Estudio de Caso Nº 1: Reloj Despertador

Adolfo A. Amador y Kevin J. Esquivel

Facultad de Ingeniería, Universidad Fidélitas

CEM-701: Programación y Diseño de Algoritmos

Ing[4]. Jonathan Arias Román, M.Sc[2]

III cuatrimestre 2024

Índice

Solución 2

Objetivo 2

Funcionamiento del Reloj 2

Configuración de la Alarma 3

Activación de la Alarma 5

Selección de Tono para la Alarma 6

Comportamiento en Diferentes Modos de Operación 7

Lista de Periféricos 9

Diagrama de Bloques 10

Diagrama de Conexión 10

Diagrama de Flujo 11

Solución

Objetivo

La solución consiste en implementar un reloj digital con alarma en un sistema basado en Arduino, simulado en Proteus, el cual permite al usuario configurar tanto la hora actual como una hora de alarma específica. Además, el sistema ofrece la posibilidad de seleccionar entre diferentes tonos para la alarma, los cuales se reproducen utilizando un buzzer. La hora y la configuración de la alarma se muestran en un display de 7 segmentos, con 6 dígitos.

Funcionamiento del Reloj

El sistema de reloj cuenta con un contador que avanza cada segundo, manteniendo un registro de los valores de segundos, minutos y horas. Utilizamos la función “millis()” del Arduino para llevar el control del tiempo sin detener el flujo de otras operaciones en el programa. Este método permite incrementar el tiempo cada segundo de manera precisa sin bloquear el funcionamiento de otros elementos del sistema.

```
// Define variable tipo long que almacena el tiempo actual
unsigned long previousMillis = 0;  // Verifica la ultima actualizacion del reloj, se inicializa en 00 00 00
                                   // Se utiliza unsigned debido a que el tiempo nunca sera negativo
unsigned long alarmCycleMillis = 0; // Ciclo de on/off de la alarma
unsigned long buzzerMillis = 0; // Intervalos para que el buzzer no suene siempre y me vuelva loco

// Define las variables del tiempo
int hour = 0, minute = 0, second = 0;
```

Para desplegar la hora actual en un display de 7 segmentos de 6 dígitos, utilizamos una técnica de multiplexado. El multiplexado consiste en encender y actualizar cada dígito del display de forma secuencial y rápida, lo que crea la ilusión visual de que todos los dígitos están encendidos simultáneamente. Para lograr esto, cada dígito se enciende durante un breve intervalo de tiempo, y los datos correspondientes al número que debe mostrarse en ese dígito se envían a un decodificador (74LS48). Este decodificador convierte el valor binario en la señal que el display necesita para representar el número de manera correcta.


```
void displayMultiplexed(int displayHour, int displayMinute, int displaySecond) { // Funcion hecha por ChatGPT para ayudar con el mutiplexing de los sigitos del display
  int digits[] = {
    displayHour / 10, displayHour % 10,
    displayMinute / 10, displayMinute % 10,
    displaySecond / 10, displaySecond % 10
  };

  displayDigit(digits[0], digit1);
  displayDigit(digits[1], digit2);
  displayDigit(digits[2], digit3);
  displayDigit(digits[3], digit4);
  displayDigit(digits[4], digit5);
  displayDigit(digits[5], digit6);
} // Final de la funcion

void displayDigit(int number, int digitPin) { // Funcion hecha por ChatGPT para ayudar con el mutiplexing de los sigitos del display
  // Logica para escribir al decodificador y pasar de binario a decimal del display
  digitalWrite(pinA, (number & 0x1) ? HIGH : LOW);
  digitalWrite(pinB, (number & 0x2) ? HIGH : LOW);
  digitalWrite(pinC, (number & 0x4) ? HIGH : LOW);
  digitalWrite(pinD, (number & 0x8) ? HIGH : LOW);

  // Habilita el pin del digito
  digitalWrite(digitPin, LOW);
  delayMicroseconds(1000);
  digitalWrite(digitPin, HIGH);
} // Final de la funcion
```

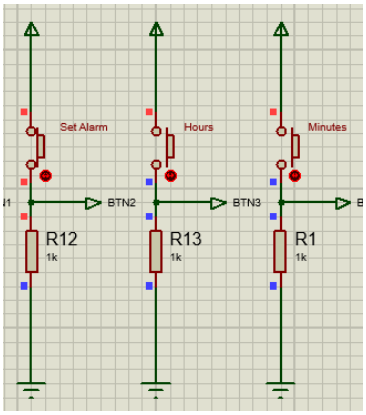
Configuración de la Alarma

El sistema permite al usuario configurar una hora específica para la alarma mediante un conjunto de botones. Para entrar en el modo de configuración de la alarma, el usuario debe presionar el botón correspondiente. Una vez en este modo, puede utilizar los botones de ajuste de horas y minutos para definir la hora a la que se activará la alarma. Esta configuración de la alarma se almacena en variables específicas de “alarmHour” y “alarmMinute”.

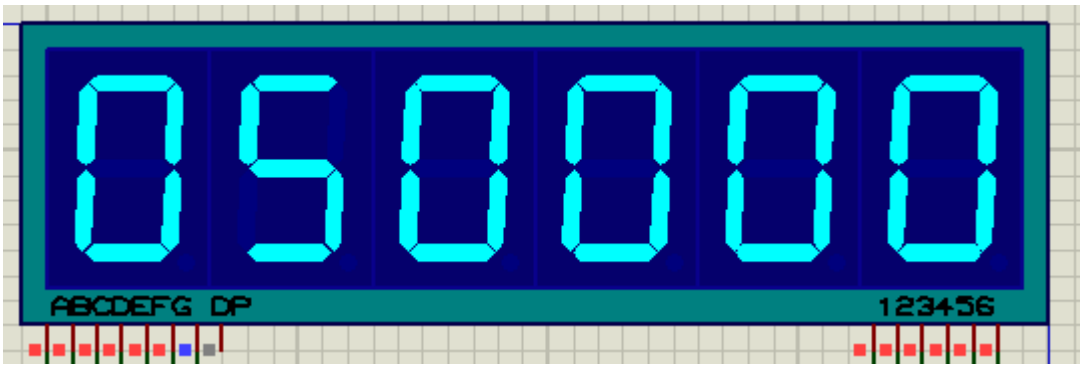
```
// Función que va a manejar la alarma (el ciclo, activar el buzzer, la activación en si, y la desactivación)
void handleAlarm() {
  if (millis() - buzzerMillis >= 1000) { // Crea los intervalos de 1 segundo para que el buzzer no esté constante
    buzzerMillis = millis();  // Reinicia el ciclo

    if (buzzerOn) {
      noTone(Buzzer);  // Apaga el buzzer
      buzzerOn = false;
    } else {
      playSelectedTone(); // Activa el buzzer con el tono seleccionado
      buzzerOn = true;
    }
  }

  // Desactiva la alarma si algún boton es presionado (no se toma en cuenta el boton de SetTone)
  if (digitalRead(setTimeButton) == HIGH || digitalRead(setAlarmButton) == HIGH ||
      digitalRead(hourButton) == HIGH || digitalRead(minuteButton) == HIGH ||
      digitalRead(secondButton) == HIGH) {
    alarmActive = false;           // Desactiva la alarma
    userDeactivatedAlarm = true;   // Hace que no se reactive
    noTone(Buzzer);                // Verifica que se apaga el buzzer
    buzzerOn = false;              // Reinicia el estado del buzzer
  }
} // Final de la función
```

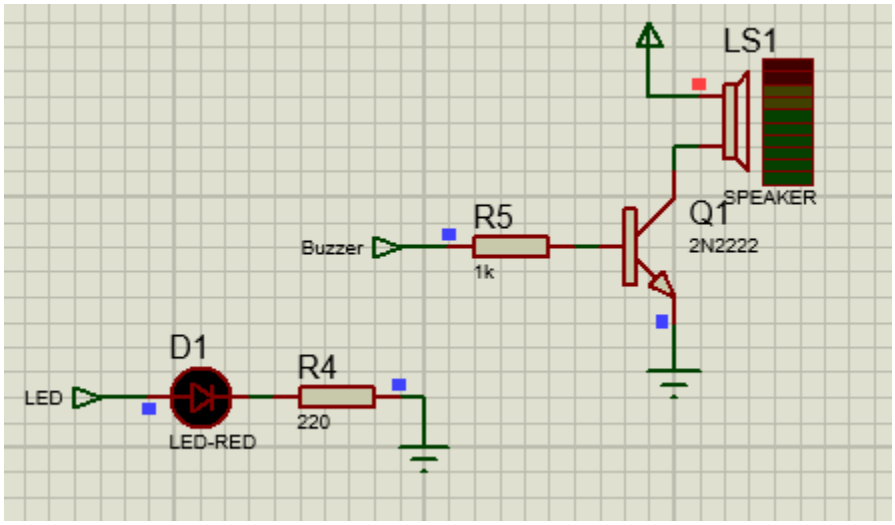


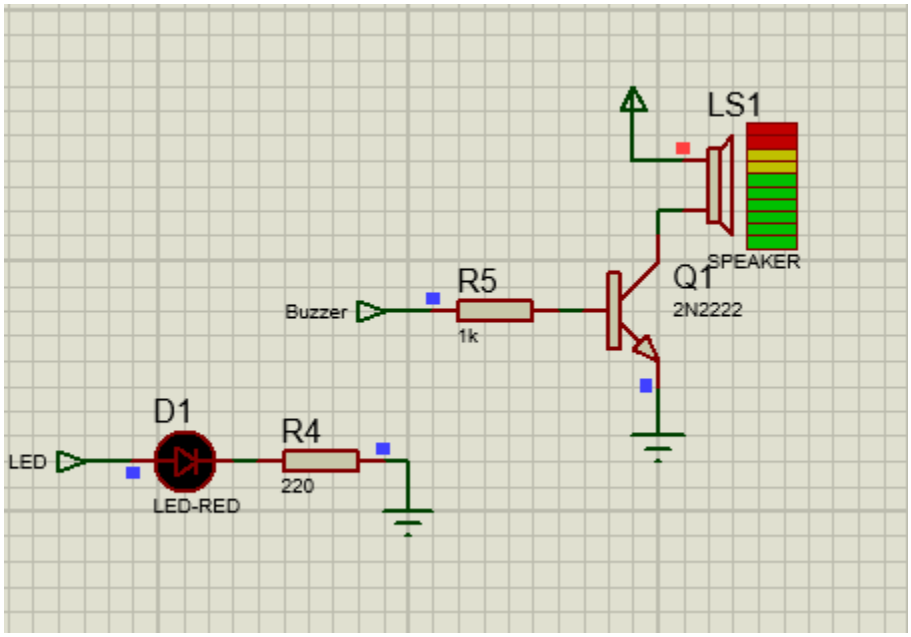
El sistema verifica constantemente si la hora del reloj coincide con la hora configurada para la alarma. En el momento en que la hora y los minutos coinciden, la alarma se activa.



Activación de la Alarma

Cuando el reloj alcanza la hora de la alarma, se activa el buzzer para indicar que la alarma está sonando. La activación del buzzer no es continua; en lugar de ello, se utiliza un patrón de encendido y apagado de un minuto de duración. Es decir, el buzzer suena durante un minuto, luego se apaga durante otro minuto, y este ciclo se repite indefinidamente hasta que el usuario presione cualquier botón, lo cual desactiva la alarma.

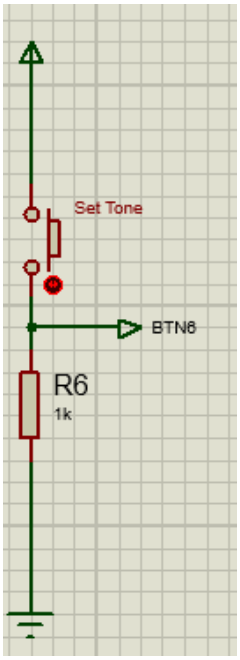




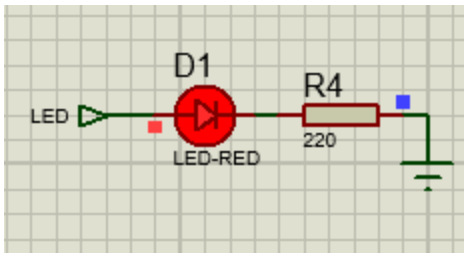
Esta funcionalidad de "alarma indefinida" asegura que la alerta sonora no pase desapercibida y permite al usuario desactivarla fácilmente mediante la pulsación de cualquiera de los botones disponibles.

Selección de Tono para la Alarma

Para añadir flexibilidad en la personalización de la alarma, el sistema permite al usuario seleccionar entre varios tonos disponibles. El usuario puede recorrer los tonos disponibles mediante un botón específico de "Selección de Tono". Cada vez que se pulsa este botón, el sistema recorre entre una serie de tonos predefinidos, asignando uno nuevo a la alarma. Para facilitar la selección, cada tono se reproduce brevemente cuando se selecciona, permitiendo al usuario escucharlo antes de tomar una decisión. Una vez seleccionado, este tono se almacena y será el que suene cuando la alarma se active.



Además, un LED indicador se enciende para señalar visualmente el tono actual que está sonando o el que ha sido seleccionado. Esto brinda retroalimentación visual adicional al usuario, mejorando la experiencia de uso.




```
// Maneja la selección del tono
void handleToneSelection() {
  if (digitalRead(toneButton) == HIGH && !alarmActive) { // Se activa si se presiona el boton de Set Tone
    delay(200); // Antirrebote
    selectedTone = (selectedTone % 5) + 1; // Hace un ciclo por los 5 tonos
    indicateTone(selectedTone);            // Muestra el tono seleccionado en el LED
    playSelectedTone();                    // Hace que suene el tono momentaneamente
    delay(200);                            // Delay para poder oir el tono, sino no servía
    noTone(Buzzer);                        // Para el tono despues de elegirlo
  }
} // Final de la función

// Hace que suene el tono elegido, hecha por ChatGPT
void playSelectedTone() {
  switch (selectedTone) {
    case 1: tone(Buzzer, 500); break;
    case 2: tone(Buzzer, 600); break;
    case 3: tone(Buzzer, 700); break;
    case 4: tone(Buzzer, 800); break;
    case 5: tone(Buzzer, 900); break;
  }
} // Final de la funcion

// Muestra el tono elegido en el LED
void indicateTone(int tone) {
  for (int i = 0; i < tone; i++) {
    digitalWrite(toneLED, HIGH);
    delay(100);
    digitalWrite(toneLED, LOW);
    delay(100);
  }
} // Final de la funcion
```

Comportamiento en Diferentes Modos de Operación

El sistema tiene tres modos principales de operación:

1. Modo Normal:En este modo, el reloj avanza continuamente y se muestra la hora actual en el display.
2. Modo de Configuración de Hora Actual: Al activar este modo, el usuario puede ajustar la hora, minutos y segundos del reloj utilizando los botones correspondientes. Durante

este modo, el reloj se pausa para permitir al usuario definir correctamente la hora actual. Este es el único modo en el cual el reloj deja de avanzar.

3. Modo de Configuración de Alarma: En este modo, el usuario puede definir la hora de la alarma sin pausar el reloj. Al salir de este modo, la alarma queda configurada y el reloj continúa avanzando de forma normal.
4. Modo de Selección de Tono: Al activar este modo, el usuario puede recorrer y seleccionar un tono específico para la alarma. Durante la selección del tono, el reloj sigue funcionando normalmente en segundo plano. Este modo no afecta el avance del tiempo en el reloj ni interfiere con la visualización de la hora en el display.

Lista de Periféricos

- **Botones:** Permiten al cliente configurar la hora por defecto y la alarma, seleccionar ya sea horas, minutos y segundos, así como seleccionar el tono de sonido de la alarma. Este componente ofrece una forma de interacción simple y directa para poder acceder a las configuraciones disponibles, ya que cada botón está dedicado a cada una de las funciones específicas.
- **Decodificador:** Su propósito es controlar los segmentos del display. El uso de pines dedicados para el decodificador permite reducir la cantidad de salidas requeridas para mostrar dígitos en el display. Esto facilita la multiplexación de varios dígitos usando menos pines del Arduino.
- **Display de 7 segmentos:** Permite mostrar en este caso la hora y la configuración de la alarma en un formato digital. Este periférico al tener 6 dígitos permite mostrar la hora, minutos y segundos en formato *hh:mm:ss* para ofrecer una experiencia clara y completa.
- **Amplificador o buzzer:** Permite emitir el sonido despertador de la alarma. Este dispositivo permite una salida audible para alertar al cliente cuando la alarma esta activa, con la capacidad de producir diferentes frecuencias de tono configurables.
- **Led indicador:** Su propósito es indicar visualmente el tono seleccionado de la alarma. Esto sin necesidad de escuchar el amplificador de sonido constantemente.

Diagrama de Bloques

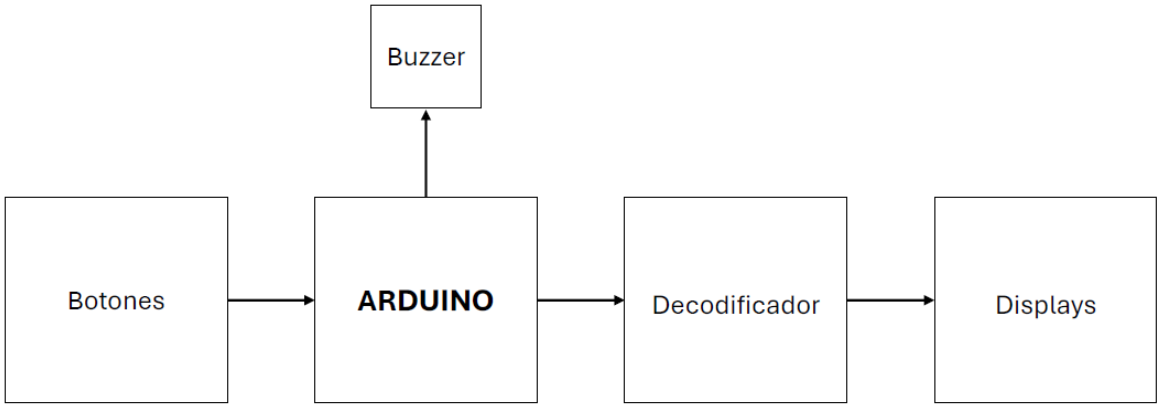


Diagrama de Conexión

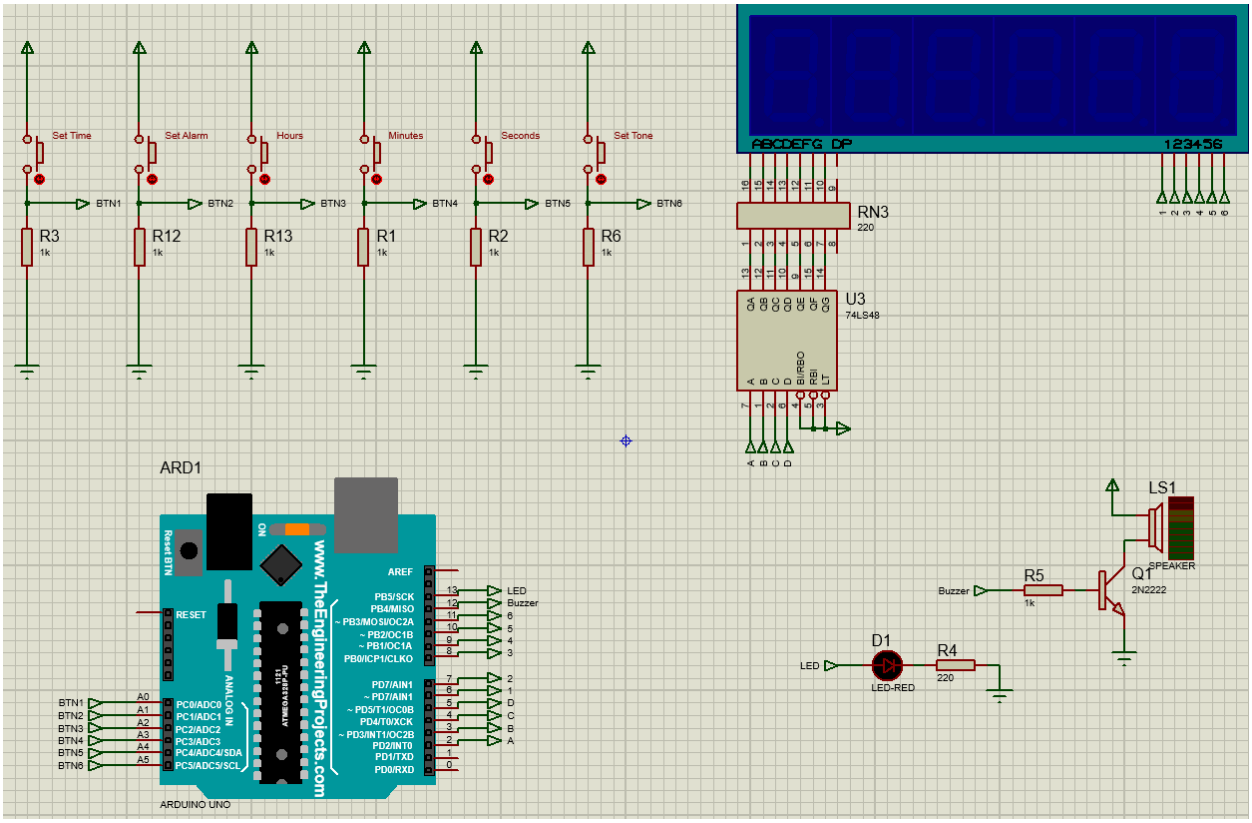


Diagrama de Flujo