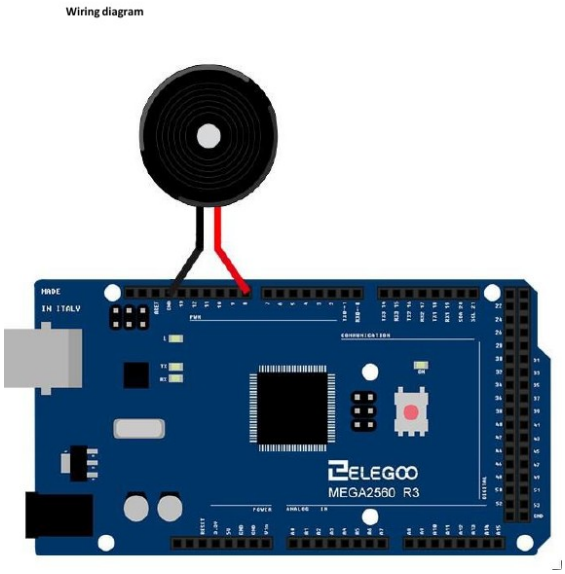
Práctica 5 Autor: Francisco Javier Bolívar Expósito

Zumbador



Código:

```
/************************** Declaración de Escalas ************************/
int c[8]= { 33, 65,  131, 262, 523, 1047, 2093, 4186 };  // Do
int cs[8]={ 35, 69,  139, 277, 554, 1109, 2217, 4435 };  // Do#
int d[8]= { 37, 73,  147, 294, 587, 1175, 2349, 4699 };  // Re
int ds[8]={ 39, 78,  156, 311, 622, 1245, 2489, 4978 };  // Re#
int e[8]= { 41, 82,  165, 330, 659, 1319, 2637, 0    };  // Mi
int f[8]= { 44, 87,  175, 349, 698, 1397, 2794, 0    };  // Fa
int fs[8]={ 46, 93,  185, 370, 740, 1480, 2960, 0    };  // Fa#
int g[8]= { 49, 98,  196, 392, 784, 1568, 3136, 0    };  // Sol
int gs[8]={ 52, 104, 208, 415, 831, 1661, 3322, 0    };  // Sol#
int a[8]= { 55, 110, 220, 440, 880, 1760, 3520, 0    };  // La
int as[8]={ 58, 117, 233, 466, 932, 1865, 3729, 0    };  // La#
int b[8]= { 62, 123, 247, 494, 988, 1976, 3951, 0    };  // Si

/***************************** TONOS ***************************************/
/* Función alarma, emite un número determinado de pitidos en un segundo,
separados por silencios de igual duración. */
void alarma(int spk, int pitidos);

/* Funcion auxiliar nota, genera un pitido de una frecuencia durante un
tiempo seguido de un silencio. */
void nota(int spk, int frecuencia, int duracion, int silencio);

/***************************** MELODIAS ************************************/
```

```
void URSS(int spk);     // Himno de la URSS

void setup()
{}

void loop()
{
  URSS(8);

 //alarma(3,10);
}

/****************************** Funcion Nota *********************************/
/* Función que emite un número determinado de pitidos en un segundo,       */
/* separando los pitidos por silencios de igual duración.                  */
/* Parámetros:                                                             */
/*  - spk: Pin sobre el que emite el sonido.                               */
/*  - pitidos: Número de pitidos que se desean emitir en un segundo, si    */
/*    toma el valor uno se emitirá un pitido de medio segundo seguido de un */
/*    silencio de igual duración, si toma el valor 10 se emitirán 10       */
/*    pitidos de 50 milisegundos y sus correspondientes silencios, si toma  */
/*    el valor cero se emitirá un pitido continuo de un segundo.           */
/*************************************************************************/

void alarma(int spk, int pitidos)
{
  int intervalo;
  int frecuencia=523;

  if (pitidos == 0) {
     tone(spk,frecuencia);
     delay(1000);
     noTone(spk);
  }
  else {
     intervalo=1000/(2*pitidos);
     for (int i=0; i<pitidos; i++)
     {
        tone(spk,frecuencia);
        delay(intervalo);
        noTone(spk);
        delay(intervalo);
     }
  }
}

/****************************** Funcion Nota *********************************/
/* Función que toca una nota indicando la frecuencia y la duración         */
/* Parámetros:                                                             */
```

```
/*  - spk: Pin sobre el que emite el sonido.                          */
/*  - frecuencia: Frecuencia de la nota, obtenida de los vectores de escala */
/*  - duracion: Tiempo, en milisegundos, durante el que se sostiene la nota */
/*  - silencio: Tiempo, en milisegundos, de silencio tras tocar la nota     */
/*****************************************************************************/

void nota(int spk, int frecuencia, int duracion, int silencio)
{
   tone(spk,frecuencia);   // suena la nota con una determinada frecuenca
   delay(duracion);        // durante un tiempo determinado
   noTone(spk);            // paramos el tono
   delay(silencio);        // mantenemos el silencio durante un tiempo
}

/************************ Funcion URSS *******************************/
/* Función que reproduce el himnno de la URSS                        */
/*********************************************************************/
void URSS (int spk)
{
 nota( spk, g[3], 395, 0   );
  nota( spk, c[4], 790, 0   );
  nota( spk, g[3], 593, 0   );
  nota( spk, a[3], 198, 0   );

  nota( spk, b[3], 790, 0   );
  nota( spk, e[3], 395, 0   );
  nota( spk, e[3], 395, 0   );
  nota( spk, a[3], 790, 0   );

  nota( spk, g[3], 593, 0   );
  nota( spk, f[3], 198, 0   );
  nota( spk, g[3], 790, 0   );
  nota( spk, c[3], 395, 0   );

  nota( spk, c[3], 395, 0   );
  nota( spk, d[3], 790, 0   );
  nota( spk, d[3], 395, 0   );
  nota( spk, e[3], 395, 0   );

  nota( spk, f[3], 790, 0   );
  nota( spk, f[3], 395, 0   );
  nota( spk, g[3], 395, 0   );
  nota( spk, a[3], 790, 0   );

  nota( spk, b[3], 390, 0   );
  nota( spk, c[4], 390, 0   );
  nota( spk, d[4], 1580, 0   );
  nota( spk, e[4], 790, 0   );
```

```
nota( spk, d[4], 593, 0   );
nota( spk, c[4], 198, 0   );
nota( spk, d[4], 790, 0   );
nota( spk, b[3], 395, 0   );

nota( spk, g[3], 395, 0   );
nota( spk, c[4], 790, 0   );
nota( spk, b[3], 593, 0   );
nota( spk, a[3], 198, 0   );

nota( spk, b[3], 790, 0   );
nota( spk, e[3], 395, 0   );
nota( spk, e[3], 395, 0   );
nota( spk, a[3], 790, 0   );

nota( spk, g[3], 593, 0   );
nota( spk, f[3], 198, 0   );
nota( spk, g[3], 790, 0   );
nota( spk, c[3], 593, 0   );

nota( spk, c[3], 198, 0   );
nota( spk, c[4], 790, 0   );
nota( spk, b[3], 593, 0   );
nota( spk, a[3], 198, 0   );

nota( spk, g[3], 395, 0   );
nota( spk, b[3], 395, 0   );
nota( spk, c[4], 395, 0   );
nota( spk, d[4], 395, 0   );

nota( spk, e[4], 1580, 0   );
nota( spk, d[4], 395, 0   );
nota( spk, c[4], 395, 0   );
nota( spk, b[3], 395, 0   );

nota( spk, c[4], 395, 0   );
nota( spk, d[4], 1185, 0   );
nota( spk, g[3], 395, 0   );
nota( spk, g[3], 395, 0   );

nota( spk, b[3], 395, 0   );
nota( spk, c[4], 395, 0   );
nota( spk, d[4], 395, 0   );
nota( spk, c[4], 1580, 0   );

nota( spk, b[3], 395, 0   );
nota( spk, a[3], 395, 0   );
nota( spk, g[3], 395, 0   );
nota( spk, a[3], 395, 0   );
```

```
nota( spk, b[3], 1185, 0   );
nota( spk, e[3], 395, 0   );
nota( spk, e[3], 395, 0   );
nota( spk, g[3], 395, 0   );

nota( spk, a[3], 395, 0   );
nota( spk, b[3], 395, 0   );
nota( spk, c[4], 790, 0   );
nota( spk, a[3], 593, 0   );

nota( spk, b[3], 198, 0   );
nota( spk, c[4], 790, 0   );
nota( spk, a[3], 593, 0   );
nota( spk, b[3], 198, 0   );

nota( spk, c[4], 790, 0   );
nota( spk, a[3], 395, 0   );
nota( spk, c[4], 395, 0   );
nota( spk, f[4], 1580, 0   );

nota( spk, f[4], 1580, 0   );
nota( spk, e[4], 395, 0   );
nota( spk, d[4], 395, 0   );
nota( spk, c[4], 395, 0   );

nota( spk, d[4], 395, 0   );
nota( spk, e[4], 1185, 0   );
nota( spk, c[4], 395, 0   );
nota( spk, c[4], 1580, 0   );

nota( spk, d[4], 1580, 0   );
nota( spk, c[4], 395, 0   );
nota( spk, b[3], 395, 0   );
nota( spk, a[3], 395, 0   );

nota( spk, b[3], 395, 0   );
nota( spk, c[4], 1185, 0   );
nota( spk, a[3], 395, 0   );
nota( spk, a[3], 1580, 0   );

nota( spk, c[4], 790, 0   );
nota( spk, b[3], 593, 0   );
nota( spk, a[3], 198, 0   );
nota( spk, g[3], 790, 0   );

nota( spk, c[3], 790, 0   );
nota( spk, c[4], 790, 0   );
nota( spk, b[3], 593, 0   );
```

```
nota( spk, a[3], 198, 0   );

nota( spk, g[3], 1185, 0   );
nota( spk, g[3], 395, 0   );
nota( spk, c[4], 790, 0   );
nota( spk, g[3], 593, 0   );

nota( spk, a[3], 198, 0   );
nota( spk, b[3], 790, 0   );
nota( spk, e[3], 395, 0   );
nota( spk, e[3], 395, 0   );

nota( spk, a[3], 790, 0   );
nota( spk, g[3], 593, 0   );
nota( spk, f[3], 198, 0   );
nota( spk, g[3], 790, 0   );

nota( spk, c[3], 395, 0   );
nota( spk, c[3], 395, 0   );
nota( spk, d[3], 790, 0   );
nota( spk, d[3], 395, 0   );

nota( spk, e[3], 395, 0   );
nota( spk, f[3], 790, 0   );
nota( spk, f[3], 395, 0   );
nota( spk, g[3], 395, 0   );

nota( spk, a[3], 790, 0   );
nota( spk, b[3], 395, 0   );
nota( spk, c[4], 395, 0   );
nota( spk, d[4], 1580, 0   );

nota( spk, e[4], 790, 0   );
nota( spk, d[4], 593, 0   );
nota( spk, c[4], 198, 0   );
nota( spk, d[4], 790, 0   );

nota( spk, b[3], 395, 0   );
nota( spk, g[3], 395, 0   );
nota( spk, c[4], 790, 0   );
nota( spk, b[3], 593, 0   );

nota( spk, a[3], 198, 0   );
nota( spk, b[3], 790, 0   );
nota( spk, e[3], 395, 0   );
nota( spk, e[3], 395, 0   );

nota( spk, a[3], 790, 0   );
nota( spk, g[3], 593, 0   );
```

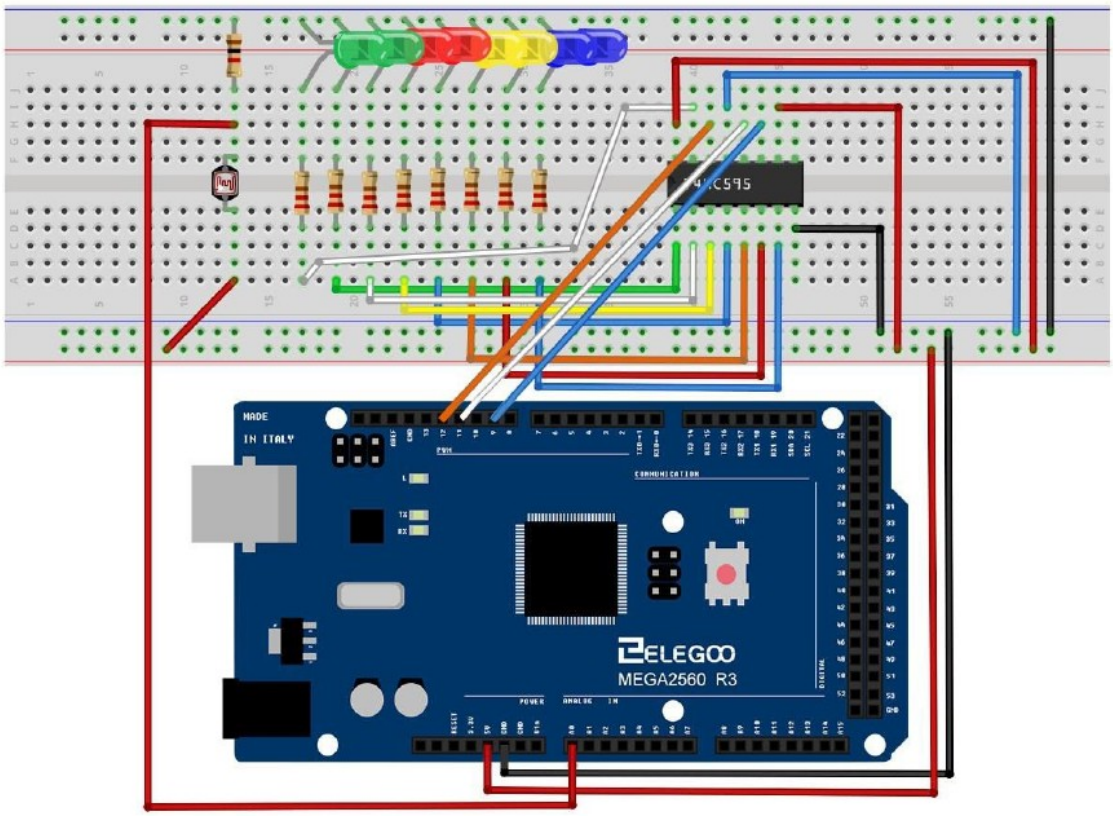
```
nota( spk, f[3], 198, 0   );
nota( spk, g[3], 790, 0   );

nota( spk, c[3], 593, 0   );
nota( spk, c[3], 198, 0   );
nota( spk, c[4], 790, 0   );
nota( spk, b[3], 593, 0   );

nota( spk, a[3], 198, 0   );
nota( spk, g[3], 395, 0   );
nota( spk, b[3], 395, 0   );
nota( spk, c[4], 395, 0   );

nota( spk, d[4], 395, 0   );
nota( spk, e[4], 1580, 0   );
nota( spk, d[4], 395, 0   );
nota( spk, c[4], 395, 0   );

nota( spk, b[3], 395, 0   );
nota( spk, c[4], 395, 0   );
nota( spk, d[4], 1185, 0   );
nota( spk, g[3], 395, 0   );

nota( spk, g[3], 395, 0   );
nota( spk, b[3], 395, 0   );
nota( spk, c[4], 395, 0   );
nota( spk, d[4], 395, 0   );

nota( spk, c[4], 1580, 0   );
nota( spk, b[3], 395, 0   );
nota( spk, a[3], 395, 0   );
nota( spk, g[3], 395, 0   );

nota( spk, a[3], 395, 0   );
nota( spk, b[3], 1185, 0   );
nota( spk, e[3], 395, 0   );
nota( spk, e[3], 395, 0   );

nota( spk, g[3], 395, 0   );
nota( spk, a[3], 395, 0   );
nota( spk, b[3], 395, 0   );
nota( spk, c[4], 790, 0   );

nota( spk, a[3], 593, 0   );
nota( spk, b[3], 198, 0   );
nota( spk, c[4], 790, 0   );
nota( spk, a[3], 593, 0   );

nota( spk, b[3], 198, 0   );
```

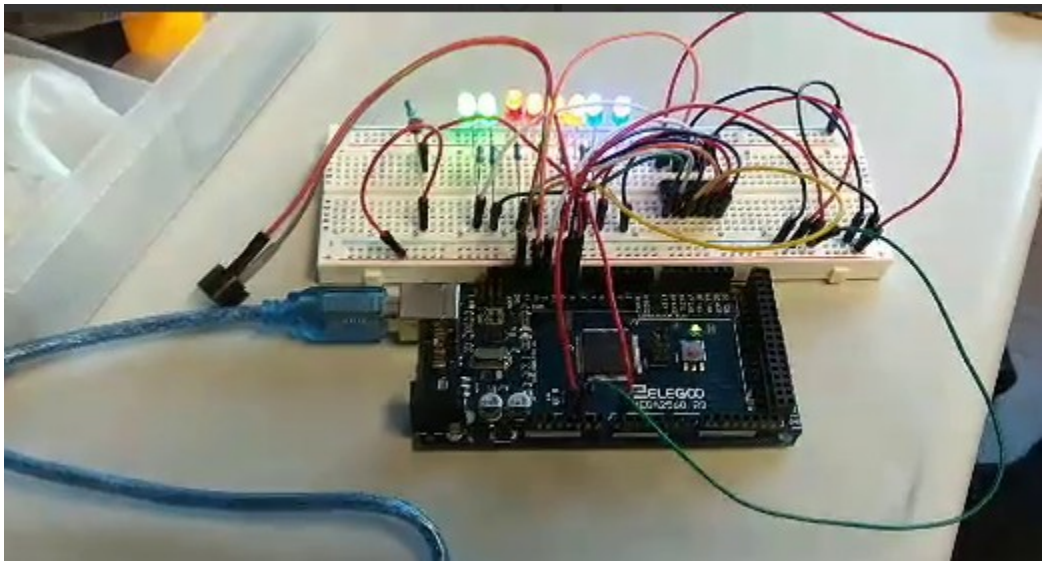
```
  nota( spk, c[4], 790, 0   );
  nota( spk, a[3], 395, 0   );
  nota( spk, c[4], 395, 0   );

  nota( spk, f[4], 1580, 0   );
  nota( spk, f[4], 1580, 0   );
  nota( spk, e[4], 395, 0   );
  nota( spk, d[4], 395, 0   );

  nota( spk, c[4], 395, 0   );
  nota( spk, d[4], 395, 0   );
  nota( spk, e[4], 1185, 0   );
  nota( spk, c[4], 395, 0   );

  nota( spk, c[4], 1580, 0   );
  nota( spk, d[4], 1580, 0   );
  nota( spk, c[4], 395, 0   );
  nota( spk, b[3], 395, 0   );

  nota( spk, a[3], 395, 0   );
  nota( spk, b[3], 395, 0   );
  nota( spk, c[4], 1185, 0   );
  nota( spk, a[3], 395, 0   );

  nota( spk, a[3], 1580, 0   );
  nota( spk, c[4], 790, 0   );
  nota( spk, b[3], 593, 0   );
  nota( spk, a[3], 198, 0   );

  nota( spk, g[3], 790, 0   );
  nota( spk, c[3], 790, 0   );
  nota( spk, c[4], 790, 0   );
  nota( spk, b[3], 593, 0   );

  nota( spk, a[3], 198, 0   );
  nota( spk, g[3], 1185, 0   );
  nota( spk, g[3], 395, 0   );
  nota( spk, g[3], 790, 0   );

  nota( spk, a[3], 395, 0   );
  nota( spk, b[3], 395, 0   );
  nota( spk, c[4], 1580, 0   );
  delay(1000);

}
```

Theremín de Luz

Para montarlo hay que juntar el circuito anterior con este

Wiring diagram





Código:

```
// variable to hold sensor value
int sensorValue;
// variable to calibrate low value
int sensorLow = 1023;
// variable to calibrate high value
int sensorHigh = 0;
// LED pin
const int ledPin = 13;

int lightPin = 0;
int latchPin = 11;
int clockPin = 9;
int dataPin = 12;

int leds = 0;

void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}

void setup() {
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  // Make the LED pin an output and turn it on
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);

  // calibrate for the first five seconds after program runs
  while (millis() < 5000) {
    // record the maximum sensor value
    sensorValue = analogRead(A0);
    if (sensorValue > sensorHigh) {
      sensorHigh = sensorValue;
    }
    // record the minimum sensor value
    if (sensorValue < sensorLow) {
      sensorLow = sensorValue;
    }
  }
  // turn the LED off, signaling the end of the calibration period
  digitalWrite(ledPin, LOW);
}

void loop() {
```

```
  int reading  = analogRead(lightPin);
  int numLEDSLit = reading / 57;  //1023 / 9 / 2
  if (numLEDSLit > 8) numLEDSLit = 8;
  leds = 0;   // no LEDs lit to start
  for (int i = 0; i < numLEDSLit; i++)
  {
    leds = leds + (1 << i);  // sets the i'th bit
  }
  updateShiftRegister();
  //read the input from A0 and store it in a variable
  sensorValue = analogRead(A0);

  // map the sensor values to a wide range of pitches
  int pitch = map(sensorValue, sensorLow, sensorHigh, 50, 4000);

  // play the tone for 20 ms on pin 8
  tone(8, pitch, 20);

  // wait for a moment
  delay(10);
}
```