Implementación del Buffer

Autor(es):

Mario Eduardo Sánchez Mejía
Fidel Alberto Zarco Áviles

21120721@morelia.tecnm.mx
l20121258@morelia.tecnm.mx

Asesor(@s):

Luis Ulises Chávez Campos

Resumen

Se desarrolla un sistema de piano digital mediante Arduino, integrando elementos electrónicos como buzzers, LEDs y botones pulsadores. El proyecto permite la comprensión de generación de tonos musicales a través de programación, empleando un buzzer pasivo y la función tone() de Arduino. El sistema implementa las ocho notas musicales básicas y una función para reproducir melodías MIDI convertidas. La práctica integra conceptos de electrónica digital, programación y principios musicales fundamentales.

Palabras clave: Arduino, Electrónica Digital, Programación, Piano Digital, Sistemas Programables

Índice

1. Introducción **2**
1.1. LEDs (Diodos Emisores de Luz) 2
1.2. Botones Pulsadores 3
1.3. Buzzer Pasivo 3
1.4. Resistencias 3

2. Consideraciones de Diseño **3**

3. Objetivo del Proyecto **4**

4. Explicación del Código **4**

Referencias **9**

Índice de figuras

1. Código de control de LED en Arduino 2
2. Variables globales y función setup 4
3. Función loop y condicionales . 4
4. Condicionales 5
5. Condicionales 5
6. Condicionales 5
7. Funciones stopCurrentMelody() y toneLed() 6
8. Función tetrisTheme() y condicionales 7
9. Función tetrisTheme() y condicionales 7

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).

1.1 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

```
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

1.2 Botones Pulsadores

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

1.4 Resistencias

Tipos y Usos (Platt, 2014)

- **Para LEDs**:
  - Rango: 220 Ω - 1 kΩ
  - Limitan corriente
  - $R = \frac{V_{fuente} - V_{led}}{I_{led}}$
- **Pull-up**:
  - Valor: 10 kΩ
  - Mantiene estado lógico
- **Código de colores**:
  - 220 Ω: Rojo-Rojo-Marrón
  - 1 kΩ: Marrón-Negro-Rojo
  - 10 kΩ: 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

```
void loop() {
        bool anyKeyPressed = false;

        // Piano keys functionality
        if (digitalRead(13) == HIGH) {
                stopCurrentMelody();
                tone(tonePin, 262, 100);  // C4 (Do
    )
                anyKeyPressed = true;
                lastToneTime = millis();
        }
        if (digitalRead(12) == HIGH) {
                stopCurrentMelody();
                tone(tonePin, 294, 100);  //D4 (Re)
                anyKeyPressed = true;
                lastToneTime = millis();
        }

```

Figura 3: Función loop y condicionales

4 Explicación del Código

```
int tonePin = 4;
int ledMusic = 3;
unsigned long lastToneTime = 0;
bool isPlaying = false;
volatile bool shouldStopMelody = false;

void setup() {
        for (int pin = 5; pin <= 13; pin++) {
                pinMode(pin, INPUT);
        }
        pinMode(tonePin, OUTPUT);
        pinMode(ledMusic, OUTPUT);
}

```

Figura 2: Variables globales y función setup

La figura 2 muestra la configuración para el pin 4 para emitir sonidos y el pin 3 para controlar un LED indicador de música. En la función setup(), establece los pines 5 a 13 como entradas y define los pines tonePin y ledMusic como salidas para controlar el sonido y el LED. Además, usa variables para llevar el control del tiempo de reproducción y permite detener la melodía si es necesario.

La figura 3 implementa la funcionalidad de un teclado de piano básico en el loop(). Define una variable anyKeyPressed para indicar si alguna tecla ha sido presionada.

Si el pin 13 detecta una señal alta (HIGH), el programa detiene la melodía actual (con stopCurrentMelody()), reproduce la nota C4 (Do) en el pin de tono (tonePin) a 262 Hz durante 100 ms, marca anyKeyPressed como verdadero, y actualiza el tiempo de la última nota tocada (lastToneTime). Del mismo modo, si el pin 12 está en HIGH, detiene la melodía, reproduce la nota D4 (Re) a 294 Hz durante 100 ms, y actualiza las mismas variables.

```
if (digitalRead(11) == HIGH) {
        stopCurrentMelody();
        tone(tonePin, 330, 100);  // E4 (Mi)
        anyKeyPressed = true;
        lastToneTime = millis();
}
if (digitalRead(10) == HIGH) {
        stopCurrentMelody();
        tone(tonePin, 349, 100);  // F4 (Fa)
        anyKeyPressed = true;
        lastToneTime = millis();
}
if (digitalRead(9) == HIGH) {
        stopCurrentMelody();
        tone(tonePin, 392, 100);  // G4 (Sol)
        anyKeyPressed = true;
        lastToneTime = millis();
}
if (digitalRead(8) == HIGH) {
        stopCurrentMelody();
        tone(tonePin, 440, 100);  // A4 (La)
        anyKeyPressed = true;
        lastToneTime = millis();
}
if (digitalRead(7) == HIGH) {
        stopCurrentMelody();
        tone(tonePin, 494, 100);  // B4 (Si)
        anyKeyPressed = true;
        lastToneTime = millis();
}

```

Figura 4: Condicionales

La figura 4 amplía la funcionalidad del teclado de piano en el loop() de Arduino, permitiendo tocar más notas musicales.

Para cada pin del 11 al 7, el programa revisa si está en HIGH (tecla presionada). Si es así, detiene cualquier melodía en curso (stopCurrentMelody()), reproduce una nota específica (Mi, Fa, Sol, La o Si) en el pin de tono (tonePin) con una frecuencia de entre 330 y 494 Hz durante 100 ms, marca que una tecla fue presionada (anyKeyPressed = true), y actualiza el tiempo en lastToneTime.

```
        if (digitalRead(6) == HIGH) {
                stopCurrentMelody();
                tone(tonePin, 523, 100);  // C5
                anyKeyPressed = true;
                lastToneTime = millis();
        }
        if (digitalRead(5) == HIGH) {
                shouldStopMelody = false;
                tetrisTheme();
        }


```

Figura 5: Condicionales

La figura 5 completa el teclado de piano y agrega una función para tocar una melodía específica.

Cuando el pin 6 está en HIGH, el programa detiene la melodía actual y reproduce la nota C5 (Do en una octava más alta) en el pin de tono (tonePin) a 523 Hz durante 100 ms. Marca que se ha presionado una tecla (anyKeyPressed = true) y actualiza el tiempo en lastToneTime.

Si el pin 5 está en HIGH, el programa habilita la reproducción de la melodía de Tetris (shouldStopMelody = false) y llama a la función tetrisTheme() para iniciar la melodía.

```

        if (!anyKeyPressed && (millis() -
    lastToneTime > 100)) {
                digitalWrite(ledMusic, LOW);
                isPlaying = false;
        }

        delay(10);
        }


```

Figura 6: Condicionales

La figura 6 muestra el apagado del LED indicador de música si no se está presionando ninguna tecla y han pasado al menos 100 ms

desde la última nota tocada.

Primero, verifica si no hay teclas presionadas (!anyKeyPressed) y si el tiempo transcurrido desde la última nota (millis() - lastToneTime) supera los 100 ms. Si ambas condiciones se cumplen, apaga el LED (digitalWrite(ledMusic, LOW)) y marca que la música no está sonando (isPlaying = false). Finalmente, incluye un pequeño retraso de 10 ms para estabilizar el ciclo del loop().

```
1
2  void stopCurrentMelody() {
3          shouldStopMelody = true;
4          noTone(tonePin);
5          digitalWrite(ledMusic, LOW);
6  }
7
8  void toneLed(int frequency, int duration) {
9
10         if (shouldStopMelody) {
11                 return;
12         }
13
14
15         for (int pin = 6; pin <= 13; pin++) {
16                 if (digitalRead(pin) == HIGH) {
17                         shouldStopMelody = true;
18                         return;
19                 }
20         }
21
22         digitalWrite(ledMusic, HIGH);
23         tone(tonePin, frequency, duration);
24         delay(duration);
25         digitalWrite(ledMusic, LOW);
26         delay(50);
27 }
28
29
30
```

Figura 7: Funciones stopCurrentMelody() y toneLed()

La figura 7 enseña la definición dos funciones en Arduino para controlar la reproducción de una melodía y un LED indicador. La función stopCurrentMelody() detiene la melodía actual, marcando la variable shouldStopMelody como true para indicar que la reproducción debe detenerse. Además, apaga el tono en el pin de sonido (tonePin) y el LED asociado (ledMusic).

La función toneLed(int frequency, int duration) reproduce un tono en el pin tonePin durante un tiempo específico. Si se ha indicado que la melodía debe detenerse (shouldStopMelody), la función termina sin hacer nada. Antes de iniciar el tono, también verifica si alguna tecla está presionada (pines del 6 al 13) y detiene la melodía si detecta que una tecla está activa. Al reproducir el tono, enciende el LED, mantiene el tono por el tiempo especificado, y luego apaga el LED. También incluye una breve pausa de 50 ms entre tonos para el parpadeo del LED.

```
1
2  void tetrisTheme() {
3          // Theme A
4          toneLed(659, 250);  // E5
5          if (shouldStopMelody) return;
6          toneLed(494, 125);  // B4
7          if (shouldStopMelody) return;
8          toneLed(523, 125);  // C5
9          if (shouldStopMelody) return;
10         toneLed(587, 250);  // D5
11         if (shouldStopMelody) return;
12         toneLed(523, 125);  // C5
13         if (shouldStopMelody) return;
14         toneLed(494, 125);  // B4
15         if (shouldStopMelody) return;
16         toneLed(440, 250);  // A4
17         if (shouldStopMelody) return;
18         toneLed(440, 125);  // A4
19         if (shouldStopMelody) return;
20         toneLed(523, 125);  // C5
21         if (shouldStopMelody) return;
22         toneLed(659, 250);  // E5
23         if (shouldStopMelody) return;
24         toneLed(587, 125);  // D5
25         if (shouldStopMelody) return;
26         toneLed(523, 125);  // C5
27         if (shouldStopMelody) return;
28         toneLed(494, 375);  // B4
29         if (shouldStopMelody) return;
30         toneLed(523, 125);  // C5
31         if (shouldStopMelody) return;
32         toneLed(587, 250);  // D5
33         if (shouldStopMelody) return;
34         toneLed(659, 250);  // E5
35         if (shouldStopMelody) return;
36         toneLed(523, 250);  // C5
37         if (shouldStopMelody) return;
38         toneLed(440, 250);  // A4
39         if (shouldStopMelody) return;
40         toneLed(440, 500);  // A4
41         if (shouldStopMelody) return;
42
43         delay(250);
44         if (shouldStopMelody) return;
45
46
47
48
```

Figura 8: Función tetrisTheme() y condicionales

El código de la figura 8 define la función tetrisTheme() que reproduce el tema de Tetris (Theme A) utilizando tonos y frecuencias específicos. Cada llamada a toneLed() reproduce una nota musical en el pin designado (tonePin) con una duración específica, representando una parte de la melodía.

La función empieza reproduciendo una secuencia de notas con sus respectivas frecuencias y duraciones, como E5, B4, C5, y así sucesivamente. Después de cada nota, verifica si se ha activado shouldStopMelody, lo cual indica si la melodía debe detenerse y, de ser así, interrumpe la función.

La melodía incluye una breve pausa de 250 ms al final para permitir un intervalo entre repeticiones, haciendo que el tema se sienta más fluido y dándole tiempo al usuario antes de la próxima iteración.

```
1
2          // Theme B
3          toneLed(494, 250);  // B4
4          if (shouldStopMelody) return;
5          toneLed(587, 250);  // D5
6          if (shouldStopMelody) return;
7          toneLed(659, 250);  // E5
8          if (shouldStopMelody) return;
9          toneLed(698, 250);  // F5
10         if (shouldStopMelody) return;
11         toneLed(659, 125);  // E5
12         if (shouldStopMelody) return;
13         toneLed(587, 125);  // D5
14         if (shouldStopMelody) return;
15         toneLed(523, 375);  // C5
16         if (shouldStopMelody) return;
17         toneLed(523, 125);  // C5
18         if (shouldStopMelody) return;
19         toneLed(587, 250);  // D5
20         if (shouldStopMelody) return;
21         toneLed(659, 250);  // E5
22         if (shouldStopMelody) return;
23         toneLed(523, 250);  // C5
24         if (shouldStopMelody) return;
25         toneLed(494, 250);  // B4
26         if (shouldStopMelody) return;
27         toneLed(440, 500);  // A4
28
29         delay(500);  // Longer pause at the end
30         }
```

Figura 9: Función tetrisTheme() y condicionales

La figura 9 amplía la función tetrisTheme() en Arduino para incluir el "Theme B"de la melodía de Tetris. La función utiliza llamadas consecutivas a toneLed() para reprodu-

cir notas como B4, D5, E5, F5 y otras, cada una con una duración determinada. Después de cada nota, verifica la variable shouldStopMelody para ver si la reproducción debe detenerse; si es así, la función finaliza inmediatamente.

Al final de "Theme B", se agrega una pausa más larga de 500 ms para crear un descanso perceptible antes de cualquier repetición o cambio en la reproducción. Esto permite un flujo más natural entre los temas y evita que la melodía se sienta apresurada.

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.