decenasSeg++;}
    if(decenasSeg == 6) {decenasSeg = 0; unidadesMin++;}
    if(unidadesMin == 10) {unidadesMin = 0; decenasMin++;}
    if(decenasMin == 6) decenasMin = 0;
    tiempo = millis();
  }
}
```

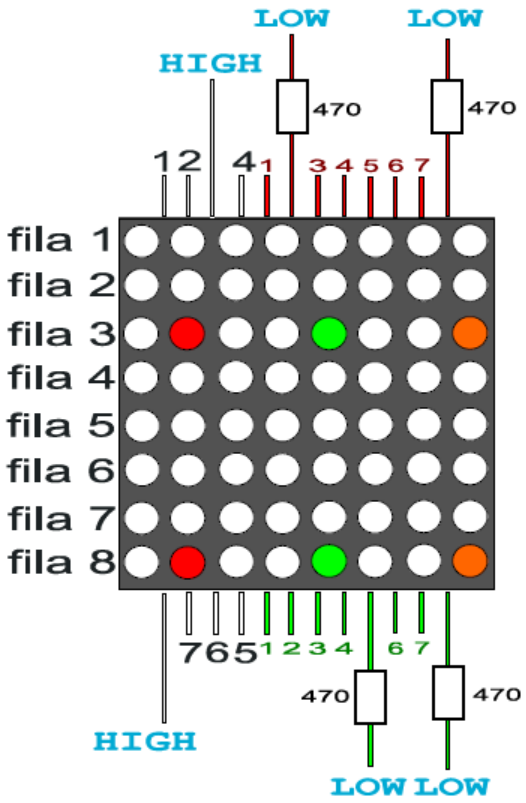
Ejemplo: Reloj_3

```
void ponHora(int x, int y, int z, int t) {
  int valores[] = {t, z, y, x};
  for(int pin=0; pin<4; pin++) { /*haré todo dentro de un for: iré llamando a cada uno de los
                                  dígitos e iré colocando en el display su valor correspondiente*/
    for(int i=0; i<4; i++) digitalWrite(digitosPines[i], LOW);
                                  //pongo a LOW todos los pines de dígitos y solo activo uno
    digitalWrite(digitosPines[pin], HIGH);
    for(int j=0; j<8; j++) digitalWrite(segmentosPines[j], !segmentosNumeros[valores[pin]][j]);
    int punto = x % 2;            /*separaré minutos y segundos con un punto parpadeante
                                  del segundo dígito (pin=1) */
    if(punto == 0 && pin == 1) digitalWrite(segmentosPines[7], LOW);
    delay(5);                     /*entre dígito y dígito debemos hacer una pequeña pausa de
                                  entre 1-5 ms. Sin pausa, la lectura es confusa, y con más de
                                  5 ms molesta la lentitud de refresco*/

  }
}
```

Matriz de LEDs 8x8

Otro dispositivo muy usado es la matriz de LEDs, en este caso una 8x8 bicolor (64 LEDs rojos y 64 LEDs verdes), con 24 pines de control. Un esquema de cómo hemos de conectar dicha matriz es el siguiente:





La forma de poder representar una imagen formada por estos 64 puntos es hacer barridos por las distintas filas o columnas, de manera análoga a como se trabajaba con el display de 4 dígitos. Para evitar variaciones en la intensidad de brillo de los LEDs, **si el barrido lo hacemos por filas** deberemos colocar las **resistencias** limitadoras de corriente (120 – 470 Ω) en los pines reservados a las **columnas** (como aparece en la figura), pero **si el barrido lo hacemos por columnas** deberá ser en los pines reservados para las **filas** donde intercalemos las **resistencias**.

Ejemplo: Cartel de Farmacia

Por la matriz de LEDs aparecerá un mensaje dinámico (como los carteles de muchas farmacias). En este caso, pondremos las resistencias en los pines de control de filas. El programa podría ser el siguiente:

```
int numeroPantallas = 20;         /*numero de veces que repite una "pantalla" antes de desplazarla un
                                  lugar. Evidentemente este parámetro decidirá también la velocidad
                                  de avance del texto por la matriz */
int tiempoBarridoColumna = 500;   /*microseconds en el que está activa cada una de las columnas.
                                  Tardará 8*500 us en barrer toda la "pantalla" */
/* AQUÍ ES DONDE DEBEMOS ESCRIBIR LA FRASE QUE QUEREMOS QUE "CIRCULE" POR NUESTRA MATRIZ 8X8
     Se debe empezar por un doble espacio, y se sustituyen la ñ por “`” y la Ñ por “^”
     (ambos símbolos se encuentran en la misma tecla) */
char fraseDinamica[] = "  Esto parece una FARMACIA DE GUARDIA + + + ";
int index = 1;          //será la posición del caracter de la cadena que está siendo mostrado
int offset = 0;         //marcará el deslizamiento de las columnas

int filaMatriz[] = {13,12,11,10,19,18,17,16};  //pines para el control de las filas de la matriz
int columnaMatriz[] = {9,8,7,6,5,4,3,2};       //pines para el control de las columnas de LED rojos

//Necesitaremos especificar la posición de cada caracter en letras[][] declarado más abajo
//comenzamos por los números
const int cero=0, uno=1, dos=2, tres=3, cuatro=4, cinco=5, seis=6, siete=7, ocho=8, nueve=9;
//las letras minúsculas
const int a=10, b=11, c=12, d=13, e=14, f=15, g=16, h=17, i=18, j=19, k=20, l=21, m=22, n=23,
          enhe=24, o=25, p=26, q=27, r=28, s=29, t=30, u=31, v=32, w=33, x=34, y=35, z=36;
//las letras mayúsculas
const int A=37, B=38, C=39, D=40, E=41, F=42, G=43, H=44, I=45, J=46, K=47, L=48, M=49, N=50,
          ENHE=51, O=52, P=53, Q=54, R=55, S=56, T=57, U=58, V=59, W=60, X=61, Y=62, Z=63;
//y algunos símbolos
const int espacio=64,exclamacion=65,comillas=66,almohadilla=67,dollar=68,porcentaje=69,
```

Ejemplo: Cartel de Farmacia_2

```
//y algunos símbolos
const int espacio=64, exclamacion=65, comillas=66, almohadilla=67, dollar=68, porcentaje=69,
          ampersand=70, parentesisAbierto=71, parentesisCerrado=72, asterisco=73, mas=74,
          coma=75, menos=76, barra=77, dosPuntos=78, puntoComa=79, menor=80, igual=81, mayor=82,
          interroganteAbierto=83, interroganteCerrado=84, arroba=85, guionBajo=86, punto=87;

byte datos[] = {0,0,0,0,0,0,0,0};  /*este array almacenará los 8 números (de 8 cifras cada uno)
                                     que configuran las 8 columnas mostradas en cada "pantallazo" */

//construiremos los bitmap (8x7) de cada caracter, siendo 1=ON y 0=OFF
//primero los números (el _ será para distinguir a los bitmaps de los “posicionadores”
const int _cero[] = {B01110, B10001, B10011, B10101, B11001, B10001, B01110, B00000};
const int _uno[] = {B00100, B01100, B00100, B00100, B00100, B00100, B01110, B00000};
const int _dos[] = {B01110, B10001, B00001, B00010, B00100, B01000, B11111, B00000};
const int _tres[] = {B11111, B00010, B00100, B00010, B00001, B10001, B01110, B00000};
const int _cuatro[] = {B00010, B00110, B01010, B10010, B11111, B00010, B00010, B00000};
const int _cinco[] = {B11111, B10000, B11110, B00001, B00001, B10001, B01110, B00000};
const int _seis[] = {B00110, B01000, B10000, B11110, B10001, B10001, B01110, B00000};
const int _siete[] = {B11111, B10001, B00010, B00100, B01000, B01000, B01000, B00000};
const int _ocho[] = {B01110, B10001, B10001, B01110, B10001, B10001, B01110, B00000};
const int _nueve[] = {B01110, B10001, B10001, B01111, B00001, B00010, B01100, B00000};
//las letras minúsculas
const int _a[] = {B00000, B00000, B01110, B00001, B01111, B10001, B01111, B00000};
const int _b[] = {B10000, B10000, B10110, B11001, B10001, B10001, B11110, B00000};
const int _c[] = {B00000, B00000, B01110, B10000, B10000, B10001, B01110, B00000};
const int _d[] = {B00001, B00001, B01101, B10011, B10001, B10001, B01111, B00000};
const int _e[] = {B00000, B00000, B01110, B10001, B11111, B10000, B01110, B00000};
const int _f[] = {B00110, B01001, B01000, B11100, B01000, B01000, B01000, B01000};
const int _g[] = {B00000, B00000, B01111, B10001, B01111, B00001, B00001, B01110};
const int _h[] = {B10000, B10000, B10110, B11001, B10001, B10001, B10001, B00000};
```

Ejemplo: Cartel de Farmacia_3

```
const int _i[] = {B00100, B00000, B00100, B01100, B00100, B00100, B01110, B00000};
const int _j[] = {B00010, B00000, B00110, B00010, B00010, B00010, B10010, B01100};
const int _k[] = {B01000, B01000, B01001, B01010, B01100, B01010, B01001, B00000};
const int _l[] = {B01100, B00100, B00100, B00100, B00100, B00100, B01110, B00000};
const int _m[] = {B00000, B00000, B11010, B10101, B10101, B10101, B10101, B00000};
const int _n[] = {B00000, B00000, B10110, B11001, B10001, B10001, B10001, B00000};
const int _enhe[] = {B01110, B00000, B10110, B11001, B10001, B10001, B10001, B00000};
const int _o[] = {B00000, B00000, B01110, B10001, B10001, B10001, B01110, B00000};
const int _p[] = {B00000, B00000, B11110, B10001, B11110, B10000, B10000, B10000};
const int _q[] = {B00000, B00000, B01101, B10011, B01111, B00001, B00011, B00001};
const int _r[] = {B00000, B00000, B10110, B11001, B10000, B10000, B10000, B00000};
const int _s[] = {B00000, B00000, B01110, B10000, B01110, B00001, B11110, B00000};
const int _t[] = {B01000, B01000, B11100, B01000, B01000, B01001, B00110, B00000};
const int _u[] = {B00000, B00000, B10001, B10001, B10001, B10011, B01101, B00000};
const int _v[] = {B00000, B00000, B10001, B10001, B10001, B01010, B00100, B00000};
const int _w[] = {B00000, B00000, B10001, B10001, B10101, B10101, B01010, B00000};
const int _x[] = {B00000, B00000, B10001, B01010, B00100, B01010, B10001, B00000};
const int _y[] = {B00000, B00000, B10001, B10001, B01111, B00001, B00001, B01110};
const int _z[] = {B00000, B00000, B11111, B00010, B00100, B01000, B11111, B00000};
//las letras mayúsculas
const int _A[] = {B01110, B10001, B10001, B10001, B11111, B10001, B10001, B00000};
const int _B[] = {B11110, B10001, B10001, B11110, B10001, B10001, B11110, B00000};
const int _C[] = {B01110, B10001, B10000, B10000, B10000, B10001, B01110, B00000};
const int _D[] = {B11110, B10001, B10001, B10001, B10001, B10001, B11110, B00000};
const int _E[] = {B11111, B10000, B10000, B11110, B10000, B10000, B11111, B00000};
const int _F[] = {B11111, B10000, B10000, B11110, B10000, B10000, B10000, B00000};
const int _G[] = {B01110, B10001, B10000, B10011, B10001, B10001, B01111, B00000};
const int _H[] = {B10001, B10001, B10001, B11111, B10001, B10001, B10001, B00000};
const int _I[] = {B01110, B00100, B00100, B00100, B00100, B00100, B01110, B00000};
const int _J[] = {B00111, B00010, B00010, B00010, B00010, B10010, B01100, B00000};
```

Ejemplo: Cartel de Farmacia_4

```
const int _K[] = {B10001, B10010, B10100, B11000, B10100, B10010, B10001, B00000};
const int _L[] = {B10000, B10000, B10000, B10000, B10000, B10000, B11111, B00000};
const int _M[] = {B10001, B11011, B10101, B10101, B10001, B10001, B10001, B00000};
const int _N[] = {B10001, B10001, B11001, B10101, B10011, B10001, B10001, B00000};
const int _ENHE[] = {B01110, B00000, B10001, B11001, B10101, B10011, B10001, B00000};
const int _O[] = {B01110, B10001, B10001, B10001, B10001, B10001, B01110, B00000};
const int _P[] = {B11110, B10001, B10001, B11110, B10000, B10000, B10000, B00000};
const int _Q[] = {B01110, B10001, B10001, B10001, B10101, B10010, B01101, B00000};
const int _R[] = {B11110, B10001, B10001, B11110, B10100, B10010, B10001, B00000};
const int _S[] = {B01110, B10001, B10000, B01110, B00001, B10001, B01110, B00000};
const int _T[] = {B11111, B00100, B00100, B00100, B00100, B00100, B00100, B00000};
const int _U[] = {B10001, B10001, B10001, B10001, B10001, B10001, B01110, B00000};
const int _V[] = {B10001, B10001, B10001, B10001, B10001, B01010, B00100, B00000};
const int _W[] = {B10001, B10001, B10001, B10101, B10101, B10101, B01010, B00000};
const int _X[] = {B10001, B10001, B01010, B00100, B01010, B10001, B10001, B00000};
const int _Y[] = {B10001, B10001, B10001, B01010, B00100, B00100, B00100, B00000};
const int _Z[] = {B11111, B00001, B00010, B00100, B01000, B10000, B11111, B00000};
//caracteres especiales
const int _espacio[] = {B00000, B00000, B00000, B00000, B00000, B00000, B00000, B00000};
const int _exclamacion[] = {B00100, B00100, B00100, B00100, B00100, B00000, B00100, B00000};
const int _comillas[] = {B01010, B01010, B01010, B00000, B00000, B00000, B00000, B00000};
const int _almohadilla[] = {B01010, B01010, B11111, B01010, B11111, B01010, B01010, B00000};
const int _dollar[] = {B00100, B01111, B10100, B01110, B00101, B11110, B00100, B00000};
const int _porcentaje[] = {B11000, B11001, B00010, B00100, B01000, B10011, B00011, B00000};
const int _ampersand[] = {B01100, B10010, B10100, B01000, B10101, B10010, B01101, B00000};
const int _parentesisAbierto[] = {B00010, B00100, B01000, B01000, B01000, B00100, B00010, B00000};
const int _parentesisCerrado[] = {B01000, B00100, B00010, B00010, B00010, B00100, B01000, B00000};
const int _asterisco[] = {B00000, B00000, B00100, B10101, B01110, B10101, B00100, B00000};
const int _mas[] = {B00000, B00000, B00100, B00100, B11111, B00100, B00100, B00000};
const int _coma[] = {B00000, B00000, B00000, B00000, B00000, B01100, B00100, B01000};
```

Ejemplo: Cartel de Farmacia_5

```
const int _menos[] = {B00000, B00000, B00000, B00000, B11111, B00000, B00000, B00000};
const int _barra[] = {B00000, B00000, B00001, B00010, B00100, B01000, B10000, B00000};
const int _dosPuntos[] = {B00000, B01100, B01100, B00000, B01100, B01100, B00000, B00000};
const int _puntoComa[] = {B00000, B00000, B01100, B01100, B00000, B01100, B00100, B01000};
const int _menor[] = {B00010, B00100, B01000, B10000, B01000, B00100, B00010, B00000};
const int _igual[] = {B00000, B00000, B11111, B00000, B11111, B00000, B00000, B00000};
const int _mayor[] = {B01000, B00100, B00010, B00001, B00010, B00100, B01000, B00000};
const int _interroganteAbierto[] = {B00100, B00000, B00100, B01000, B10000, B10001, B01110, B00000};
const int _interroganteCerrado[] = {B01110, B10001, B00001, B00010, B00100, B00000, B00100, B00000};
const int _arroba[] = {B01110, B10001, B00001, B01101, B10101, B10101, B01110, B00000};
const int _guionBajo[] = {B00000, B00000, B00000, B00000, B00000, B00000, B11111, B00000};
const int _punto[] = {B00000, B00000, B00000, B00000, B00000, B01100, B01100, B00000};

/*declaro el abecedario: un array con los arrays de los bitmap de los distintos carateres que voy a
mostrar en la matriz. Como los elementos del array son cadenas de caracteres, utilizo int*   */
const int* letras[] = { _cero,_uno,_dos,_tres,_cuatro,_cinco,_seis,_siete,_ocho,_nueve, //del 0 al 9
  _a, _b, _c, _d, _e, _f, _g, _h, _i, _j,                                    //del 10 al 19
  _k, _l, _m, _n, _enhe, _o, _p, _q, _r, _s,                                 //del 20 al 29
  _t, _u, _v, _w, _x, _y, _z, _A, _B, _C,                                    //del 30 al 39
  _D, _E, _F, _G, _H, _I, _J, _K, _L, _M,                                    //del 40 al 49
  _N, _ENHE, _O, _P, _Q, _R, _S, _T, _U, _V,                                 //del 50 al 59
  _W, _X, _Y, _Z, _espacio,_exclamacion,_comillas,_almohadilla,_dollar,_porcentaje,//del 60 al 69
  _ampersand, _parentesisAbierto, _parentesisCerrado, _asterisco, _mas,
  _coma, _menos, _barra, _dosPuntos, _puntoComa,                             //del 70 al 79
  _menor, _igual, _mayor, _interroganteAbierto, _interroganteCerrado,
  _arroba, _guionBajo, _punto };                                             //del 80 al 86
```

Ejemplo: Cartel de Farmacia_6

```
void setup() {
  for(int i=0; i<8; i++) {                  //configuro los pines de control
    pinMode(filaMatriz[i], OUTPUT);
    pinMode(columnaMatriz[i], OUTPUT);
  }
}

void loop() {
  actualizaMatriz();
}

void actualizaMatriz() {
  construyePantalla();
  muestraPantalla(numeroPantallas);
}

//Utilizaré el siguiente array con potencias de 2 para manejar los bits de cada bitmap
const int potencias[] = {1,2,4,8,16,32,64,128};
            //cada número corresponde, en binario, a 00000001, 00000010, 00000100,..., 10000000

void construyePantalla() {     //carga el actual estado de desplazamiento al array datos[]
  int caracterAnterior = cogeCaracter(fraseDinamica[index-1]); //por esto empezamos en index=1
  int caracterActual = cogeCaracter(fraseDinamica[index]);
  int caracterPosterior = cogeCaracter(fraseDinamica[index+1]);
  for(int fila=0; fila<8; fila++) {         //hago un barrido por la fila
    datos[fila] = 0;                        //reseteo a 0 la fila que estoy considerando
    for(int columna=0; columna<8; columna++) { //hago un barrido por la columna
    /*construiré un número decimal a base de sumas de tal forma que tenga unos en los lugares
    donde quiero que se encienda el LED */
```

Ejemplo: Cartel de Farmacia_7

```
      datos[fila] = datos[fila] +
                    (potencias[columna] & (letras[caracterAnterior][fila] << (offset+7) ));
    /*cuando muestro un caracter en el borde derecho de la matriz (offset==0), se deberían mostrar
     en las dos columnas del borde izquierdo las dos últimas columnas del caracter anterior */
      datos[fila] = datos[fila] +
           (potencias[columna] & (letras[caracterActual][fila] << offset ));
    //muestro el caracter actual, y lo voy desplazando a través del offset hacia la izquierda
      datos[fila] = datos[fila] +
           (potencias[columna] & (letras[caracterPosterior][fila] >> (7-offset) ));
    /*cuando voy desplazando el caracter actual, irá apareciendo por la derecha las primeras
    columnas del caracter posterior. Se ha puesto (offset+7) y (7-offset) porque los caracteres
    tienen una anchura de 5 columnas, y quiero que entre caracter y caracter haya 2 columnas
    de separación */
     }
   }
   offset++;                                  //he terminado una fila, pasaría a construir la siguiente
   if(offset == 6) {                          //significaría que he desplazado una letra entera
     offset=0; index++;                       //cargo un caracter nuevo y pongo offset a cero
     if(index == sizeof(fraseDinamica)-2) index = 1;    //si termina la frase, la reinicio
   }
}

void muestraPantalla(int repeticiones) {
  for(int n=0; n<repeticiones; n++) {       //muestra la pantalla acutal un número de repeticones
    for(int columna=0; columna<8; columna++) { //haré el barrido de la matriz columna por columna
      for(int i=0; i<8; i++) digitalWrite(filaMatriz[i], LOW);  /*pone a LOW todos los pines de
                                                                  control de filas */
      for(int i=0; i<8; i++) { //solo pondré en LOW el pin de la columna que estamos considerando
        if(i ==c olumna) digitalWrite(columnaMatriz[i], LOW);
        else digitalWrite(columnaMatriz[i], HIGH);     //el resto de pines los pongo a HIGH…
      }                                                //… (apago la columna de LEDs)
```

Ejemplo: Cartel de Farmacia_8

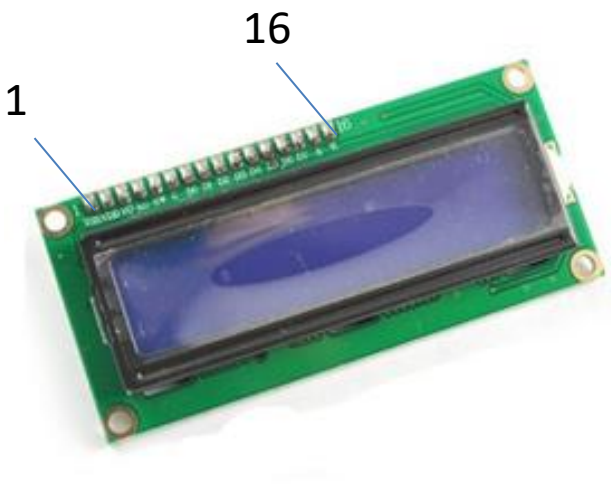
```
      for(int fila=0; fila<8; fila++) { //haré el barrido, dentro de la columna activa, por la fila
        int Bit = (datos[columna] >> fila) & 1;       /*detectaré los "1" de las fila si al
                                                      multiplicarlo por 1 da 1 */
        if(Bit == 1) digitalWrite(filaMatriz[fila], HIGH); /*si el bit en el array datos[] es un 1,
                                                           encenderá el LED  */
      }
      delayMicroseconds(tiempoBarridoColumna);        //es el tiempo que cada columna está encendida
    }
  }
}
          /* veamos cómo asociamos cada caracter de la cadena con su bitmap: asocio a cada caracter
          la variable correspondiente que almacena su bitmap */
int cogeCaracter(char caracter) {
  int returnValue;
  switch(caracter) {
    case '0': returnValue = cero; break;      case '1': returnValue = uno; break;
    case '2': returnValue = dos; break;       case '3': returnValue = tres; break;
    case '4': returnValue = cuatro; break;    case '5': returnValue = cinco; break;
    case '6': returnValue = seis; break;      case '7': returnValue = siete; break;
    case '8': returnValue = ocho; break;      case '9': returnValue = nueve; break;
    case 'A': returnValue = A; break;         case 'a': returnValue = a; break;
    case 'B': returnValue = B; break;         case 'b': returnValue = b; break;
    case 'C': returnValue = C; break;         case 'c': returnValue = c; break;
    case 'D': returnValue = D; break;         case 'd': returnValue = d; break;
    case 'E': returnValue = E; break;         case 'e': returnValue = e; break;
    case 'F': returnValue = F; break;         case 'f': returnValue = f; break;
    case 'G': returnValue = G; break;         case 'g': returnValue = g; break;
    case 'H': returnValue = H; break;         case 'h': returnValue = h; break;
    case 'I': returnValue = I; break;         case 'i': returnValue = i; break;
    case 'J': returnValue = J; break;         case 'j': returnValue = j; break;
    case 'K': returnValue = K; break;         case 'k': returnValue = k; break;
```

Ejemplo: Cartel de Farmacia_9

```
    case 'L': returnValue = L; break;          case 'l': returnValue = l; break;
    case 'M': returnValue = M; break;          case 'm': returnValue = m; break;
    case 'N': returnValue = N; break;          case 'n': returnValue = n; break;
    case '^': returnValue = ENHE; break;       case '`': returnValue = enhe; break;
    case 'O': returnValue = O; break;          case 'o': returnValue = o; break;
    case 'P': returnValue = P; break;          case 'p': returnValue = p; break;
    case 'Q': returnValue = Q; break;          case 'q': returnValue = q; break;
    case 'R': returnValue = R; break;          case 'r': returnValue = r; break;
    case 'S': returnValue = S; break;          case 's': returnValue = s; break;
    case 'T': returnValue = T; break;          case 't': returnValue = t; break;
    case 'U': returnValue = U; break;          case 'u': returnValue = u; break;
    case 'V': returnValue = V; break;          case 'v': returnValue = v; break;
    case 'W': returnValue = W; break;          case 'w': returnValue = w; break;
    case 'X': returnValue = X; break;          case 'x': returnValue = x; break;
    case 'Y': returnValue = Y; break;          case 'y': returnValue = y; break;
    case 'Z': returnValue = Z; break;          case 'z': returnValue = z; break;
    case ' ': returnValue = espacio; break;    case '!': returnValue = exclamacion; break;
    case '"': returnValue = comillas; break;   case '#': returnValue = almohadilla; break;
    case '$': returnValue = dollar; break;     case '%': returnValue = porcentaje; break;
    case '&': returnValue = ampersand; break;  case '(': returnValue = parentesisAbierto; break;
    case ')': returnValue = parentesisCerrado; break;    case '*': returnValue = asterisco; break;
    case '+': returnValue = mas; break;        case ',': returnValue = coma; break;
    case '-': returnValue = menos; break;      case '/': returnValue = barra; break;
    case ':': returnValue = dosPuntos; break;  case ';': returnValue = puntoComa; break;
    case '<': returnValue = menor; break;      case '=': returnValue = igual; break;
    case '>': returnValue = mayor; break;      case '¿': returnValue = interroganteAbierto; break;
    case '?': returnValue = interroganteCerrado; break;  case '@': returnValue = arroba; break;
    case '_': returnValue = guionBajo; break;  case '.': returnValue = punto; break;
  }
  return returnValue;
}
```

Pantalla LCD 16x2

Uno de los dispositivos más espectaculares para nuestra Arduino es sin duda la pantalla LCD. En el mercado podemos encontrar gran variedad, pero el modelo HD44780 es el más extendido. Consta de 16 pines, pero solo necesitamos conectar 6 con los pines de salida de Arduino.

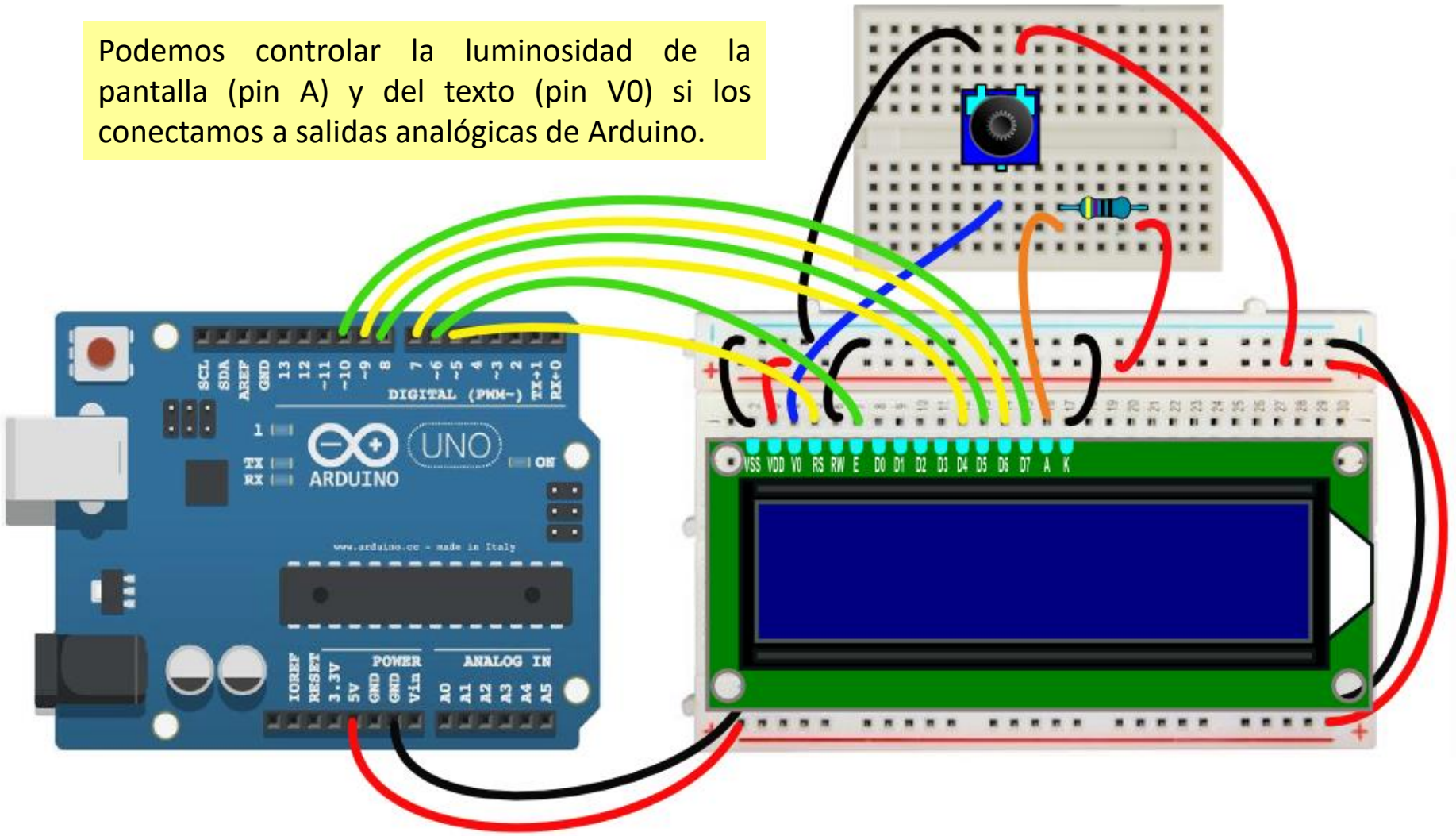


Pines de la LCD		
1	**VSS**	0 V
2	**VDD**	5 V
3	**V0**	Potenciómetro o una resistencia de 2,2K a 0 V para controlar la luminosidad de los caracteres
4	**RS**	Register select
5	**R/W**	0 V (solo vamos a escribir, no a leer datos)
6	**E**	Enable
7	DB0	(sólo para transmitir datos a 8 bits, cosa que no será necesaria nunca)
8	DB1	
9	DB2	
10	DB3	
11	**DB4**	BUS de transmisión de datos a 4 bits
12	**DB5**	
13	**DB6**	
14	**DB7**	
15	**LED+ A**	Resistencia de 470 Ω a 5 V para la iluminación del fondo de la pantalla
16	**LED- K**	0 V

Pantalla LCD 16x2_2

Así pues, podemos conectar la LCD así:

Podemos controlar la luminosidad de la pantalla (pin A) y del texto (pin V0) si los conectamos a salidas analógicas de Arduino.



Pantalla LCD 16x2_3 #include <LiquidCristal.h>

Veamos cómo controlar nuestro dispositivo LCD a través de Arduino:

En primer lugar, debemos abrir la librería LiquidCrystal.h:

```
#include <LiquidCristal.h>
```

Luego debemos declarar, como si de una variable se tratara, nuestra pantalla LCD, y debemos asignarle un nombre. Además debemos especificar los pines que vamos a usar para comunicar Arduino con la LCD:

```
LiquidCrystal nombreLcd(RS, E, D4, D5, D6, D7);
```

Dentro del `void setup()` debemos iniciar la librería para nuestra LCD, utilizando la función `begin()` y especificando el tamaño, que en nuestro caso es de 16x2 caracteres:

```
nombreLcd.begin(16, 2);
```

Pantalla LCD 16x2_4

Las funciones para controlar nuestra LCD son:

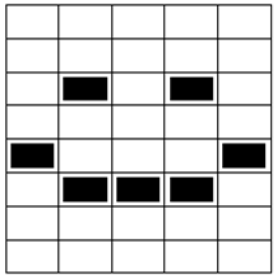
```
nombreLcd.setCursor(0, 7);   /*Sitúa el cursor de nuestra pantalla de 16x2 en
                              la primera línea, en el sexto carácter */

nombreLcd.print("Hola!");    /*Imprime en la posición fijada con anterioridad lo
                              que esté entre comillas, y sitúa el cursor tras el
                              último caracter. Actúa de manera análoga al print
                              en Serial.print(); Por ejemplo: lcd.print(val, DEC);
                              imprime el valor val en base 10 */

nombreLcd.clear();           /*Borra todo lo que hubiera en la pantalla hasta ese
                              momento y sitúa el cursor en la posición (0,0) */

nombreLcd.write(symbol[i]); /*Se emplea para mandar un caracter a la LCD. Puedo
                              mandar el caracter '+' (cuyo código ASCII es el 43)
                              de varias formas: */
        byte mas = B00101011;        //la B es para especificar que está en binario
        nombeLcd.write(mas);         //lo puede mandar como variable
        nombreLcd.write(B00101011); //lo puede mandar en código binario
        nombreLcd.write(43);         //lo puede mandar en base decimal
        nombreLcd.write(0x2B);       //lo puede mandar en base hexadecimal (0x)
```

Pantalla LCD 16x2_5

```
nombreLcd.createChar(numero, dato);
```

/*Esta función permite imprimir un símbolo creado por nosotros mismos, puesto que cada caracter posee una matriz de 5x8 píxeles. Para ello debemos generar dicho símbolo a través de 8 números binarios de 5 bits, correspondientes a cada una de las filas que componen dicho símbolo. Nuestro display LCD puede almacenar 8 símbolos diseñados por nosotros, y debemos indicar dónde lo almacena a través de dicho número (de 0 a 7).
Se suele incluir dentro del void setup, y luego, para imprimir dicho símbolo en la pantalla utilizamos la función nombreLcd.write(número). Veamos un ejemplo, imprimiremos una carita sonriendo: */

```
byte sonrisa[] = { B00000,
                   B00000,
                   B01010,
                   B00000,
                   B10001,
                   B01110,
                   B00000,
                   B00000};
void setup() {
  nombreLcd.createChar(0, sonrisa);   //será mi caracter 0
}

void loop() {
  nombreLcd.write(0);
}
```



Pantalla LCD 16x2_6

```
nombreLcd.scrollDisplayRight();

nombreLcd.scrollDisplayLeft();

        /*Estas funciones permiten desplazar lo que esté en la pantalla una posición hacia la derecha o a la izquierda (Right o Left). El máximo de caracteres (de longitud de una frase) que es posible almacenar con Arduino para luego ir desplazándola a través de una línea de la pantalla LCD es de 40. A partir de ahí, imprimirá en la segunda línea de textos (posición (0,1)) el caracter 41 de la frase */
```

Ejemplo: Pez nadando en su pecera

Consistirá en mostrar en nuestra pantalla LCD un pez que de desplaza por la pantalla, y va haciendo pequeñas pompitas. El programa puede ser el siguiente:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7); //establezco los pines para RS,E,D4,D5,D6,D7

byte pescadoIzquierda[] = {B00000,B00000,B00000,B00000,B01101,B11110,B01101,B00000};
byte pescadoDerecha[] = {B00000,B00000,B00000,B00000,B10110,B01111,B10110,B00000};
byte pescadoCentro[] = {B00000,B00000,B00000,B00000,B00100,B01110,B00100,B00000};
byte burbuja1[] = {B00010,B00000,B00100,B00010,B00100,B01110,B00100,B00000};
byte burbuja2[] = {B00000,B00100,B00010,B00000,B00100,B01110,B00100,B00000};
byte burbuja3[] = {B00100,B00000,B00000,B00000,B00100,B01110,B00100,B00000};
byte x = 0;                  //x e y son números muy pequeños,…
byte y = 0;                  //…así que los defino como byte para optimizar memoria
int tiempo = 600;

void setup() {
  lcd.begin(16,2);
  lcd.createChar(0, burbuja1);       //creo los caracteres de los peces
  lcd.createChar(1, burbuja2);
  lcd.createChar(2, burbuja3);
  lcd.createChar(3, pescadoIzquierda);
  lcd.createChar(4, pescadoDerecha);
  lcd.createChar(5, pescadoCentro);
}
```

Ejemplo: Pez nadando en su pecera_2

```
void loop() {
  desplazarDerecha(9);      //el pez nadará hacia la derecha 10 posiciones
  pararCentro();            //se parará mirando al frente
  pompas();                 //respirará echando pompas
  y = 1;                    //bajará a la fila de abajo
  desplazarIzquierda(5);    //ahora nadará hacia la izquierda 6 posiciones
  pararCentro();            //se parará otra vez
  pompas();                 //hará pompas otra vez
  y = 0;                    //subirá a la fila de arriba
  desplazarDerecha(11);     //nadará hacia la derecha 12 posiciones y se perderá
  delay(tiempo*10);         //tras un tiempo considerable...
  x = 0;
  y = 0;                    //...aparecerá de nuevo en la posición 0,0
}

void desplazarDerecha(int posiciones) {
  lcd.setCursor(x, y);
  lcd.write(4);             //es el pescado hacia la derecha
  delay(tiempo);
  for(int i=0; i<posiciones; i++) {
    lcd.scrollDisplayRight();
    delay(tiempo);
    x++;
  }
  lcd.clear();
}
```

Ejemplo: Pez nadando en su pecera_3

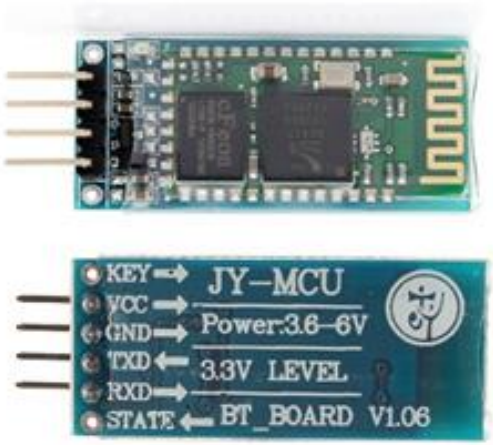
```
void desplazarIzquierda(int posiciones) {
  lcd.setCursor(x, y);
  lcd.write(3);
  delay(tiempo);
  for(int i=0; i<posiciones; i++) {
    lcd.scrollDisplayLeft();
    delay(tiempo);
    x--;
  }
  lcd.clear();
}

void pararCentro() {
  lcd.setCursor(x, y);
  lcd.write(5);
  delay(tiempo);
  lcd.clear();
}

void pompas() {
  for(int i=0; i<3; i++) {
    lcd.setCursor(x, y);
    lcd.write(i);
    delay(tiempo);
  }
  lcd.clear();
}
```

Módulo Bluetooth

Podemos establecer comunicación entre nuestra Arduino y otro dispositivo que disponga de comunicación Bluetooth (un móvil, por ejemplo), y de esta manera controlar nuestros actuadores (luces, motores, sonido, etc…) a través de él.

Para ello es necesario conectar un módulo Bluetooth, como puede ser el HC-06 (solo puede actuar como esclavo) o el HC-05 (puede actuar como maestro o esclavo):

HC-06



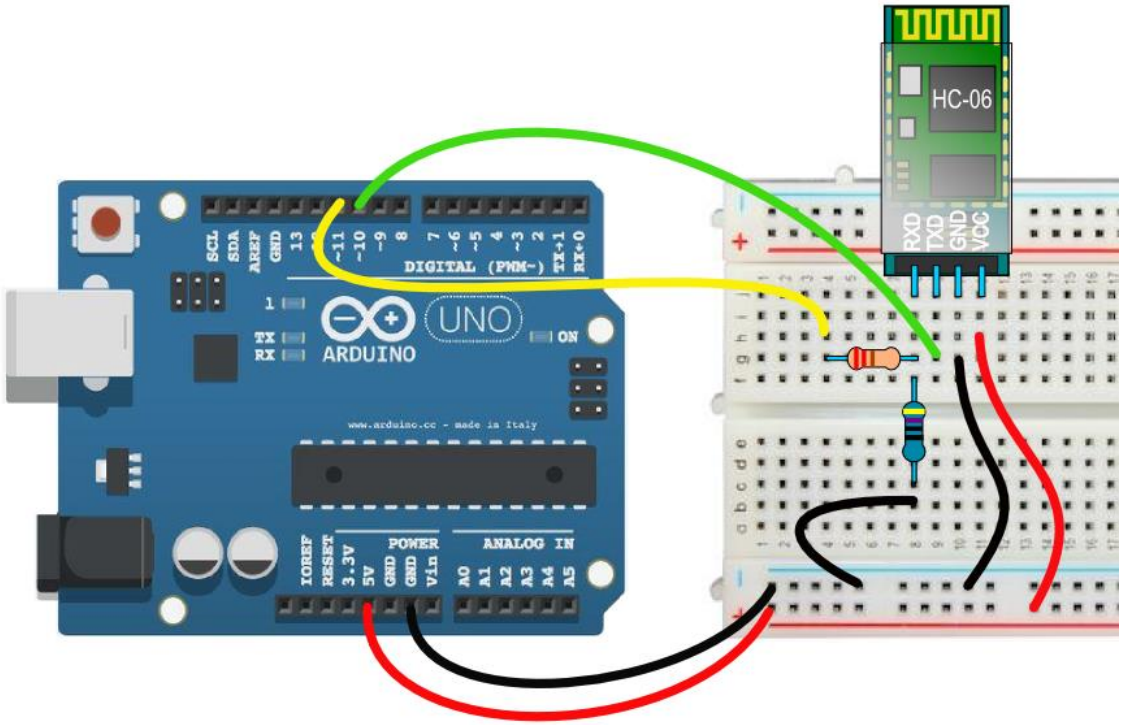
HC-05

VCC: a 5V.
GND: a 0V.
TXD: transmisión de datos.
RXD: recepción de datos (a 3,3V, y habrá que hacer un divisor de tensión).
KEY: poner a nivel alto para entrar en modo configuración.
STATE: para conectar un LED para visualizar la comunicación de datos.

Módulo Bluetooth_2 `#include <SoftwareSerial.h>`

Para evitar problemas de comunicación entre el PC (a través de los pines 0 y 1), vamos a utilizar la librería `SoftwareSerial` para poder comunicarme a través de otros pines (por ejemplo utilizaremos para recibir datos: RX→10, que lo conectaría al TXD del módulo, por donde transmite el Bluetooth, y para transmitir: TX→11 , que se conectaría al RXD del módulo, por donde recibe).

Arduino		Bluetooth
RX	←	TXD
TX	→	RXD

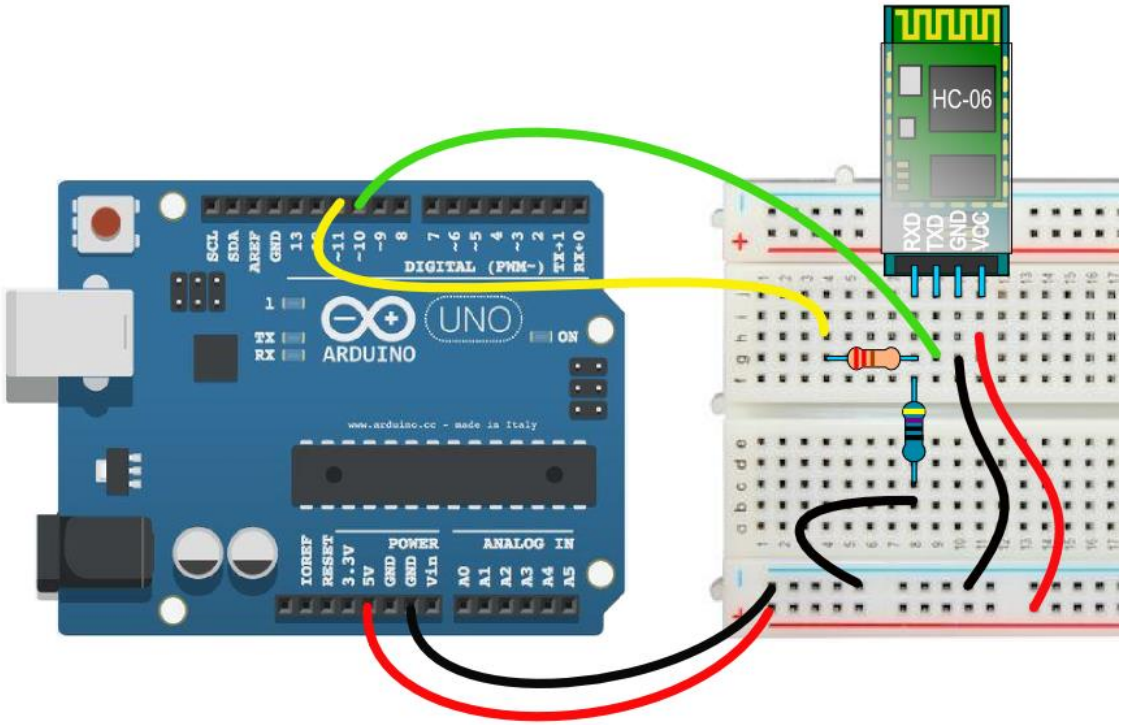


RXD: para conseguir señales de 3,3V en RXD, utilizaré una resistencia de 220 Ω y otra de 470 Ω (así obtendré 3,4V, que nos vale).

Módulo Bluetooth_2 #include <SoftwareSerial.h>

Realmente no es estrictamente necesario utilizar esta librería: podríamos haber utilizado los pines 0 y 1, que son (RX y TX respectivamente). El problema es que debo desconectar el módulo Bluetooth cuando vaya a cargar el sketch del PC a la Arduino, pues para dicho proceso necesitan estar libres dichos pines.

Arduino		Bluetooth
RX	←	TXD
TX	→	RXD



RXD: para conseguir señales de 3,3V en RXD, utilizaré una resistencia de 220 Ω y otra de 470 Ω (así obtendré 3,4V, que nos vale).

Comandos AT – Configurar el módulo Bluetooth

Los comandos AT sirven para configurar el módulo de Bluetooth. Se enviarán a través del Monitor Serie, pero teniendo seleccionada la opción “sin ajuste de línea”:

Pero antes deberemos meter el siguiente código:

```
#include <SoftwareSerial.h> //permite establecer comunicación serie en otros pins

SoftwareSerial BT(10,11);   //declaro el Bluetooth (al que llamo “BT”), e indico los
                            //pines de comunicación de Arduino: 10→RX y  11→TX

void setup() {
  BT.begin(9600);           //establezco comunicación con el módulo Bluetooth
  Serial.begin(9600);       //establezco comunicación con el PC
}

void loop() {
  if(BT.available()) {              //si hay datos disponibles con el módulo BT…
    Serial.write(BT.read());        //… los imprimo en el Monitor Serie
  }
  if(Serial.available()) {          //si hay datos disponibles con el PC…
     BT.write(Serial.read());       //… los mando al BT
  }
}
```

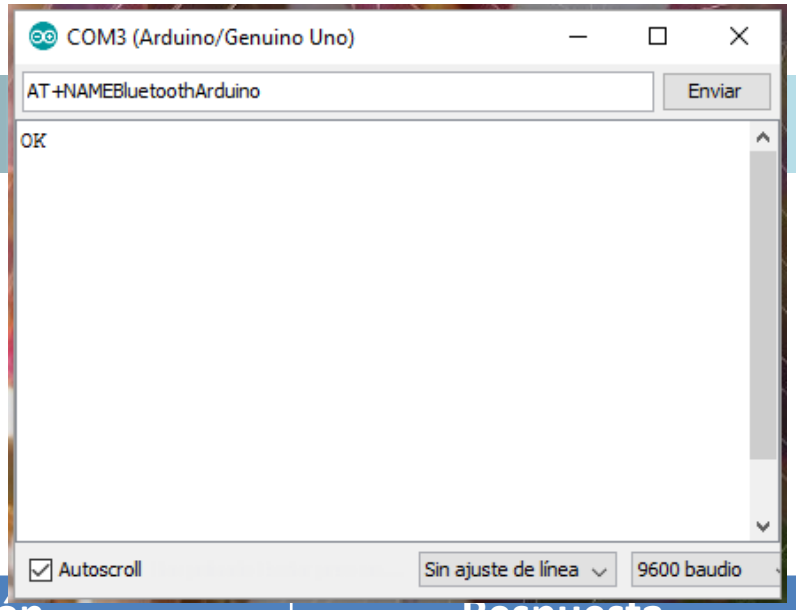
Comandos AT_2

Los comandos AT son los siguientes:

Comando AT	Descripción	Respuesta
AT	Test de comunicación	OK
AT+VERSION	Retorna la versión del módulo	OKlinvorV1.8
AT+BAUD*x*	Modifica los baudios de la comunicación, siendo *x* un múltiplo de 1200 bps (por ejemplo: AT+BAUD4 establecerá una velocidad de comunicación de 9600 bps, que es la opción por defecto) X = 1, 2,…, 9, A, B, C	OK9600
AT+NAME*x*	Modifica el nombre del módulo	OK*x*
AT+PIN*xxxx*	Modifica el pin de acceso	OK*xxxx*

Comandos AT_2

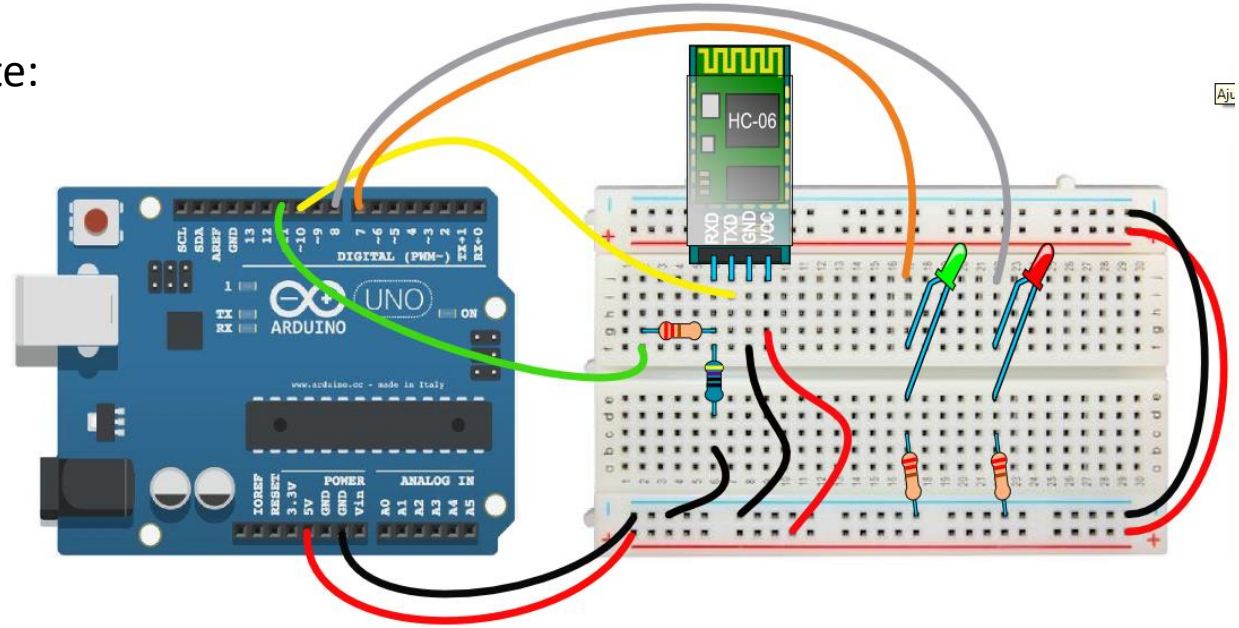
Los comandos AT son los siguientes:



Comando AT	Descripción	Respuesta
AT	Test de comunicación	OK
AT+VERSION	Retorna la versión del módulo	OKlinvorV1.8
AT+BAUD*x*	Modifica los baudios de la comunicación, siendo *x* un múltiplo de 1200 bps (por ejemplo: AT+BAUD4 establecerá una velocidad de comunicación de 9600 bps, que es la opción por defecto) X = 1, 2,…, 9, A, B, C	OK9600
AT+NAME*x*	Modifica el nombre del módulo	OK*x*
AT+PIN*xxxx*	Modifica el pin de acceso	OK*xxxx*

Ejemplo: Control de 2 LEDs desde un móvil

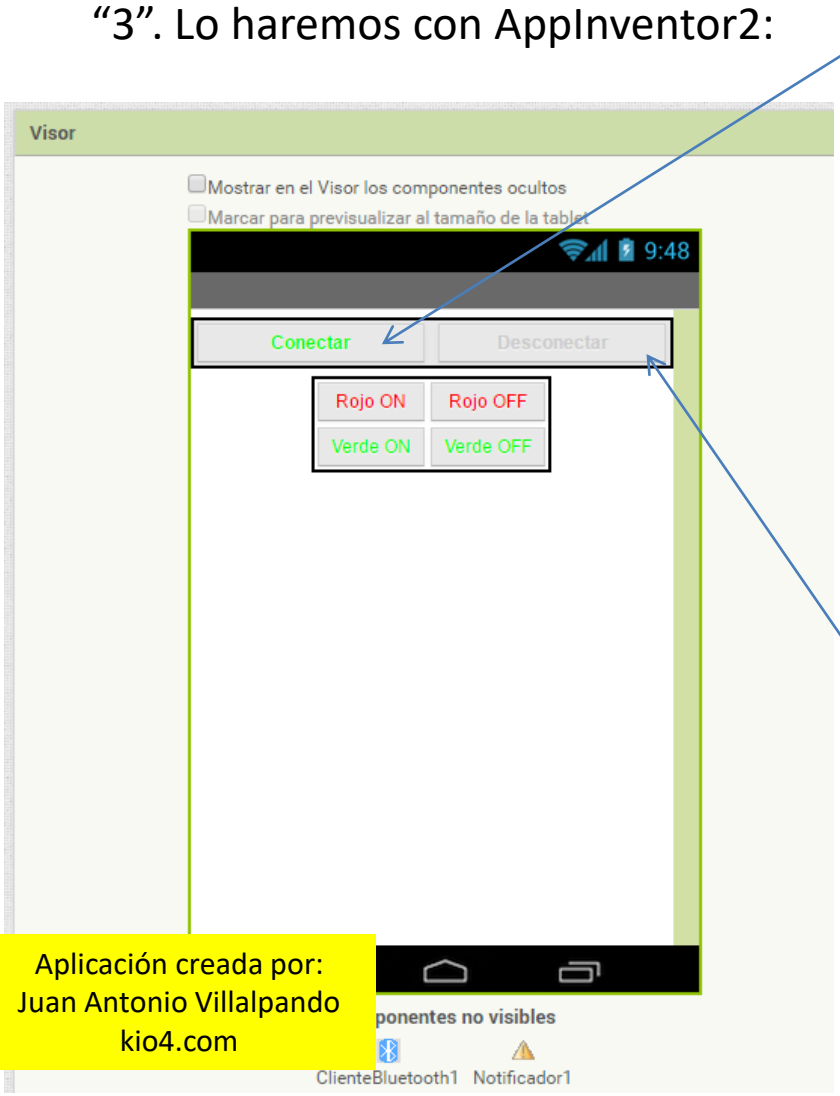
El montaje será el siguiente:

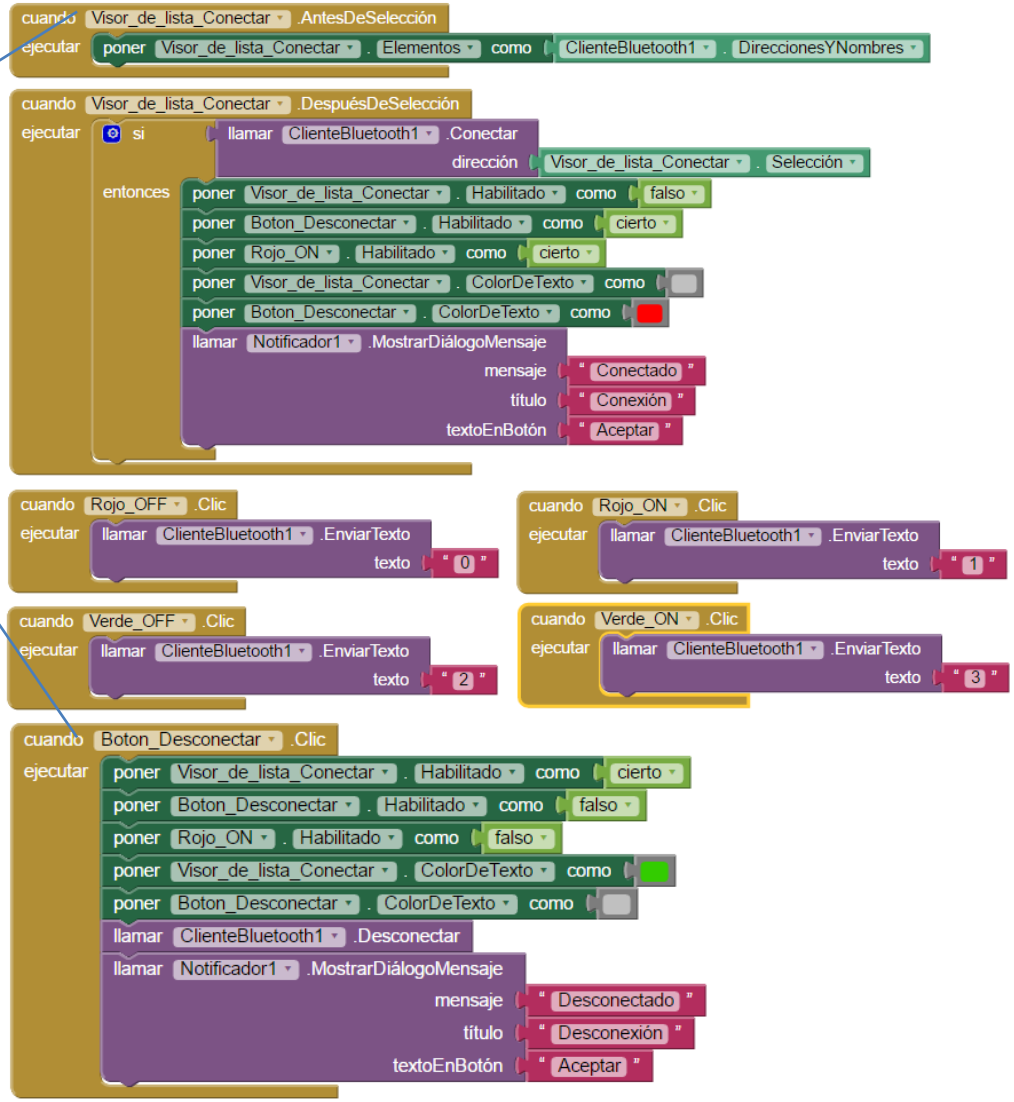



```
#include <SoftwareSerial.h>

SoftwareSerial BT(10, 11);

int val;
int ledRojo = 8;
int ledVerde = 7;

void setup() {
  BT.begin(9600);
  pinMode(ledRojo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
}

void loop() {
  if(BT.available()) val = BT.read();
  if(val == '0') digitalWrite(ledRojo, LOW);
  if(val == '1') digitalWrite(ledRojo, HIGH);
  if(val == '2') digitalWrite(ledVerde, LOW);
  if(val == '3') digitalWrite(ledVerde, HIGH);
}
```

Ejemplo: Control de 2 LEDs desde un móvil_2

Ahora solo falta subir a nuestro móvil una aplicación que mande “0”, “1”, “2” y “3”. Lo haremos con AppInventor2:

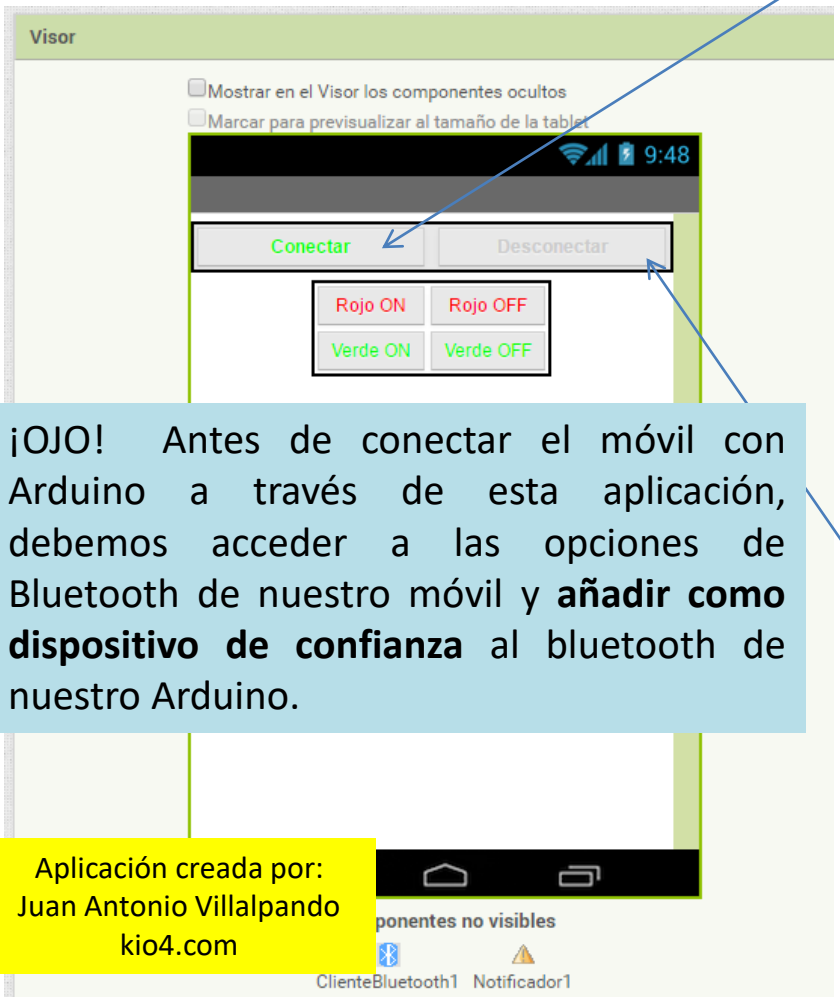




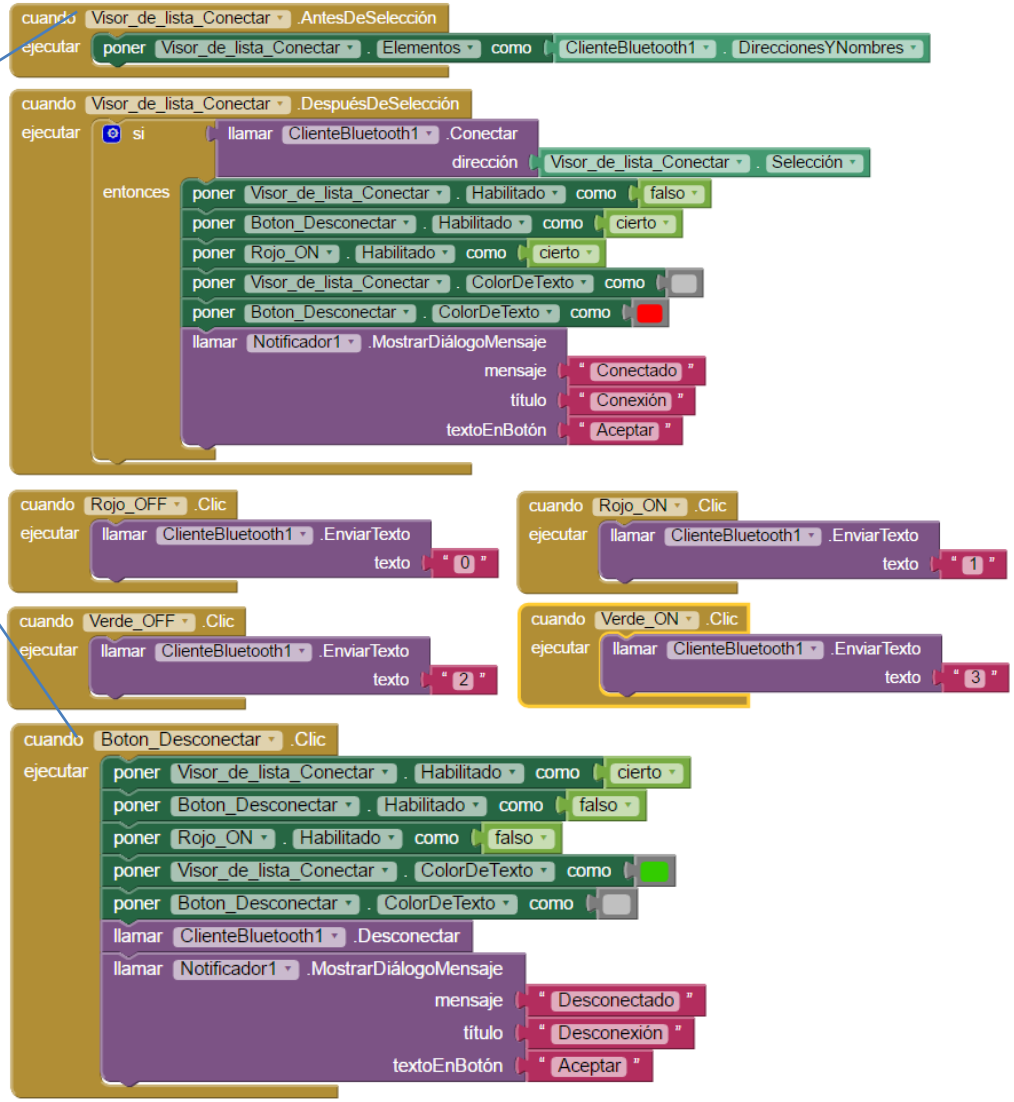
Aplicación creada por:
Juan Antonio Villalpando
kio4.com

Ejemplo: Control de 2 LEDs desde un móvil_2

Ahora solo falta subir a nuestro móvil una aplicación que mande “0”, “1”, “2” y “3”. Lo haremos con AppInventor2:



¡OJO! Antes de conectar el móvil con Arduino a través de esta aplicación, debemos acceder a las opciones de Bluetooth de nuestro móvil y **añadir como dispositivo de confianza** al bluetooth de nuestro Arduino.

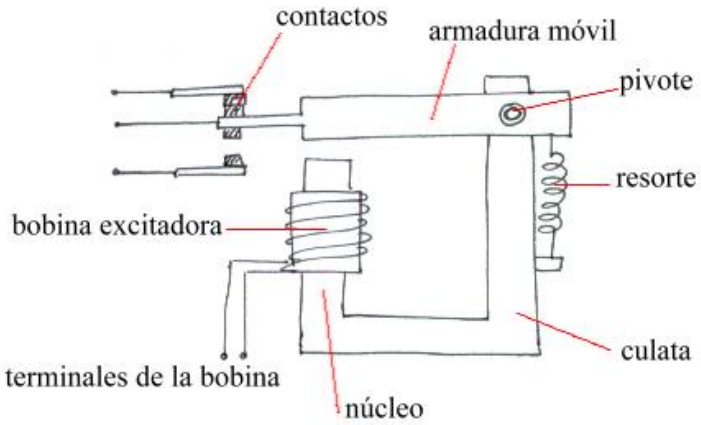


Aplicación creada por:
Juan Antonio Villalpando
kio4.com

Relé

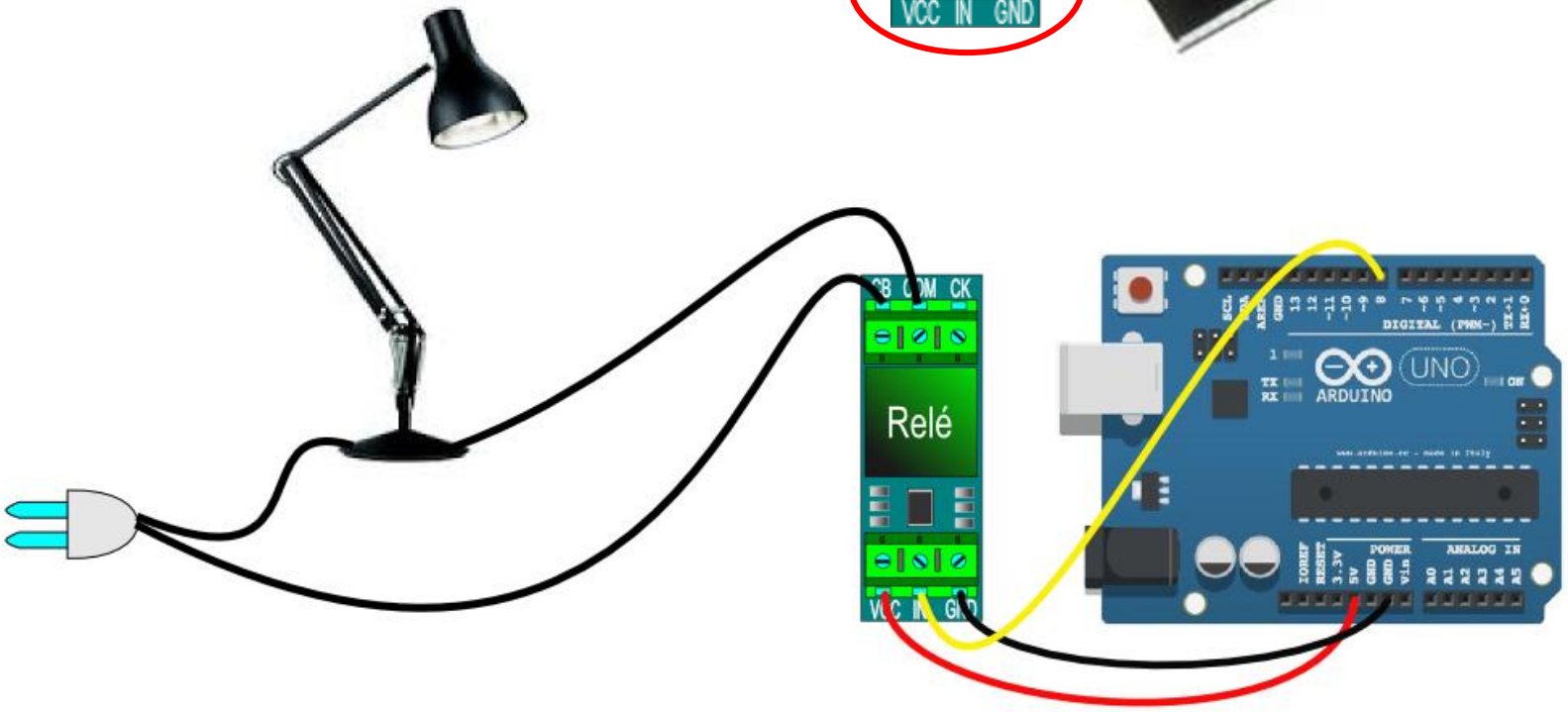
Arduino solo puede manejar pequeñas intensidades (40 mmA en los pines de salida) y pequeños voltios (5 V). ¿Puede Arduino controlar algún aparato eléctrico, como una lámpara, un ventilador, etc…?

La respuesta es que sí: con ayuda de un **relé**. Un relé es un dispositivo que permite controlar con una pequeña corriente (por los terminales de la bobina) la apertura o cierre de un circuito por el que puede pasar una corriente mucho más elevada.



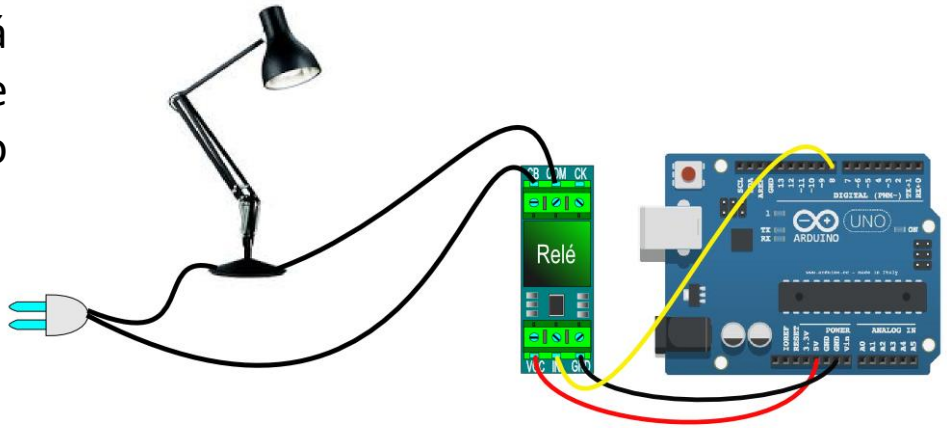
Relé_2

Así pues, los terminales de control o mando se conectarán a la Arduino, y los terminales de fuerza o potencia al circuito de mayor tensión e intensidad que queremos hacer funcionar:



Ejemplo: Encendido de un flexo con el teclado

Utilizaremos el mismo montaje de la diapositiva anterior. El flexo encenderá cuando apretemos la tecla “e” (de encender), y se apagará cuando apretemos la tecla “a” (de apagar).




```
int tecla;

void setup() {
  Serial.begin(9600);
  pinMode(8, OUTPUT);        //controlará al relé
}

void loop() {
  tecla = Serial.read();     /* tecla será el último caracter que se introduzca en el
                             Monitor Serie */
  if(tecla == 'e') digitalWrite(8, HIGH);      //activo el electroimán, y hago mover…
                                               //… los contactos, para encender la luz
  if(tecla == 'a') digitalWrite(8, LOW);       //desactivo el electroimán, y el muelle
                                               //hace cambiar a los contactos.
}
```

Reducir el número de salidas

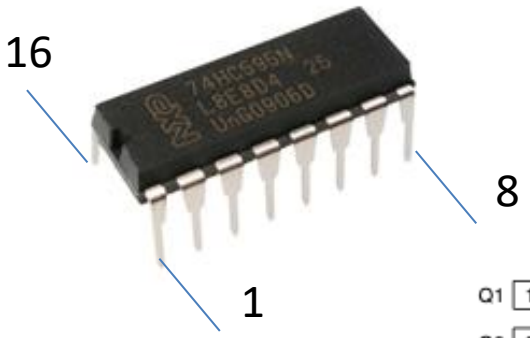
En este capítulo hemos visto cómo la utilización de un simple display de 4 dígitos de 7-segmentos o una pantalla LCD acaparaba una cantidad muy grande de salidas de nuestro Arduino (12 pines y 6 pines respectivamente), lo cual es un grave problema en el caso de que queramos construir proyectos que requieran de varias salidas y entradas (por ejemplo, sería imposible gobernar una matriz de 8x8 leds bicolor Arduino UNO, ya que requeriría de 24 pines).

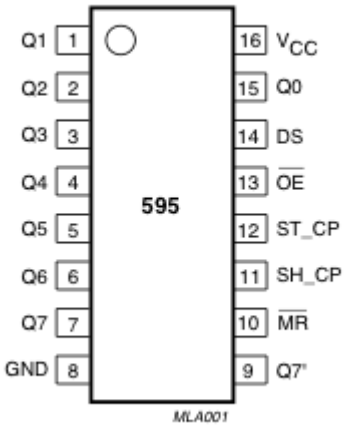
Cabe la posibilidad de utilizar un registro de desplazamiento (Shift Register) como el integrado **74HC595** (es un registro de desplazamiento de 8 bits con entrada serie y salida paralelo), o algunos integrados diseñados específicamente para gobernar dispositivos como un display de 7-segmentos (integrado **decodificador 7447**) o una pantalla LCD 16X2 (**módulo I2C**).

Veamos ahora esta forma de ahorrar pines de nuestro Arduino.

74HC595 Shift Register

El integrado 74HC595 permite mandar simultáneamente (en paralelo) 8 bits diferentes como salida, habiéndose introducido estos por un solo cable (en serie) ayudándose de una señal de reloj y una señal de latch.

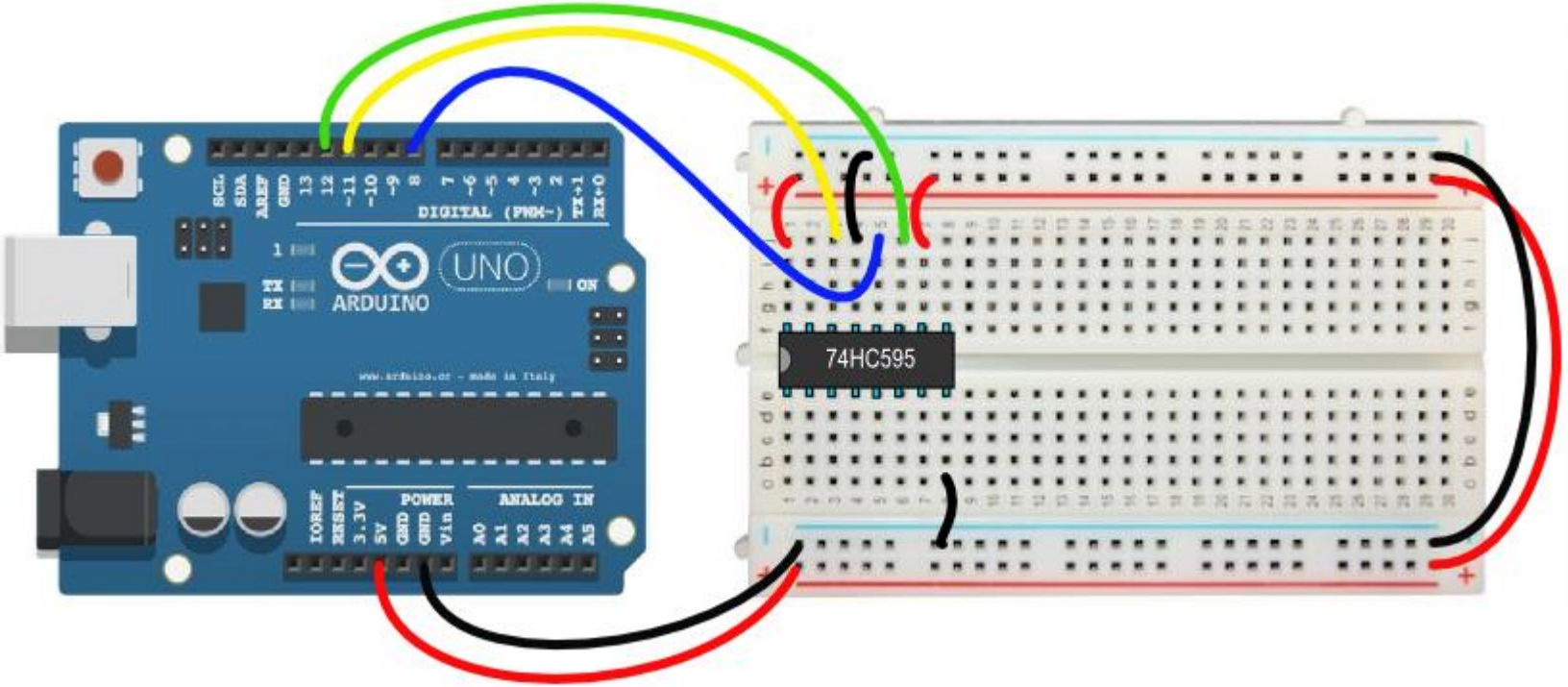




Pines del 74HC595		
1	**Q1**	Salidas
2	**Q2**	
3	**Q3**	
4	**Q4**	
5	**Q5**	
6	**Q6**	
7	**Q7**	
8	**GND**	0 V
9	**Q7'**	Para gobernar otro 74HC595 (que sería su DS)
10	**MR**	Master Reclear (se activa en LOW)
11	**SH_CP**	Señal de reloj
12	**ST_CP**	Señal de latch (actualiza las salidas cuando pasa de LOW a HIGH (flanco de subida))
13	**OE**	Output Enable (se activa en LOW)
14	**DS**	Data Serial (entradas en serie)
15	**Q0**	Salida
16	**VCC**	0 V

74HC595 Shift Register_2

¿Cómo se conecta este integrado? Vemos que nuestra Arduino solo necesitará de 3 salidas para controlarlo: la señal de reloj (**SH_CP**), la señal de latch(**ST_CP**) y los datos serie (**DS**). En este ejemplo: reloj → pin12; latch → pin8; datos → pin11



Posteriormente conectaríamos las salidas (Q0, …, Q7) al dispositivo o dispositivos que quisiéramos controlar.

(Se recomienda incorporar un pequeño condensador (1 µF) entre **GND** y **ST_CP** para estabilizar la señal de latch)

74HC595 Shift Register_3

¿Cómo funciona este integrado? En primer lugar, deberé almacenar en una variable tipo `byte` la combinación de bits que quiero poner como salida:

```
byte bitsSalida = B00110011;        //cada uno de esos bits será un valor de salida

/*el byte también se puede poner como combinación de dos dígitos en base hexadecimal (se indica comenzando por 0X):
  B00000000 = 0X00    B00011111 = 0X1F    B01001010 = 0X8A    B11111111 = 0XFF */
```

En segundo lugar, deberé volcar dicha combinación de bits a la memoria del chip. Esto se realizará a través de los pines **SH_CP** y **DS** mediante la función `shiftOut`:

```
shiftOut(pinDS, pinSH_CP, MSBFIRST, bitsSalida);
```

Debemos especificar los pines de DS y del reloj, así como el byte del que debe copiar los datos. El tercer parámetro presenta dos posibilidades, dependiendo de en qué orden queremos que vaya introduciendo los bits de `B00110011`:

`MSBFIRST`: Most Significant Bit First `//así sería Q7=0, Q6=0, Q5=1, … Q0=1`

`LSBFIRST`: Least Significant Bit First `//así sería Q7=1, Q6=1, Q5=0, … Q0=0`

74HC595 Shift Register_4

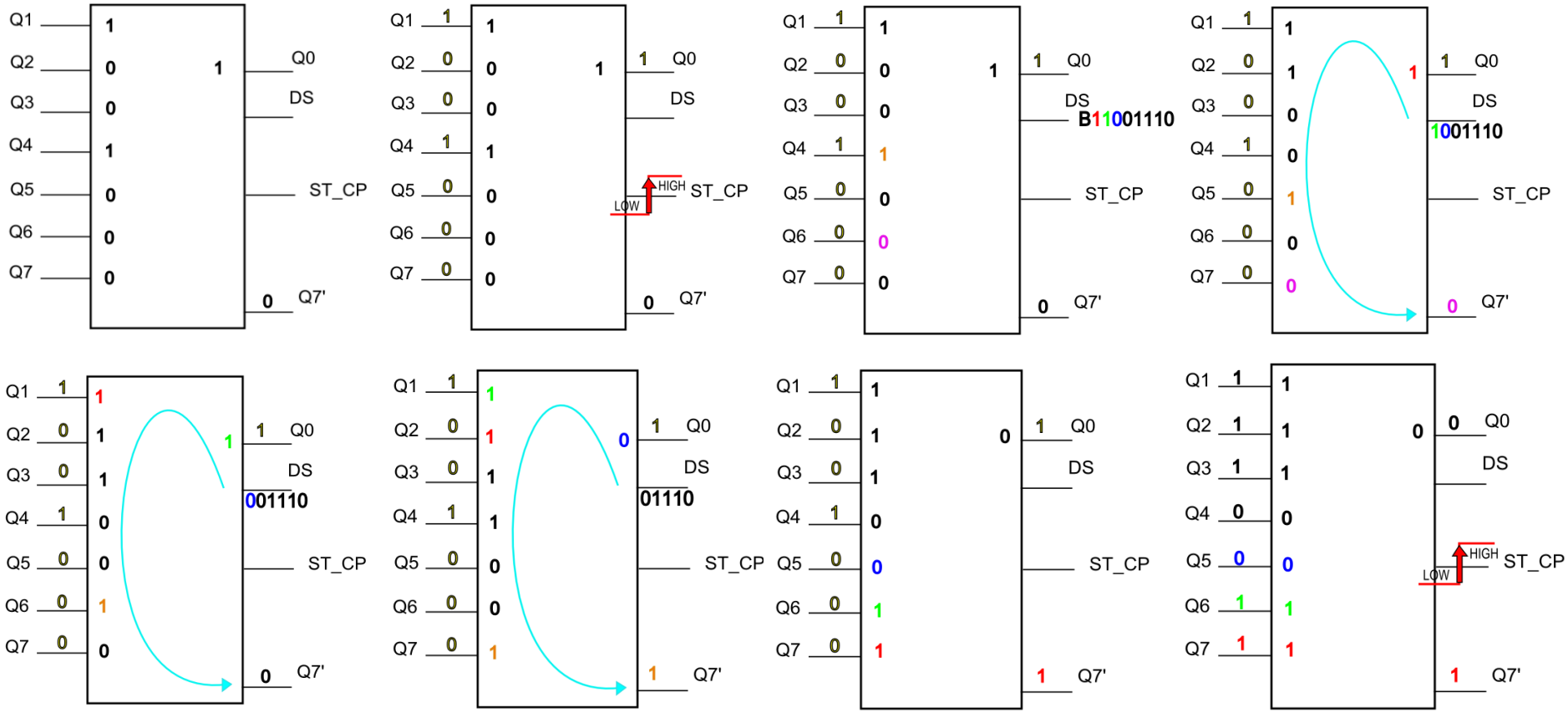
Esta carga de memoria debe producirse con la señal de latch en LOW, y la actualización en los valores de salida no se producirá hasta que la señal de latch no vuelva al estado HIGH:

```
byte bitsSalida = B00110011;

digitalWrite(pinST_CP, LOW);
shiftOut(pinDS, pinSH_CP, MSBFIRST, bitsSalida);
digitalWrite(pinST_CP, HIGH);

//las salidas se hacen efectivas en el flanco de subida de la señal de latch
```

Veamos de una forma más gráfica cómo se realiza el proceso de registro por desplazamiento:

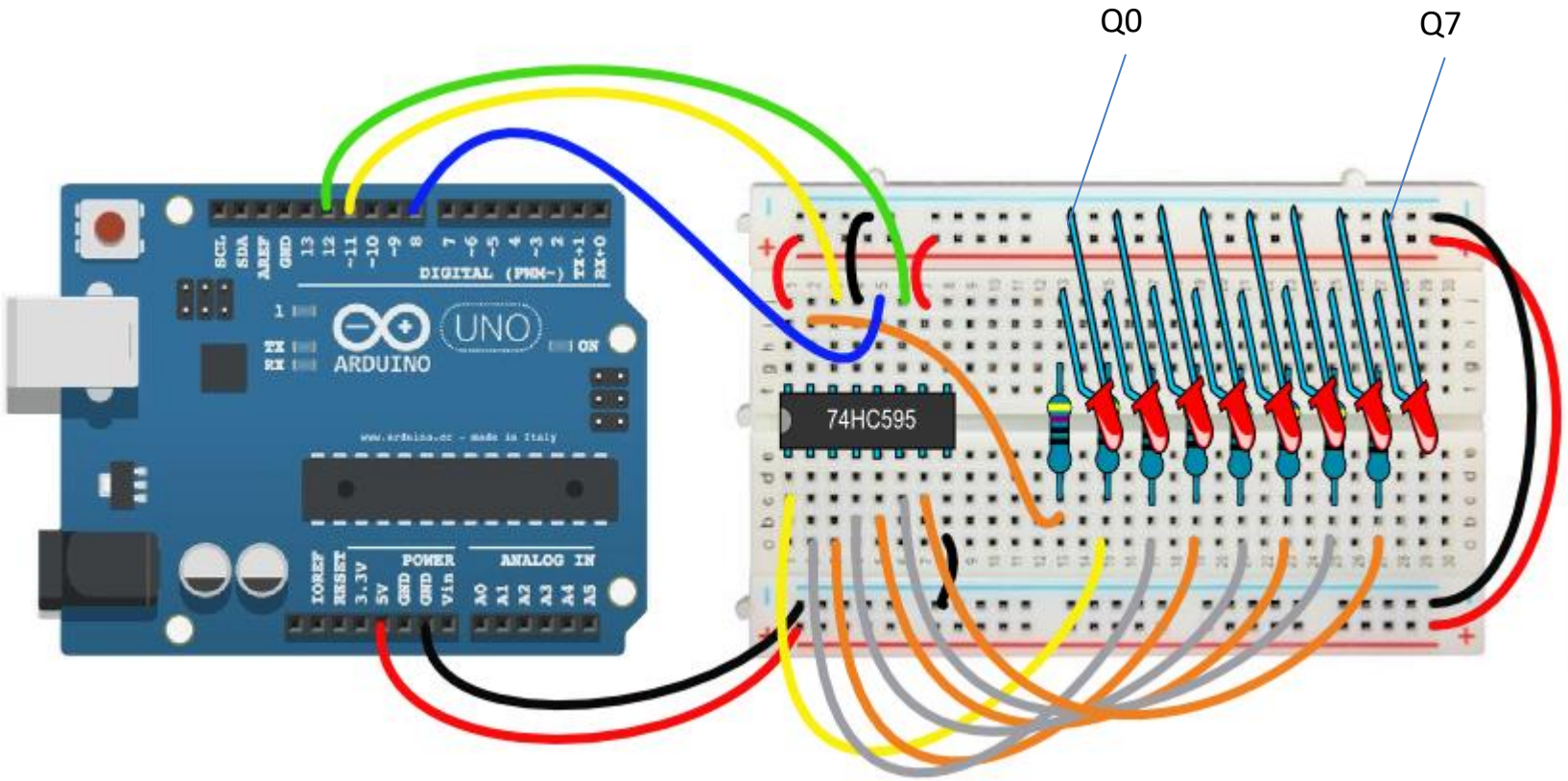
74HC595 Shift Register_5

A cada flanco de subida en el latch, se actualizan los valores de las salidas. Si luego introducimos el `byte B11001110` (con `shiftOut`), el procedimiento es “ir empujando” como si de una fila se tratara, bit a bit, hasta que los nuevos 8 bits ocupen las posiciones nuevas (en el ejemplo lo hacen en el orden MSBFIRST). Al siguiente flanco de subida, se actualizarán los valores de las salidas. Vemos que Q7’ va tomando los valores de los diferentes bits que van pasando por ahí, sin necesidad de “actualizar” las salidas.



Ejemplo: Encendido progresivo de 8 LEDs

El montaje será el siguiente:



Ejemplo: Encendido progresivo de 8 LEDs_2

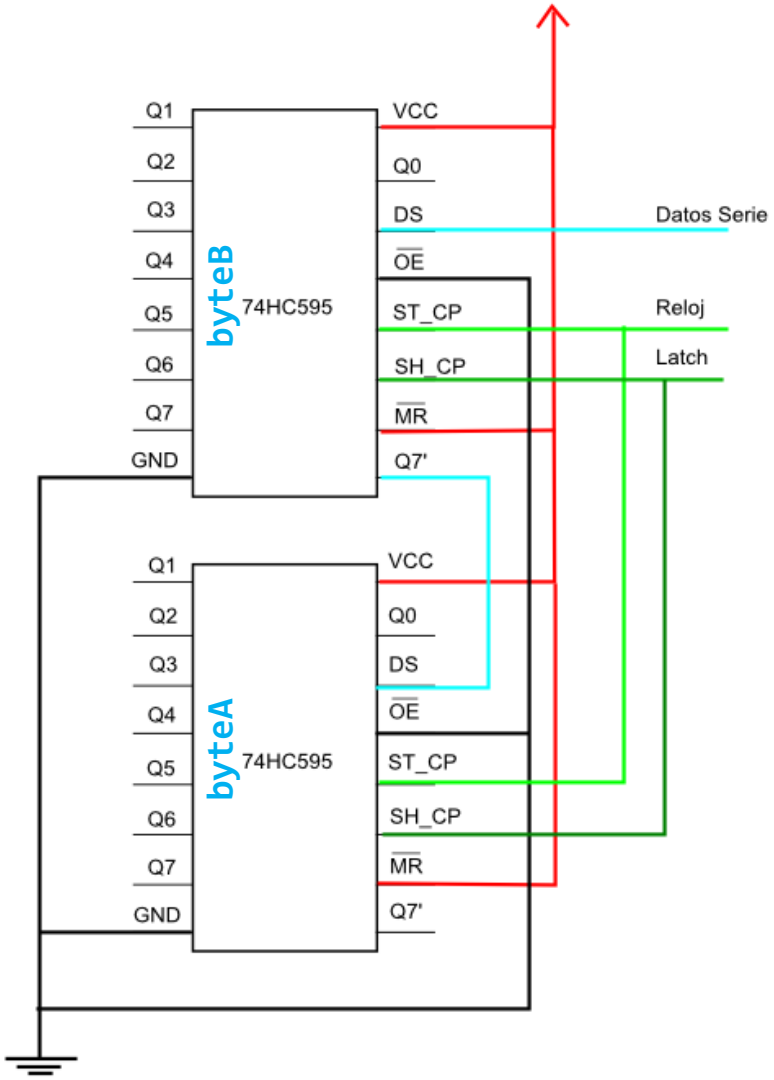
```
int latchPin = 8;   //conecto el latch (ST_CP)al pin 8
int clockPin = 12;  //conecto el reloj (SH_CP) al pin 12
int dataPin = 11;   //conecto los datos (DS) al pin 11
byte salidasLuz[] {
  B00000000,          //todos los LEDs apagados
  B00000001,          //solo enciende el LED conectado a la salida Q0
  B00000011,          //encienden Q0 y Q1
  B00000111,          //etc...
  B00001111,
  B00011111,
  B00111111,
  B01111111,
  B11111111 };

void setup() {
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop() {
  for(int i=0; i<9; i++) {
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, salidasLuz[i]);
    digitalWrite(latchPin, HIGH);
    delay(1000);      //a cada segundo cambian las salidas
  }
}
```

Concatenando varios 74HC595 Shift Register

¿Es 8 el máximo de salidas digitales que puedo controlar con nuestro 74HC595? Con un solo registro de desplazamiento, sí. Pero si concateno un segundo registro, de tal manera que la salida Q7' del primero se conecta a la entrada DS del segundo… ¡podría controlar 8 + 8 = 16 salidas con solo 3 pines de Arduino!

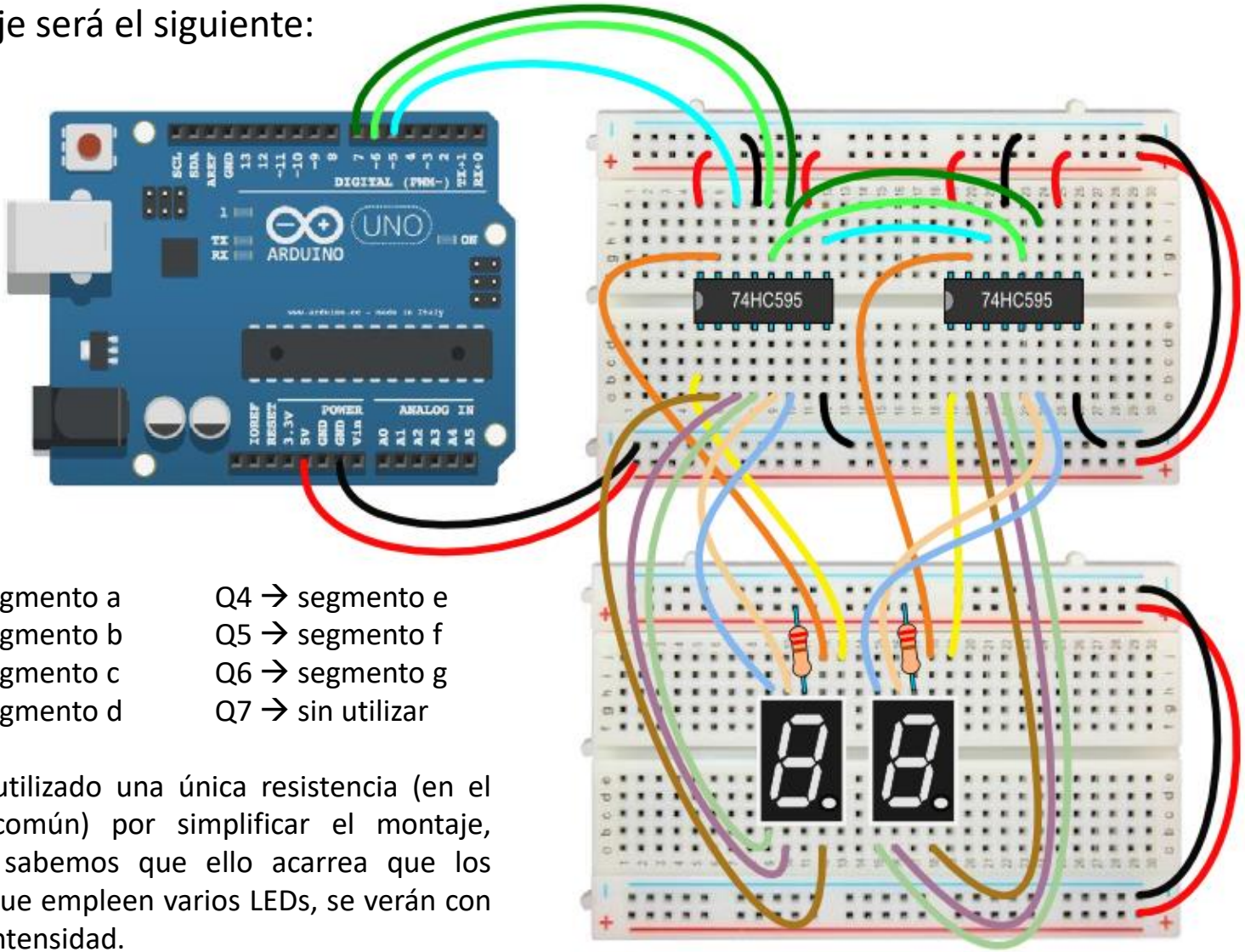
Para ello deberé volcar a través de DS en primer lugar el byte correspondiente a los estados de las 8 salidas del segundo registro, y luego volcar el byte correspondiente a los estados de las 8 salidas del primer registro.

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



Ejemplo: Crono: gobierno de 2 displays 7-segmentos

El montaje será el siguiente:



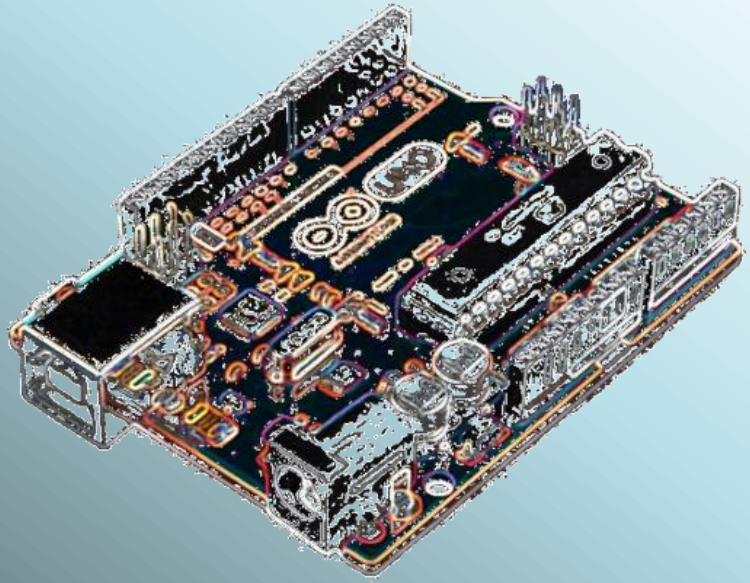
Q0 → segmento a
Q1 → segmento b
Q2 → segmento c
Q3 → segmento d
Q4 → segmento e
Q5 → segmento f
Q6 → segmento g
Q7 → sin utilizar

Hemos utilizado una única resistencia (en el ánodo común) por simplificar el montaje, aunque sabemos que ello acarrea que los dígitos que empleen varios LEDs, se verán con menos intensidad.

Ejemplo: Crono: gobierno de 2 displays 7-segmentos_2

```
int latchPin = 6;  //conecto el latch (ST_CP)al pin 6
int clockPin = 7;  //conecto el reloj (SH_CP) al pin 7
int dataPin = 5;   //conecto los datos (DS) al pin 5
byte digitos[] = {    //almacenaré en 1 byte la combinación de los segmentos a,b,c,d,e y f
  B00000010,          //forma el número cero en lógica inversa
  B10011110,          //forma el número 1
  B00100101,          //forma el número 2
  B00001100,          //etc...
  B10011000,          //Como nuestro display es de ánodo común, cada segmento encenderá…
  B01001000,          //…cuando haya un 0 en el pin correspondiente, y un 1 lo apagará.
  B01000000,          //Es decir: cada byte significa Babcdefg0 (como el dp no se utiliza…
  B00011110,          //…, completaré el byte un el último 0).
  B00000000,          //Eso significa que deberé utilizar el modo LSBFIRST para cargar los…
  B00001000 };        //…bytes en el registro, para que Q0 sea a, Q1 sea b, Q2 sea c, etc…
int tiempo = 0;

void setup() {
  for(int i=5; i<8; i++) pinMode(i, OUTPUT);
}

void loop() {
  int d = tiempo/10;
  int u = tiempo - d*10;
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, digitos[u]);
  shiftOut(dataPin, clockPin, LSBFIRST, digitos[d]);
  digitalWrite(latchPin, HIGH);
  delay(1000);      //a cada segundo cambian las salidas
  tiempo++;
  if(tiempo >99) tiempo = 0;
}
```

FIN del Capítulo V



Daniel Gallardo García
Profesor de Tecnología
Jerez de la Frontera
danielprofedetecno@gmail.com