



Transistor BJT y MOSFET

Autor(es):

Mario Eduardo Sánchez Mejía

Fidel Alberto Zarco Áviles

21120721@morelia.tecnm.mx

120121258@morelia.tecnm.mx

Asesor(@s):

Luis Ulises Chávez Campos

Resumen

Se presenta el desarrollo e implementación de un sistema de control electrónico basado en transistores como interruptores, utilizando microcontroladores Arduino. La implementación incorpora componentes específicos como el transistor BD235 y el MOSFET IRLZ14, junto con potenciómetros, para gestionar cargas de alta potencia, particularmente bombillas de 12V. El desarrollo experimental facilita la comprensión práctica del comportamiento de transistores en aplicaciones de conmutación, además de demostrar la implementación efectiva de sistemas de control mediante señales tanto digitales como analógicas. La integración resultante evidencia la aplicación práctica de principios fundamentales en electrónica de potencia y programación de microcontroladores en sistemas embebidos.

Palabras clave: Arduino, Transistores, MOSFET, Sistemas de Control, Electrónica de Potencia



Índice

| | | | |
|--|----------|--|-----------|
| 1. Introducción | 3 | 7. Extensiones del Sistema | 10 |
| 1.1. LEDs (Diodos Emisores de Luz) | 3 | 7.1. Modo de Práctica | 10 |
| 1.2. Botones Pulsadores | 3 | 7.2. Sistema de Detección de Errores | 10 |
| 1.3. Buzzer Pasivo | 4 | 8. Conclusiones y Recomendaciones | 10 |
| 1.4. Resistencias | 4 | 8.1. Resumen de Características . | 10 |
| 2. Consideraciones de Diseño | 4 | 8.2. Recomendaciones de Uso . . . | 10 |
| 3. Objetivo del Proyecto | 5 | Referencias | 12 |
| 4. Desarrollo de la Practica | 5 | | |
| 4.1. Material Utilizado | 5 | | |
| 4.2. Esquemático del Circuito . . . | 5 | | |
| 4.3. Explicación de las Conexiones | 5 | | |
| 4.4. Descripción del Código | 6 | | |
| 4.5. Observaciones y Comentarios | 6 | | |
| 5. Documentación del Piano Digital con Tema de Tetris | 6 | | |
| 5.1. Descripción General | 6 | | |
| 5.2. Configuración Inicial | 7 | | |
| 5.3. Bucle Principal y Control de Notas | 7 | | |
| 5.4. Función Auxiliar de Control LED-Tono | 7 | | |
| 5.5. Implementación del Tema de Tetris | 8 | | |
| 5.6. Tabla de Frecuencias | 8 | | |
| 5.7. Control de Tiempo y Sincronización | 8 | | |
| 5.7.1. Estructura del Tema Musical | 8 | | |
| 5.8. Diagrama de Conexiones . . . | 9 | | |
| 6. Detalles de Implementación | 9 | | |
| 6.1. Estructura de Datos | 9 | | |
| 6.2. Control de Tiempo Avanzado | 9 | | |
| 6.3. Sistema de Efectos LED . . . | 9 | | |
| 6.4. Optimización de Memoria . . | 9 | | |

Índice de figuras

| | |
|---|----|
| 1. Código de control de LED en Arduino | 3 |
| 2. Definición de Variables Globales | 7 |
| 3. Función de Configuración . . . | 7 |
| 4. Bucle Principal y Control de Notas | 7 |
| 5. Función de Control LED-Tono | 7 |
| 6. Implementación del Tema de Tetris | 8 |
| 7. Mecanismos de Control Temporal | 8 |
| 8. Estructura Musical del Tema de Tetris | 8 |
| 9. Diagrama de Conexiones del Sistema | 9 |
| 10. Definición de Arrays para Notas y Pines | 9 |
| 11. Implementación de Control de Tiempo | 9 |
| 12. Sistema de Patrones LED . . | 9 |
| 13. Estructuras de Datos Optimizadas | 10 |
| 14. Implementación del Modo Práctica | 10 |
| 15. Sistema de Verificación | 10 |

1 Introducción

Esta práctica se centra en la creación de un piano básico utilizando Arduino, donde aprenderemos a integrar múltiples componentes electrónicos para crear un sistema interactivo musical (Monk, 2017). Los pianos electrónicos modernos utilizan circuitos y componentes para generar distintas notas musicales, y en esta práctica simularemos este funcionamiento de manera simplificada (Evans et al., 2013).

```
1 const int ledPin = 13;
2
3 void setup() {
4     pinMode(ledPin, OUTPUT);
5 }
6
7 void loop() {
8     digitalWrite(ledPin, HIGH); // Encender
9     delay(1000);
10    digitalWrite(ledPin, LOW);  // Apagar
11    delay(1000);
12 }
13
```

Figura 1: Código de control de LED en Arduino

1.1 LEDs (Diodos Emisores de Luz)

Características y Conexión (Platt, 2014)

- **Polaridad:**
 - Ánodo (+): Pata más larga
 - Cátodo (-): Pata más corta
- **Especificaciones:**
 - Voltaje típico: 2,0 V - 3,3 V
 - Corriente: 20 mA
- **Conexión:**
 - Ánodo → Resistencia → Pin
 - Cátodo → GND

Características y Conexión (Scherz and Monk, 2016)

- **Tipo:** Interruptores (NO)
- **Características:**
 - Sin polaridad específica
 - Requieren resistencia pull-up
 - Estado normal: Abierto
- **Conexión:**
 - Terminal 1 → Pin Arduino
 - Terminal 2 → GND
 - R pull-up 10 kΩ → 5 V



1.3 Buzzer Pasivo

Características y Operación (Arduino, 2024)

■ Características:

- Sin oscilador interno
- Requiere señal PWM
- Voltaje: 3 V-12 V

■ Conexión:

- (+) → Pin PWM Arduino
- (-) → GND

■ Notas musicales:

- | | |
|-------------------------|--------------------------|
| • DO (C4): 261,63 Hz | • SOL (G4): 392,00 Hz |
| • RE (D4): 293,66 Hz | • LA (A4): 440,00 Hz |
| • MI (E4): 329,63 Hz | • SI (B4): 493,88 Hz |
| • FA (F4): 349,23 Hz | • DO (C5): 523,25 Hz |

1.4 Resistencias

Tipos y Usos (Platt, 2014)

■ Para LEDs:

- Rango: 220 Ω - 1 k Ω
- Limitan corriente
- $R = \frac{V_{fuente} - V_{led}}{I_{led}}$

■ Pull-up:

- Valor: 10 k Ω
- Mantiene estado lógico

■ Código de colores:

- 220 Ω : Rojo-Rojo-Marrón
- 1 k Ω : Marrón-Negro-Rojo
- 10 k Ω : Marrón-Negro-Naranja

2 Consideraciones de Diseño

Recomendaciones Importantes (Banzi and Shiloh, 2014)

■ Pines Arduino:

- PWM (3,5,6,9,10,11) → buzzer
- LEDs en pines consecutivos
- Botones con pull-up interno

■ Seguridad:

- Verificar polaridad
- Máx. 40 mA/pin
- Resistencias adecuadas
- Desconectar al modificar

■ Código:

- Constantes para pines/notas
- Debounce en botones
- Funciones estructuradas



3 Objetivo del Proyecto

Objetivos

El objetivo es crear un piano digital funcional que integre múltiples componentes electrónicos (Evans et al., 2013). Se desarrollarán habilidades en:

- Manejo de entradas/salidas digitales
- Generación de tonos con PWM
- Interacciones usuario-dispositivo
- Sistemas multicomponente

4.2 Esquemático del Circuito

Conexiones Principales

[Incluir imagen del esquemático]

- **LEDs:** Conectados a pines 11-19
- **Botones:** Conectados a pines 2-10
- **Buzzer:** Conectado al pin PWM 20

4 Desarrollo de la Practica

4.1 Material Utilizado

Lista de Componentes

- 1 Placa Arduino UNO
- 9 Botones pulsadores
- 9 LEDs
- 9 Resistencias de 1 k Ω para LEDs
- 9 Resistencias de 10 k Ω para pull-up
- 1 Buzzer pasivo
- Cables jumper
- 1 Protoboard

Detalles de Conexion

- **Conexion de LEDs:**
 - Anodo -> Resistencia 1 k Ω -> Pin Arduino
 - Catodo -> GND
 - Resistencia limita corriente a 20 mA
- **Conexion de Botones:**
 - Terminal 1 -> Pin Arduino
 - Terminal 2 -> GND
 - Resistencia pull-up interna habilitada
- **Conexion del Buzzer:**
 - Terminal positivo -> Pin PWM Arduino
 - Terminal negativo -> GND
 - No requiere resistencia limitadora



4.4 Descripción del Código

Estructura del Programa

- **Variables Globales:**
 - Arrays para pines de botones y LEDs
 - Constantes para frecuencias de notas
 - Pin designado para el buzzer
- **Funcion setup():**
 - Configura pines de botones como INPUT_PULLUP
 - Configura pines de LEDs como OUTPUT
 - Inicializa pin del buzzer
- **Funcion loop():**
 - Lee estado de botones
 - Genera tonos correspondientes
 - Controla LEDs asociados
- **Funcion playMelody():**
 - Implementa melodía especial
 - Control de tiempos y secuencias

4.5 Observaciones y Comentarios

Consideraciones Importantes

- El uso de resistencias pull-up internas simplifica el circuito
- La función tone() bloquea algunas interrupciones
- Se recomienda implementar debounce en los botones
- La melodía puede personalizarse según necesidades
- Los LEDs proporcionan retroalimentación visual útil
- El código es escalable para más notas/botones
- **Mejoras Posibles:**
 - Implementar control de volumen
 - Agregar más melodías predefinidas
 - Mejorar el manejo de múltiples botones
 - Añadir efectos de sonido adicionales

5 Documentación del Piano Digital con Tema de Tetris

5.1 Descripción General

Este proyecto implementa un piano digital con Arduino que incluye ocho notas musicales básicas y una función especial que reproduce el tema musical de Tetris. El sistema utiliza

botones como teclas de piano y proporciona retroalimentación visual mediante un LED.

5.3 Bucle Principal y Control de Notas

```
1 int tonePin = 4;  
2 int ledMusic = 3;  
3
```

Figura 2: Definición de Variables Globales

La figura 2 muestra la definición de los pines principales del sistema. El `tonePin` se utiliza para la generación de sonidos, mientras que `ledMusic` controla el LED indicador.

5.2 Configuración Inicial

```
1 void setup() {  
2     for (int pin = 5; pin <= 13; pin++) {  
3         pinMode(pin, INPUT);  
4     }  
5     pinMode(tonePin, OUTPUT);  
6     pinMode(ledMusic, OUTPUT);  
7 }  
8
```

Figura 3: Función de Configuración

La función `setup()` mostrada en la figura 3 realiza la configuración inicial del sistema. Los pines 5-13 se configuran como entradas para los botones, el pin 4 (`tonePin`) como salida para el buzzer, y el pin 3 (`ledMusic`) como salida para el LED indicador.

```
1 void loop() {  
2     if (digitalRead(13) == HIGH) {  
3         tone(tonePin, 262, 100);  
4     }  
5     if (digitalRead(12) == HIGH) {  
6         tone(tonePin, 294, 100);  
7     }  
8     if (digitalRead(11) == HIGH) {  
9         tone(tonePin, 330, 100);  
10    }  
11    if (digitalRead(10) == HIGH) {  
12        tone(tonePin, 349, 100);  
13    }  
14    if (digitalRead(9) == HIGH) {  
15        tone(tonePin, 392, 100);  
16    }  
17    if (digitalRead(8) == HIGH) {  
18        tone(tonePin, 440, 100);  
19    }  
20    if (digitalRead(7) == HIGH) {  
21        tone(tonePin, 494, 100);  
22    }  
23    if (digitalRead(6) == HIGH) {  
24        tone(tonePin, 523, 100);  
25    }  
26    if (digitalRead(5) == HIGH) {  
27        digitalWrite(ledMusic, HIGH);  
28        tetrisTheme();  
29        digitalWrite(ledMusic, LOW);  
30    }  
31    delay(100);  
32 }  
33
```

Figura 4: Bucle Principal y Control de Notas

5.4 Función Auxiliar de Control LED-Tono

```
1 void toneLed(int ledState, int frequency,  
2             int duration) {  
3     digitalWrite(ledMusic, ledState);  
4     tone(tonePin, frequency, duration);  
5     delay(duration);  
6     digitalWrite(ledMusic, LOW);  
7 }  
8
```

Figura 5: Función de Control LED-Tono

5.5 Implementacion del Tema 5.7 Control de Tiempo y Sincronizacion de Tetris

```
1 void tetrisTheme() {  
2     toneLed(HIGH, 659, 250);  
3     toneLed(HIGH, 494, 125);  
4     toneLed(HIGH, 523, 125);  
5     toneLed(HIGH, 587, 250);  
6     toneLed(HIGH, 523, 125);  
7     toneLed(HIGH, 494, 125);  
8     toneLed(HIGH, 440, 250);  
9     toneLed(HIGH, 440, 125);  
10    toneLed(HIGH, 523, 125);  
11    toneLed(HIGH, 659, 250);  
12    toneLed(HIGH, 587, 125);  
13    toneLed(HIGH, 523, 125);  
14    toneLed(HIGH, 494, 375);  
15    toneLed(HIGH, 523, 125);  
16    toneLed(HIGH, 587, 250);  
17    toneLed(HIGH, 659, 250);  
18    toneLed(HIGH, 523, 250);  
19    toneLed(HIGH, 440, 250);  
20    toneLed(HIGH, 440, 500);  
21  
22    delay(250);  
23  
24    toneLed(HIGH, 494, 250);  
25    toneLed(HIGH, 587, 250);  
26    toneLed(HIGH, 659, 250);  
27    toneLed(HIGH, 698, 250);  
28    toneLed(HIGH, 659, 125);  
29    toneLed(HIGH, 587, 125);  
30    toneLed(HIGH, 523, 375);  
31    toneLed(HIGH, 523, 125);  
32    toneLed(HIGH, 587, 250);  
33    toneLed(HIGH, 659, 250);  
34    toneLed(HIGH, 523, 250);  
35    toneLed(HIGH, 494, 250);  
36    toneLed(HIGH, 440, 500);  
37  
38    delay(500);  
39 }  
40
```

Figura 6: Implementacion del Tema de Tetris

```
1 delay(100);  
2 delay(250);  
3 delay(500);  
4  
5 tone(tonePin, frequency, duration);  
6
```

Figura 7: Mecanismos de Control Temporal

5.7.1. Estructura del Tema Musical

```
1 toneLed(HIGH, 659, 250);  
2 toneLed(HIGH, 494, 125);  
3 toneLed(HIGH, 523, 125);  
4  
5 toneLed(HIGH, 587, 250);  
6 toneLed(HIGH, 523, 125);  
7 toneLed(HIGH, 494, 125);  
8  
9 toneLed(HIGH, 494, 250);  
10 toneLed(HIGH, 587, 250);  
11 toneLed(HIGH, 659, 250);  
12
```

Figura 8: Estructura Musical del Tema de Tetris

5.6 Tabla de Frecuencias

5.8 Diagrama de Conexiones

| 1 | Arduino | | Componente |
|----|---------|----|-----------------|
| 2 | ----- | | |
| 3 | Pin 4 | -> | Buzzer (+) |
| 4 | GND | -> | Buzzer (-) |
| 5 | Pin 3 | -> | LED Musical (+) |
| 6 | GND | -> | LED Musical (-) |
| 7 | Pin 13 | -> | Boton Do (C4) |
| 8 | Pin 12 | -> | Boton Re (D4) |
| 9 | Pin 11 | -> | Boton Mi (E4) |
| 10 | Pin 10 | -> | Boton Fa (F4) |
| 11 | Pin 9 | -> | Boton Sol (G4) |
| 12 | Pin 8 | -> | Boton La (A4) |
| 13 | Pin 7 | -> | Boton Si (B4) |
| 14 | Pin 6 | -> | Boton Do (C5) |
| 15 | Pin 5 | -> | Boton Tetris |
| 16 | | | |

Figura 9: Diagrama de Conexiones del Sistema

6 Detalles de Implementación

6.1 Estructura de Datos

La implementación utiliza estructuras de datos simples y eficientes para el manejo de las notas musicales. El siguiente código muestra la organización básica:

```
1 const int NOTAS_BASICAS = 8;  
2 const int FRECUENCIAS[NOTAS_BASICAS] = {262, 294, 330, 369, 415, 466, 522, 587};  
3 const int PINES_BOTONES[NOTAS_BASICAS] = {13, 12, 11, 10, 9, 8, 7, 6};
```

Figura 10: Definición de Arrays para Notas y Pines

6.2 Control de Tiempo Avanzado

El sistema implementa un control de tiempo preciso para la reproducción musical. A con-

tinuación se muestra la implementación detallada:

```
1 unsigned long previousMillis = 0;  
2 const long interval = 100;  
3  
4 void timedTone(int frequency, int duration) {  
5   unsigned long currentMillis = millis();  
6   if (currentMillis - previousMillis >= interval) {  
7     previousMillis = currentMillis;  
8     tone(tonePin, frequency, duration);  
9   }  
10 }
```

Figura 11: Implementación de Control de Tiempo

6.3 Sistema de Efectos LED

El sistema de iluminación LED se ha implementado con diferentes patrones para indicar el estado del piano:

```
1 void ledPattern(int pattern) {  
2   switch(pattern) {  
3     case 0:  
4       digitalWrite(ledMusic, HIGH);  
5       delay(50);  
6       digitalWrite(ledMusic, LOW);  
7       break;  
8     case 1:  
9       for(int i = 0; i < 3; i++) {  
10        digitalWrite(ledMusic, HIGH);  
11        delay(100);  
12        digitalWrite(ledMusic, LOW);  
13        delay(100);  
14      }  
15      break;  
16   }  
17 }
```

Figura 12: Sistema de Patrones LED

6.4 Optimización de Memoria

La siguiente implementación muestra cómo se ha optimizado el uso de memoria:

```
1 struct NotaMusical {
2     uint16_t frecuencia;
3     uint8_t duracion;
4 };
5
6 const PROGMEM NotaMusical tetrisNotas[] = {
7     {659, 250}, {494, 125}, {523, 125},
8     {587, 250}, {523, 125}, {494, 125},
9     {440, 250}, {440, 125}, {523, 125}
10};
```

Figura 13: Estructuras de Datos Optimizadas

```
1 bool verificarSistema() {
2     bool sistemaCorrecto = true;
3
4     for (int i = 0; i < NOTAS_BASICAS; i++) {
5         pinMode(PINES_BOTONES[i], INPUT);
6         if (digitalRead(PINES_BOTONES[i]) == HIGH) {
7             sistemaCorrecto = false;
8             ledPattern(1);
9         }
10    }
11
12    return sistemaCorrecto;
13}
```

Figura 15: Sistema de Verificación

7 Extensiones del Sistema 8 Conclusiones y Recomendaciones

7.1 Modo de Práctica

Se implementó un modo de práctica con el siguiente código:

```
1 void modoPractica() {
2     static uint8_t notaActual = 0;
3     static unsigned long tiempoInicio = 0;
4
5     if (millis() - tiempoInicio > 2000) {
6         tiempoInicio = millis();
7         toneLed(HIGH, FRECUENCIAS[notaActual], 500);
8         notaActual = (notaActual + 1) % NOTAS_BASICAS;
9     }
10 }
```

Figura 14: Implementación del Modo Práctica

7.2 Sistema de Detección de Errores

Se implementó un sistema básico de detección de errores:

8.1 Resumen de Características

El piano digital implementado ofrece las siguientes características principales:

- 8 notas musicales básicas
- Tema musical de Tetris pregrabado
- Retroalimentación visual mediante LED
- Sistema de control de tiempo preciso
- Modo de práctica
- Sistema de detección de errores

8.2 Recomendaciones de Uso

Para un funcionamiento óptimo del sistema, se recomienda:

- Verificar las conexiones antes de cada uso



- Mantener un tiempo mínimo entre pulsaciones de botones
 - Utilizar una fuente de alimentación es-
- table
- Realizar pruebas periódicas del sistema de verificación



Referencias

- Arduino. Tone function reference. <https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/>, 2024. Available at <https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/>.
- Massimo Banzi and Michael Shiloh. *Getting Started with Arduino*. Maker Media, Inc, 3 edition, 2014. ISBN 978-1449363338.
- Martin Evans, Joshua Noble, and Jordan Hochenbaum. *Arduino in Action*. Manning Publications, 2013. ISBN 978-1617290244.
- Simon Monk. *Programming Arduino: Getting Started with Sketches*. McGraw-Hill Education, 2 edition, 2017. ISBN 978-1259641633.
- Charles Platt. *Encyclopedia of Electronic Components Volume 2: LEDs, LCDs, Audio, Thyristors, Digital Logic, and Amplification*. Make Community, LLC, 2014. ISBN 978-1449334185.
- Paul Scherz and Simon Monk. *Practical Electronics for Inventors*. McGraw-Hill Education, 4 edition, 2016. ISBN 978-1259587542.