UNIVERSIDAD DE ANTIOQUIA

Facultad de Ingeniería

Informe parcial l

Informática II

Santiago Palacio Cárdenas
CC: 1045076775

Juan José Díaz Zuluaga
CC: 1001456540

Docente
Aníbal José Guerra Soler

2023-2
UNIVERSIDAD DE ANTIOQUIA
FACULTAD DE INGENIERÍA

SEDE MEDELLÍN
Septiembre 2023

Medellín, Colombia

Introducción

Las matrices de LEDs son dispositivos electrónicos que se utilizan para crear imágenes y patrones. Se componen de LEDs que se pueden encender y apagar de forma individual. Las matrices de LEDs se utilizan en una variedad de aplicaciones, como pantallas de publicidad, letreros electrónicos y juguetes.

El objetivo de este proyecto es controlar una matriz de LEDs de 8x8 mediante un Arduino. Para ello, se debe diseñar un sistema de control que permita encender y apagar los LEDs y generar diferentes secuencias.

Análisis del problema

Esta aplicación tiene como finalidad la implementación de un sistema de control destinado a una matriz de LEDs de 8x8, empleando un Arduino. Este desafío técnico engloba la gestión de diversos componentes, lo que requiere un sólido entendimiento en electrónica y programación. Además, es relevante destacar que esta necesidad surge debido a la versatilidad para crear una amplia variedad de efectos visuales que poseen numerosas aplicaciones en la vida cotidiana.

Los requisitos del sistema son los siguientes:

- El máximo número de pines digitales de Arduino para utilizar es 7.
- No se debe usar NeoPixel en la implementación.
- Se deben investigar las propiedades y el funcionamiento del integrado 74HC595 y considerar utilizarlo en su implementación.

Las tareas que se definieron para el desarrollo de los algoritmos son las siguientes:

- Establecer el control de versiones con GitHub.(Enlace al Repositorio)
- Creación del proyecto en Tinkercad.(Enlace al proyecto)
- Realizar las conexiones necesarias del Arduino con los integrados.
- Implementar una función que permita enviar los datos a los circuitos integrados.
- Implementar una función que permita verificar el funcionamiento de los LEDs.
- Implementar funciones que permitan generar cada uno de los patrones especificados.
- Publik: Implementar una función que permita al usuario seleccionar las opciones del sistema.

Consideraciones para la alternativa de solución propuesta

La alternativa de solución propuesta consiste en utilizar un registro de desplazamiento de 8 bits para controlar los ánodos y otro registro de desplazamiento de 8 bits para controlar los cátodos. De esta forma, se puede encender un LED determinado al establecer valores para cada una de las salidas de los circuitos integrados y generar los patrones que se requieren en este proye

Esta solución presenta las siguientes ventajas:

- Es simple de implementar y de entender. La lógica es clara y directa, lo que la hace ideal para proyectos educativos y de hobby.

- Es eficiente en términos de recursos. Los registros de desplazamiento son dispositivos relativamente sencillos y económicos, lo que hace que esta solución sea asequible para una amplia gama de proyectos.

- Permite controlar los LEDs de forma individual. Esto proporciona una gran flexibilidad para crear diseños gráficos y animaciones.

Las consideraciones para la implementación de esta solución incluyen:

- La tensión de alimentación de la matriz de LEDs. El circuito debe estar diseñado para proporcionar la tensión adecuada a los LEDs y que se puedan encender.
- La corriente máxima que puede soportar cada LED. El circuito debe limitar la corriente a un nivel seguro para los LEDs.

- Un circuito simple y que evite la cargas de corriente altas, ya que fácilmente pueden quemarse los integrado o los leds

En general, la alternativa de solución propuesta es un camino sólido que ofrece una serie de ventajas significativas, y además es una excelente opción para proyectos que requieren la capacidad de controlar LEDs de forma rápida y sencilla.

Esquema del desarrollo de los algoritmos

Las tareas que se definieron en el desarrollo de los algoritmos son las siguientes:

- Conectar la matriz de LEDs al Arduino.
- Implementar la función verificación() para revisar el funcionamiento de los LEDs.
- Implementar la función imagen() para mostrar una imagen en la matriz de LEDs.
- Implementación de las funciones rect_patt(), X_patt(), arr_patt() y alt_patt() para generar los patrones deseados.
- Lograr la recepción de datos de entrada a través del puerto de comunicación serial: Los datos de entrada pueden ser números binarios o variables relacionadas al control del programa.
- Convertir los datos de entrada a un formato compatible con el registro de desplazamiento: Los datos de entrada deben manipularse para que coincidan con las necesidades del programa
- Enviar los datos al registro de desplazamiento: Los datos convertidos se envían al registro de desplazamiento mediante el puerto serie de la placa Arduino.
- Actualizar el estado de la matriz de LED: El estado de la matriz de LED se actualiza cada vez que se envían datos al registro de desplazamiento.

Algoritmos implementados

Los algoritmos implementados son los siguientes:

- Algoritmo para recibir los datos de entrada: Este algoritmo utiliza el puerto serie de la placa Arduino y ciclos while para recibir la información ingresada por el usuario

- Algoritmo para convertir los datos de entrada: Este algoritmo lee la información del puerto serial para transformarla en valores aptos para el programa.

- Algoritmo para enviar los datos al registro de desplazamiento: Este algoritmo transmite la secuencia binaria a los circuitos integrados para que pasen a las salidas unidas a los ánodos y cátodos de los leds

- void patt_print(int latchPin, int dataPin, int clockPin, int a, int b): Esta es la función más importante del código, ya que es la que se encarga de enviar la información al chip integrado. Envía el valor binario a los dos 74HC595 uno para los cátodos y otro para los ánodos. Y además requiere los pines de latch, data y reloj para implementar la función ShiftOut() que es la que transmite los valores binarios.

- void off(): Esta función apaga todos los LEDs en la matriz estableciendo el valor del "Shift Register" de los ánodos a 0B00000000 o 0 y el de los cátodos a 0B11111111 a 255.

- void verificacion(int iter, int delays): Esta función enciende y apaga todos los LEDs de la matriz de forma secuencial. Recibe un entero que especifica la cantidad de iteraciones que se realiza y otro entero que define cuanto tiempo hay entre verificaciones. Esta secuencia se implementa utilizando un bucle "for" que itera la cantidad de veces que se deben encender todos los leds. Utiliza la función 'patt_print' para configurar los valores de los cátodos y ánodos de la matriz. En el ánodo se envía 255 y en los cátodos 0. Además se utiliza la función off() para apagar la matriz cuando finalizan los ciclos.

- void imagen(unsigned long, dur, int del): Esta función permite al usuario ingresar una imagen de 8x8 como una cadena de 8 caracteres, la imagen se ingresa como una serie de valores binarios (0 y 1), cada uno de los cuales representa un LED. La función luego enciende los LEDs de acuerdo con la imagen ingresada. Para cada fila de la matriz. Los valores se ingresan a través del puerto serie. La función luego ilumina la matriz según los valores ingresados durante un tiempo especificado 'dur' y con un retardo 'del' entre cada iteración. Esta función se implementó utilizando un array de memoria dinámica para almacenar los valores decimales de las 8 cadenas de datos.
  Para encenderlos, se emplea también la función patt_print() para imprimir las 8 líneas en la matriz.

- void rect_patt(unsigned long dur, unsigned long inicio): Esta función muestra el primer patrón de iluminación en forma de rectángulo que se solicita. Recibe las variables dur e inicio que permiten saber en qué momento de la ejecución se invoca la función y utiliza un ciclo while para que la ejecución se dé en un tiempo menor a la

duración esperada. Este patrón utiliza una matriz con memoría dinámica para almacenar los valores que deben ser enviados a los integrados de cátodos y ánodos. Y se apoya en las simetrías del patrón para generar las secuencias necesarias. Finalmente se imprime usando patt_print().

- void X_patt(unsigned long dur, unsigned long inicio): En cada iteración del bucle, se establecen los valores de los registros de desplazamiento para encender los LEDs correspondientes a la X. Similar a 'rect_patt', toma 'dur' como la duración de la animación y 'inicio' como el tiempo en que comenzó. Esta función utiliza un arreglo unidimensional 'p' para almacenar los valores de cátodos y ánodos correspondientes a cada paso.

- void alt_patt(unsigned long dur, unsigned long inicio): Esta función al igual que las dos anteriores recibe los parámetros de inicio y duración para que la figura solo se muestre durante este periodo de tiempo. Y genera la secuencia de una manera mucho más sencilla que las anteriores, ya que en esta se puede especificar con facilidad a través 4 valores el patrón solicitado.
  patt_print(latchPin, dataPin ,clockPin, 51, 219); 0B00110011 &0B11011011)
  patt_print(latchPin, dataPin ,clockPin, 204, 109); 0B11001100 & 0B01101101)

- void arr_patt(unsigned long dur, unsigned long inicio): Esta función muestra un patrón de iluminación en forma de una flecha, en la que nuevamente se usan los parámetros duración e inicio como en las funciones anteriores. Para generar esta secuencia se usa una matriz dinámica donde se almacenan los valores que se envían a los dos circuitos integrados. Para lograrlo, se resta un bit a la izquierda y se suma siempre uno a la derecha para que las filas de 4 leds se desplacen una posición y creen el patrón necesatio.

- void publik(): Esta función maneja la entrada del usuario a través del puerto serie. Espera que el usuario ingrese un número entre 1 y 3 para seleccionar una de las opciones: verificación, imagen por pantalla o patrones. Luego, solicita al usuario que ingrese un tiempo de retardo. Llama a las funciones correspondientes según la opción seleccionada.

- En el setup(), se inicializan los pines utilizados y se configura la comunicación a través del puerto serie. Y en el loop() se ejecuta continuamente la función publik() que concentra a todas las funciones definidas anteriormente para esperar la entrada del usuario y ejecutar las opciones seleccionadas. Las otras funciones se utilizan para mostrar diferentes patrones en la matriz de LEDs controlada por el registro de desplazamiento.

Problemas de desarrollo que afrontó

- El principal problema que se enfrentó durante el desarrollo del proyecto fue la dificultad para controlar los LEDs de forma individual. Para ello, se utilizó un registro de desplazamiento de 8 bits para controlar los ánodos y otro registro de desplazamiento de 8 bits para controlar los cátodos. De esta forma, se puede encender un LED determinado al establecer un 1 en el bit correspondiente del registro de ánodos y un 0 en el bit correspondiente del registro de cátodos.

- Dificultad para diseñar un algoritmo para convertir los datos de entrada a un formato compatible con el registro de desplazamiento. Este algoritmo debe tener en cuenta que cada LED se representa con un bit, por lo que los datos de entrada deben ser convertidos a un formato binario de 8 bits.

- Sincronización entre el algoritmo para enviar los datos al registro de desplazamiento y la actualización del estado de la matriz de LED. Estos dos algoritmos deben estar sincronizados para que el estado de la matriz de LED se actualice correctamente.

- El uso del integrado 74HC595. Este es un circuito que permite registrar datos y con el uso del latch, enviarlos como una trama conjunta o en serie. Este es un circuito que no tiene mucha información disponible y que requiere de investigación profunda para entender correctamente su funcionamiento.

- La principal dificultad que se enfrentó fue la de generar los patrones deseados. Para ello, se utilizaron secuencias lógicas para controlar los registros de desplazamiento.

Evolución de la solución

La solución original del proyecto consistía en utilizar un solo algoritmo para recibir, convertir y enviar los datos al registro de desplazamiento. Sin embargo, este algoritmo era demasiado complejo y difícil de depurar.

Para resolver este problema, se decidió dividir la solución en cuatro algoritmos separados. Esto facilitó el desarrollo y la depuración de cada algoritmo.

Además, se tuvieron en cuenta las siguientes consideraciones durante la implementación:}

- Eficiencia: El algoritmo debe ser eficiente para evitar retrasos en la actualización del estado de la matriz de LED.
- Verificabilidad: El algoritmo debe ser verificable para garantizar que funciona correctamente.
- Mantenibilidad: El algoritmo debe ser fácil de mantener y modificar.

Conclusiones

El proyecto ha sido un éxito, ya que se ha implementado un algoritmo que permite controlar una matriz de LED 8x8 mediante una placa Arduino. El algoritmo es eficiente, verificable y mantenible. Además, es flexible, puesto que permite mostrar imágenes, patrones o secuencias de encendido y apagado de LED.

Es válido además resaltar que este es un proyecto que requiere de una alta interdisciplinariedad, ya que hay conocimientos involucrados en distintas áreas y es un ejercicio valioso para salir de la zona de confort en cuanto a la programación y el control.