ESCUELA DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería Informática

PERIFÉRICOS E INTERFACES

PRÁCTICAS DE LABORATORIO 17/18

Módulo 3 – Práctica 3

Programación de interfaces paralelos

"Práctica básica y mejorada"

Ana Isabel Santana Medina

Grupo: 1.18.43

Periodo de realización: 10ª – 15ª semana

Índice

1. Competencias y objetivos básicos de la práctica

Los siguientes datos e información detallada corresponden a la tercera práctica base de la asignatura Periféricos e Interfaces y la mejora de la misma.

Una breve descripción de esta es que se centra en la puesta en práctica de conocimientos teóricos asociados al módulo 3 de la asignatura relativos a los periféricos de entrada y salida de datos. En esta práctica se plantea la programación y control de periféricos a bajo nivel proponiendo al estudiante la implementación de un periférico sencillo, un "Turnomatic", haciendo uso de los puertos de entrada salida de una placa Arduino.

Para realizar la práctica hemos hecho uso de una tarjeta “Arduino Mega” y el circuito del buzzer o altavoz, el display de siete segmentos, 5 pulsadores y el teclado, adecuadamente montado en una tarjeta experimental de laboratorio.

Con la realización de esta práctica hemos alcanzado las siguientes competencias:

1. Capacidad para entender e interrelacionar los diferentes componentes hardware y software que integran un interfaz sencillo de un sistema basado en microcontrolador.
2. Capacidad para diseñar, implementar y verificar el correcto funcionamiento de dispositivos externos sencillos para ser conectados al interfaz paralelo.
3. Capacidad para el desarrollo de programas que permitan el control básico del dispositivo externo haciendo uso de un lenguaje de programación.
4. Capacidad para desarrollar programas que facilite el uso del dispositivo externo a un usuario final.
5. Capacidad para aprender y aplicar nuevos conceptos de forma autónoma e interdisciplinar.
6. Capacidad para emplear la creatividad en la resolución de los problemas.

Gracias a la consecución de los siguientes objetivos:

1. Conocer la estructura interna de un interfaz típico de entrada/salida a través del estudio y manejo de los puertos de una tarjeta Arduino.

2. Poner en práctica los conocimientos básicos sobre las entradas/salidas paralelas en un sistema computador.

3. Conocer y entender la funcionalidad de los diferentes pines de los puertos de E/S del microcontrolador, así como los múltiples aspectos relativos a la conexión con el hardware externo.

4. Conocer los aspectos prácticos de funcionamiento de los diferentes componentes básicos a utilizar en la práctica.

5. Entender la integración y conexión de componentes básicos para diseñar periféricos sencillos que cumplan con una funcionalidad determinada.

6. Analizar, diseñar e implementar el software de control del periférico para su correcto funcionamiento en base a la funcionalidad especificada.

7. Verificar y depurar la integración hardware/software realizada para la aplicación hasta alcanzar un funcionamiento satisfactorio y fiable. Rediseñar si fuese necesario.

2. Descripción y realización de la práctica

Como habíamos comentado anteriormente hemos hecho uso de la placa Arduino, que es la encargada de controlar los periféricos:

- El Display 7 segmentos encargada de mostrar los números que correspondan
- 2 botones o pulsadores para incrementar y disminuir los números
- Un altavoz para reproducir frecuencias
- Un teclado matricial de 3x4 para cambiar el tono de la frecuencia

En primer lugar, he declarado las variables globales, he inicializado los pines correspondientes y he activado los puertos a utilizar, siendo estos el PORTA, PORTC, y

PORTL. En segundo lugar, tras finalizar el setup y en el loop programar los pulsadores de forma adecuada; he procedido a la comprensión y activación de las interrupciones, así como la creación del manejador de las mismas. En tercer lugar, a los métodos para aumentar y decrementar las unidades y las decenas de los números a mostrar.

3. Código de la práctica básica

```
/**
 *
 * Autor: Ana Isabel Santana Medina
 * Grupo: 43
 * Fecha: 26/04/18
 *
 */

//#############################################################################################
// Definición de variables del programa
//#############################################################################################

/**
 * Variables del play 7 Segmentos
 */

const int unidades = 49;    // Señal D4-C1 (pin 49, PL0)
const int decenas = 48;     // Señal D3-C2 (pin 48, PL1)
const int centenas = 47;  // Señal D2-C3 (pin 47, PL2)
const int millares = 46;  // Señal D1-nn (pin46, PL3)

int unidad = 0;   // Número a mostrar en el 7S
int decena = 0;   // Número a mostrar en el 7S
int centena = 0;  // Número a mostrar en el 7S
int millar = 0;   // Número a mostrar en el 7S

const int numeros[10] = {63,6,91,79,102,109,125,7,127,103};  // Números del 0 al 9

/**
```

```
 * Variables de los pulsadores y altavoz
 */

const int pup = 34;    // Pin botón arriba (PC3-in)
const int pdown = 31;  // Pin botón abajo (PC6-in)
const int pright = 30; // Pin botón derecha (PC7-in)
const int pleft = 32;  // Pin botón izquierda (PC5-in)
const int penter = 33; // Pin botón centro (enter) (PC4-in)

const int speaker = 37; // Pin altavoz (PC0-out)

/**
 * Variables auxiliares
 */

int puntero = 0;              // Variable controlar barrido

int frecuencia = 200;       // Variable para guardar el tono actual
char alm = ' ';            // Buffer donde almacena el '*' si se pulso

//############################################################################################
// Comienzo del programa
//############################################################################################

/**
* Preparación en la que activamos los puertos y las interrupciones.
 * Esta solo se realiza una vez.
 */

void setup() {

  Serial.begin(9600);     // Monitor Serie

//############################################################################################

  DDRA = B11111111; // Definimos el PORTA de salida (Bus de salida conectado al 7S)
```

```
  DDRC = B00000001; // Definimos el PORTC de entrada (pulsadores) salvo PC0 (salida frecuencia del altavoz)
  PORTC = B11111000; // Activamos el pull-up interno de las líneas entrada

  DDRL = B00001111; // Definimos el PORTL de entrada (teclado) menos los que son de salida (7S)
  PORTL = B11111111; // Activamos el pull-up interno de las líneas entrada PC7-PC3

//#############################################################################################

  tone(3, 100); // tone(pin, frequency) permite generar una señal por un pin de la frecuencia que se desee
  attachInterrupt(1, interruption, CHANGE); // Activamos interrupciones
}

/**
 * Bucle único y principal, todo lo demás son métodos adicionales
 */

void loop() {

  if(digitalRead(pup)) == LOW){
    incrementarUnidad();
    beep();
  }else if(digitalRead(pdown) == LOW){
    decrementarUnidad();
    beep();
  }else if(digitalRead(pup) == LOW && digitalRead(pdown)){
    decena = 0;
    unidad = 0;
    beep();
  }
  delay(100);
}

//#############################################################################################
// Funciones del turnomatic
//#############################################################################################
```

```
/**
  * Manejador de interrupciones
 */

void interruption(){

  // Manejador 7 Segmentos

  if(puntero == 0){
    encenderplay(unidad, puntero);
  }else if(puntero == 1){
    encenderplay(decena, puntero);
  }else if(puntero == 2){
    PORTA = 0;          // Evitar efecto fantasma
    PORTL = B11111011;  // Encender centenas
  }

  // Manejador teclado
  char tecla = lecturaTeclado();
  if (tecla != 'n'){
    cambiarFrecuencia(tecla);
  }

  if(puntero == 2){
    PORTL = B11111111;  // Apagar centenas
  }

  puntero++;

  // Resetear
  if(puntero == 3){
    puntero = 0;
  }
}

/**
  * Funcion detecta que tecla hemos apretado
 */
```

```
char lecturaTeclado(){
  if(puntero == 0){
    if (digitalRead(42) == LOW){
      return '1';
    }else if (digitalRead(43) == LOW){
      return '4';
    }else if (digitalRead(44) == LOW){
      return '7';
    }else if (digitalRead(45) == LOW){
      return '*';
    }
  }else if(puntero == 1){
    if (digitalRead(42) == LOW){
      return '2';
    }else if (digitalRead(43) == LOW){
      return '5';
    }else if (digitalRead(44) == LOW){
      return '8';
    }else if (digitalRead(45) == LOW){
      return '0';
    }
  }else{
    if (digitalRead(42) == LOW){
      return '3';
    }else if (digitalRead(43) == LOW){
      return '6';
    }else if (digitalRead(44) == LOW){
      return '9';
    }else if (digitalRead(45) == LOW){
      return '#';
    }
  }
  return 'n';
}

/**
```

```
  * Funcion para cambiar la frecuencia con la que se emite el sonido del turnomatic
al gusto siempre y cuando se haya pulsado '*' antes
 */


void cambiarFrecuencia (char c)
{
  if (alm == '*'){
    switch(c){
      case '0':
        frecuencia = 200;
        alm = ' ';
        break;
      case '1':
        frecuencia = 400;
        alm = ' ';
        break;
      case '2':
        frecuencia = 600;
        alm = ' ';
        break;
      case '3':
        frecuencia = 800;
        alm = ' ';
        break;
      case '4':
        frecuencia = 1000;
        alm = ' ';
        break;
      case '5':
        frecuencia = 1200;
        alm = ' ';
        break;
      case '6':
        frecuencia = 1400;
        alm = ' ';
        break;
      case '7':
        frecuencia = 1600;
```

```
        alm = ' ';
        break;
      case '8':
        frecuencia = 1800;
        alm = ' ';
        break;
      case '9':
        frecuencia = 2000;
        alm = ' ';
        break;
      default:
        break;
    }
  }else if (c == '*'){
    alm = c;
  }
}

/**
  * Funcion para manejar las decenas y unidades con el turnomatic
 */

void encenderplay(int n, int p){
  if (p == 0){
    PORTL = B11111110;  // Encender unidades
    PORTA = numeros[n];
  }else if(p == 1){
    PORTL = B11111101;  // Encender decenas
    PORTA = numeros[n];
  }
}

/**
  * Función que incementa las unidades
 */

void incrementarUnidad(){
  unidad++;
```

```
  if (unidad > 9){
    unidad = 0;
    incrementarDecena();
  }
}

/**
  * Función que incrementa las decenas
 */
void incrementarDecena(){
  decena++;
  if (decena > 9){
    decena = 0;
    unidad = 0;
  }
}

/**
  * Función que decrementa las unidades
 */
void decrementarUnidad(){
  unidad--;
  if (unidad < 0 && decena > 0){
    decrementarDecena();
    unidad = 9;
  }else if (unidad < 0){
    unidad = 9;
    decena = 9;
  }
}

/**
  * Función que decrementa las decenas
 */
void decrementarDecena(){
  decena--;
  if (decena == 0){
    unidad = 0;
```

```
  }else if (decena < 0){
    decena = 0;
  }
}


/**
 * Función que reproduce un pitido con una frecuencia determinada
 */


void beep(){
    noTone(3);  // noTone(pin) Desactivamos para aplicar la modificacion de la
frecuencia
    tone(speaker, frecuencia, 100);  // tone(pin, frequency, duration) Pita
    delay(100);
    noTone(speaker);  // Dejamos de aplicar la frecuencia al pin del speaker
    tone(3, 100); // Ponemos la frecuencia por defecto
}
```

4. Mejora de la práctica

La mejora de esta práctica la he basado en cuatro funcionalidades:

- Mostrar un menú.
- Encender un segmento del display de siete segmentos según un número pulsado entre el 1 y el 7 del teclado.
- Cambiar la frecuencia del altavoz con el 0, el 8 y el 9.
- Reproducir una canción con el * o con el #.

5. Código de la práctica mejorada

```
/**
 *
 * Autor: Ana Isabel Santana Medina
 * Grupo: 43
 * Fecha: 05/05/18
 *
 */
```

```
//#########################################################################
// Definición de variables del programa
//#########################################################################

/**
 * String con el menú que se muestra
 */

const String menu[] = {
    "Funciones principales: \n\n",                                      /** 0[1] */
    "  1.- Turnomatic\n",                                               /** 1[2] */
    "Funciones adicionales: \n",                                        /** 3[4] */
    "\n  2.- Encender segmento (1-7)\n",                                /** 4[5] */
    "  3.- Cambiar frecuencia (0, 8, 9) o reproducir cancion (*, #)\n\n" /** 5[6] */
};

/**
 * Variables del display 7 Segmentos
 */

const int unidades = 49;    // Señal D4-C1 (pin 49, PL0)
const int decenas = 48;     // Señal D3-C2 (pin 48, PL1)
const int centenas = 47;    // Señal D2-C3 (pin 47, PL2)
const int millares = 46;    // Señal D1-nn (pin46, PL3)

int unidad = 0;   // Número a mostrar en el 7S
int decena = 0;   // Número a mostrar en el 7S
int centena = 0;  // Número a mostrar en el 7S
int millar = 0;   // Número a mostrar en el 7S

const int numeros[10] = {63,6,91,79,102,109,125,7,127,103};  // Números del 0 al 9
const int segmentos[7] = {1, 2, 4, 8, 16, 32, 64}; // Situación de los segmentos

/**
 * Variables de los pulsadores y altavoz
 */

const int pup = 34; // Pin botón arriba (PC3-in)
```

```
const int pdown = 31; // Pin botón abajo (PC6-in)
const int pright = 30; // Pin botón derecha (PC7-in)
const int pleft = 32; // Pin botón izquierda (PC5-in)
const int penter = 33; // Pin botón centro (enter) (PC4-in)

const int speaker = 37; // Pin altavoz (PC0-out)

/**
 * Variables auxiliares
 */

int puntero = 0;              // Variable controlar barrido

int frecuencia = 200;         // Variable para guardar el tono actual

boolean asterisco = false;    // Variable booleana para el asterisco
boolean teclaMejora = false;  // Variable booleana para el 0, 8, 9, *, #

int frecuenciaMejora;

/**
 * Notas musicales
 */

int c[5]={131,262,523,1046,2093};   // 4 octavas de Do
int cs[5]={139,277,554,1108,2217};  // Do#
int d[5]={147,294,587,1175,2349};   // Re
int ds[5]={156,311,622,1244,2489};  // Re#
int e[5]={165,330,659,1319,2637};   // Mi
int f[5]={175,349,698,1397,2794};   // Fa
int fs[5]={185,370,740,1480,2960};  // Fa#
int g[5]={196,392,784,1568,3136};   // Sol
int gs[5]={208,415,831,1661,3322};  // Sol#
int a[5]={220,440,880,1760,3520};   // La
int as[5]={233,466,932,1866,3729};  // La#
int b[5]={247,494,988,1976,3951};   // Si

//#############################################################################
```

```
// Comienzo del programa
//#############################################################################################


/**
 * Preparación en la que activamos los puertos, sus correspondientes y las interrupciones.
 * Esta solo se realiza una vez.
 */


void setup() {


  Serial.begin(9600);     // Monitor Serie


  Serial.println("______   _____ ________  ____  _____  ____   ____  ________  ____  _____  _____  ______      ___    ");
  Serial.println("|_   _ \\ |_   _||_   __  ||_   \\|_   _||_  _| |_  _||_   __  ||_  \\|_   _||_   _||_   _| .'   `.  ");
  Serial.println(" | |_) |  | |    | |_ \\_|  |   \\ | |    \\ \\   / /    | |_ \\_|  |   \\ | |    | |    | |  .-.  \\ ");
  Serial.println(" |  __'.  | |    |  _| _   | |\\ \\| |     \\ \\ / /     |  _| _   | |\\ \\| |    | |    | |  | || |   | | ");  // Mensaje de bienvenida
  Serial.println("_| |__) |_| |_  _| |__/ | _| |_\\   |_     \\ ' /     _| |__/ | _| |_\\   |_  _| |_  _| |_.' /\\  `-'  / ");
  Serial.println("|_____/|_____||_____||_____|\\____|     \\_/      |_____||_____|\\____||_____||_____||_____|`.___.'  \n");



//#############################################################################################


  DDRA = B11111111; // Definimos el PORTA de salida (Bus de salida conectado al 7S)


  DDRC = B00000001; // Definimos el PORTC de entrada (pulsadores) salvo PC0 (salida frecuencia del altavoz)
  PORTC = B11111000; // Activamos el pull-up interno de las líneas entrada


  DDRL = B00001111; // Definimos el PORTL de entrada (teclado) menos los que son de salida (7S)
  PORTL = B11111111; // Activamos el pull-up interno de las líneas entrada PC7-PC3


//#############################################################################################
```

```
  tone(3, 100); // tone(pin, frequency) permite generar una señal por un pin de la frecuencia que se desee
  attachInterrupt(1, interrupciones, CHANGE); // Activamos interrupciones

  showMenu();  // Mostramos el menú
}

/**
 * Bucle único y principal, todo lo demás son métodos adicionales
 */

void loop() {

  if(!teclaMejora){
    if (digitalRead(penter) == LOW){  // Reset
      decena = 0;
      unidad = 0;
      beep();
    }else if (digitalRead(pup) == LOW){
      incrementarUnidad();
      beep();
    }else if (digitalRead(pdown) == LOW){
      decrementarUnidad();
      beep();
    }
  }else{
    if (digitalRead(pup) == LOW && digitalRead(pdown) == LOW){  // Vuelve a método principal
      teclaMejora = false;
      beep();
    }
  }

  delay(100);

  if(teclaMejora){
    noTone(3);
    interrupcionesMejora();
  }
```

```
}

//##################################################################################
// Funciones adicionales
//##################################################################################

/**
 * Función que muestra el menú
 */

void showMenu() {

  Serial.println("*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*\n");
  for (int i = 0; i <= 4; i++) {
    Serial.println(menu[i]);
  }
  Serial.println("\n*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*\n");
}

/**
 * Función que reproduce un pitido con una frecuencia determinada
 */

void beep(){
    noTone(3);  // noTone(pin) Desactivamos para aplicar la modificacion de la
frecuencia
    tone(speaker, frecuencia, 100);  // tone(pin, frequency, duration) Pita
    delay(100);
    noTone(speaker);  // Dejamos de aplicar la frecuencia al pin del speaker
    tone(3, 100); // Ponemos la frecuencia por defecto
}

//##################################################################################
// Funciones del turnomatic
//##################################################################################

/**
```

```
  * Manejador de interrupciones
 */

void interrupciones(){

  // Manejador 7 Segmentos

  if(puntero == 0){
    encenderSS(unidad, puntero);
  }else if(puntero == 1){
    encenderSS(decena, puntero);
  }else if(puntero == 2){
    PORTA = 0;          // Evitar efecto fantasma
    PORTL = B11111011;  // Encender centenas
  }

  // Manejador teclado
  char tecla = lecturaTeclado();
  if (tecla != 'n'){
    cambiarFrecuencia(tecla);
  }

  if(puntero == 2){
    PORTL = B11111111;  // Apagar centenas
  }

  puntero++;

  // Resetear
  if(puntero == 3){
    puntero = 0;
  }
}

/**
  * Funcion detecta que tecla hemos apretado
 */
```

```
char lecturaTeclado(){
  if(puntero == 0){
    if (digitalRead(42) == LOW){
      return '1';
    }else if (digitalRead(43) == LOW){
      return '4';
    }else if (digitalRead(44) == LOW){
      return '7';
    }else if (digitalRead(45) == LOW){
      asterisco = true;
      return '*';
    }
  }else if(puntero == 1){
    if (digitalRead(42) == LOW){
      return '2';
    }else if (digitalRead(43) == LOW){
      return '5';
    }else if (digitalRead(44) == LOW){
      return '8';
    }else if (digitalRead(45) == LOW){
      return '0';
    }
  }else{
    if (digitalRead(42) == LOW){
      return '3';
    }else if (digitalRead(43) == LOW){
      return '6';
    }else if (digitalRead(44) == LOW){
      return '9';
    }else if (digitalRead(45) == LOW){
      if(asterisco){
        teclaMejora = true;
      }
      return '#';
    }
  }
    return 'n';
}
```

```
/**
  * Funcion para cambiar la frecuencia con la que se emite el sonido del turnomatic
al gusto siempre y cuando se haya pulsado '*' antes
 */


void cambiarFrecuencia (char c){
  if (asterisco){
    switch(c){
      case '0':
        frecuencia = 200;
        asterisco = false;
        break;
      case '1':
        frecuencia = 400;
        asterisco = false;
        break;
      case '2':
        frecuencia = 600;
        asterisco = false;
        break;
      case '3':
        frecuencia = 800;
        asterisco = false;
        break;
      case '4':
        frecuencia = 1000;
        asterisco = false;
        break;
      case '5':
        frecuencia = 1200;
        asterisco = false;
        break;
      case '6':
        frecuencia = 1400;
        asterisco = false;
        break;
      case '7':
```

```
        frecuencia = 1600;
        asterisco = false;
        break;
      case '8':
        frecuencia = 1800;
        asterisco = false;
        break;
      case '9':
        frecuencia = 2000;
        asterisco = false;
        break;
      default:
        break;
    }
  }
}

/**
  * Funcion para manejar las decenas y unidades con el turnomatic
  */

void encenderSS(int n, int p){
  if (p == 0){
    PORTL = B11111110;  // Encender unidades
    PORTA = numeros[n];
  }else if(p == 1){
    PORTL = B11111101;  // Encender decenas
    PORTA = numeros[n];
  }

/**
  * Función que incementa las unidades
 */

void incrementarUnidad(){
  unidad++;
  if (unidad > 9){
    unidad = 0;
```

```
    incrementarDecena();
  }
}

/**
  * Función que incrementa las decenas
 */

void incrementarDecena(){
  decena++;
  if (decena > 9){
    decena = 0;
    unidad = 0;
  }
}

/**
  * Función que decrementa las unidades
 */

void decrementarUnidad(){
  unidad--;
  if (unidad < 0 && decena > 0){
    decrementarDecena();
    unidad = 9;
  }else if (unidad < 0){
    unidad = 9;
    decena = 9;
  }
}

/**
  * Función que decrementa las decenas
 */

void decrementarDecena(){
  decena--;
  if (decena == 0){
```

```
    unidad = 0;
  }else if (decena < 0){
    decena = 0;
  }
}

//#############################################################################################
// Implementación de mejoras
//#############################################################################################

/**
 * Manejador de interrupciones para la mejora
 */

void interrupcionesMejora(){

  // Manejador 7 Segmentos

  if(puntero == 0){
    PORTA = 0;          // Evitar efecto fantasma
    PORTL = B11111110;  // Encender unidades
  }else if(puntero == 1){
    PORTA = 0;          // Evitar efecto fantasma
    PORTL = B11111101;  // Encender decenas
  }else if(puntero == 2){
    PORTA = 0;          // Evitar efecto fantasma
    PORTL = B11111011;  // Encender centenas
  }

  // Manejador teclado

  char tecla = lecturaTeclado();
  if (tecla != 'n'){ // n = no
    teclaPulsada(tecla);
  }

  puntero++;
```

```
  // Resetear

  if(puntero == 3){
    puntero = 0;
  }
}

/**
  * Funcion para encender el segmento que se desee, quedando reservados el 0, el 8, y el 9 para cambio
  * de frecuencias y el * y el # reservados para reproducir una canción
 */

void teclaPulsada(char tecla){
  boolean beepMejora = false;
  switch(tecla){
      case '0':
        frecuenciaMejora = 1600;
        beepMejora = true;
        break;
      case '1':
        encenderSegmento(0);
        break;
      case '2':
        encenderSegmento(1);
        break;
      case '3':
        encenderSegmento(2);
        break;
      case '4':
        encenderSegmento(3);
        break;
      case '5':
        encenderSegmento(4);
        break;
      case '6':
        encenderSegmento(5);
        break;
```

```
      case '7':
        encenderSegmento(6);
        break;
      case '8':
        frecuenciaMejora = 1800;
        beepMejora = true;
        break;
      case '9':
        frecuenciaMejora = 2000;
        beepMejora = true;
        break;
      case '*':
        cancion();
        tone(3, 100);
        break;
      case '#':
        cancion();
        tone(3, 100);
        break;
    }

    if(beepMejora){
      tone(speaker, frecuenciaMejora, 200);
      delay(100);
      noTone(speaker);
    }
}

/**
 * Función para encender un segmento del 7 Segmentos
 */

void encenderSegmento(int tecla){
  PORTL = B11111110;  // Encender unidades
  PORTA = segmentos[tecla];
  delay(500);
}
```

```
 /**
  * Manejador de frecuencia para la canción
  */

void nota(int frecuenciaSong, int retardo){
    tone(speaker, frecuenciaSong); // suena la nota frec recibida
    delay(retardo);                // para despues de un tiempo t
}

/**
 * Canción
 */

void cancion(){
  nota(a[1],800);
  noTone(speaker);
  delay(400);

  nota(e[1],800);
  noTone(speaker);
  delay(400);

  nota(a[1],800);
  noTone(speaker);
  delay(200);

  nota(e[1],400);
  noTone(speaker);
  delay(200);

  nota(a[1],400);
  noTone(speaker);
  delay(200);

  nota(as[1],200);
  noTone(speaker);
  delay(100);
```

```
  nota(b[1],800);
  noTone(speaker);
  delay(400);

  nota(fs[1],800);
  noTone(speaker);
  delay(400);

  nota(b[1],800);
  noTone(speaker);
  delay(200);

  nota(fs[1],400);
  noTone(speaker);
  delay(200);

  nota(b[1],400);
  noTone(speaker);
  delay(200);

  nota(as[1],200);
  noTone(speaker);
  delay(100);

  nota(a[1],800);
  noTone(speaker);
  delay(400);

  nota(e[1],800);
  noTone(speaker);
  delay(400);

  nota(a[1],800);
  noTone(speaker);
  delay(400);
}
```

5. Conclusiones

Las conclusiones que he obtenido de la realización de esta primera práctica han sido que, en primer lugar, las competencias que se habían planteado han sido logradas gracias al estudio y trabajo de la misma.

6. Comentarios finales

Como comentarios finales, he de añadir que, en mi opinión, han resultado suficiente los recursos que nos han facilitado para la realización de la práctica, puesto que aparte de estos, teníamos la capacidad de acceder a internet. Pero he de añadir que el tiempo que teníamos para realizar la práctica no me pareció suficiente; es una práctica que no podemos hacer en casa con nuestro propio circuito, ya que necesitamos muchos componentes, y en estas fechas tenemos muchos más problemas para realizar las prácticas, sobretodo por la cantidad de gente que se junta en los laboratorios después de clases y no hay placas para todos.

7. Recursos empleados

Para la realización de esta práctica he hecho uso de los siguientes recursos:

- **Arduino IDE**
- **Enunciado de la práctica 2017**
- **Transparencias de teoría**
- **Transparencias de la práctica**
- **Página web de Arduino**
- **Documentación varia**