**PARCIAL I, INFORMÁTICA II 2023**

**ANÁLISIS, DISEÑO Y POSIBLE IMPLEMENTACIÓN DE SOLUCIÓN.**

**GÓMEZ CARDONA, VICTOR JULIÁN**

**VILORIA COLÓN, PEDRO ANDRÉS**

**GUERRA SOLER, ANIBAL JOSE**

**UDEA**

**2023**

Tabla de contenido

[**COMPRENSIÓN DEL PROBLEMA** 3](#_Toc146556414)

[**ALTERNATIVAS DE SOLUCIÓN** 3](#_Toc146556415)

[**ESQUEMA DE TAREAS** 5](#_Toc146556416)

[**Alternativa 1: Control Multiplexado de LEDs** 5](#_Toc146556417)

[**Alternativa 2: Almacenamiento de Patrones en Arreglos** 6](#_Toc146556418)

[**PROBLEMAS DE DESARROLLO QUE AFRONTÓ.** 6](#_Toc146556419)

[**EVOLUCIÓN DE LA SOLUCIÓN Y CONSIDERACIONES PARA TENER EN CUENTA EN LA IMPLEMENTACIÓN** 7](#_Toc146556420)

[**BIBLIOGRAFÍA** 7](#_Toc146556421)

# **COMPRENSIÓN DEL PROBLEMA**

El problema se divide en cinco tareas principales:

1. Establecer las conexiones físicas para activar 64 LEDs siguiendo una estructura dada, con una restricción de uso máximo de 7 pines digitales del Arduino.
2. Crear una función de verificación para asegurar que todos los LEDs funcionan correctamente.
3. Desarrollar una función de imagen que permita al usuario mostrar patrones personalizados en la matriz de LEDs a través del monitor serial.
4. Generar cuatro patrones específicos en la matriz de LEDs a partir de secuencias, evitando predefinir trivialmente qué LEDs se encienden.
5. Implementar una función de control serial que permita al usuario seleccionar acciones, como la verificación de LEDs, la visualización de imágenes y la alternancia entre patrones.

Luego de comprender el problema y de realizar investigaciones relacionadas con posibles soluciones al problema podríamos plantear de primer momento estas alternativas:

# **ALTERNATIVAS DE SOLUCIÓN**

**CONTROL MULTIPLEXADO DE LEDS:**

El control multiplexado de LEDs es una alternativa eficiente y elegante para manejar una matriz de LEDs utilizando un número limitado de pines digitales en un Arduino. Esta técnica implica dividir la matriz en filas y columnas y alternar entre ellas para encender los LEDs de manera secuencial. Cada fila y columna se activa durante un breve período de tiempo antes de pasar a la siguiente, lo que da la impresión de que todos los LEDs están encendidos al mismo tiempo. Es importante anotar que esta alternativa nos ofrece unas ventajas, pero en el mismo sentido nos propone algunos retos en la implementación, brevemente los describiremos.

**Ventajas:**

* **Optimización de Pines:** Esta técnica permite controlar una gran cantidad de LEDs utilizando un número mínimo de pines digitales, lo que cumple con la restricción de un máximo de 7 pines.
* **Eficiencia en el Uso de Recursos:** Al encender los LEDs de forma secuencial, se utiliza menos corriente y recursos computacionales en comparación con un enfoque en el que todos los LEDs están activos simultáneamente.
* **Flexibilidad:** Permite crear efectos de animación y patrones personalizados con relativa facilidad, ya que se puede controlar cada LED individualmente.

**Retos:**

* **Programación Compleja:** Implementar el control multiplexado puede requerir una lógica de programación más compleja para determinar cuáles filas y columnas activar en cada ciclo de actualización.
* **Tiempo de Refresco:** Es importante seleccionar el tiempo de refresco adecuado para evitar el parpadeo visible de los LEDs y garantizar una apariencia continua.

Una implementación de esta alternativa sería así:

* Configurar las conexiones físicas: Conectar los LEDs de la matriz en filas y columnas, y conectar las filas y columnas a los respectivos pines digitales. Es fundamental tener en cuenta el esquema de conexión y la restricción de uso de pines.
* Se debe desarrollar funciónes de control: Crear una función que active secuencialmente las filas y columnas de la matriz para encender y apagar los LEDs. Esto se logra mediante bucles que alternan las filas y columnas en función del tiempo de refresco deseado; ello debe ser dependiente a las condiciones dadas en las imágenes propuestas para la actividad evitando la solución trivial.
* Generar patrones personalizados: Utilizar la función de control para mostrar patrones personalizados en la matriz, controlando cuáles LEDs se encienden en cada ciclo de actualización.
* Implementar funciones de usuario: Para interactuar con la matriz, desarrollar funciones que permitan al usuario seleccionar acciones a través del puerto serial, como mostrar patrones específicos o realizar verificaciones.

**ALMACENAMIENTO DE PATRONES EN ARREGLOS**

Esta alternativa se centra en almacenar patrones de LEDs en arreglos bidimensionales en la memoria del Arduino. Cada arreglo representa un patrón y contiene información sobre qué LEDs deben encenderse en cada fila y columna de la matriz de LEDs. Al programar el Arduino, se puede acceder a estos arreglos para mostrar patrones específicos en la matriz.

**Ventajas:**

* **Flexibilidad en la Creación de Patrones:** Esta técnica permite crear patrones personalizados con facilidad. Los patrones se definen como matrices de LEDs y se pueden modificar o ampliar según sea necesario.
* **Eficiencia en el Uso de Recursos:** En comparación con otras técnicas de generación de patrones, esta alternativa utiliza menos recursos computacionales, ya que los patrones se almacenan en memoria y no se generan en tiempo real.
* **Reutilización de Patrones:** Los patrones almacenados pueden reutilizarse en múltiples proyectos sin necesidad de reprogramación, lo que ahorra tiempo en desarrollos futuros.

Al igual que en la otra alternativa esta también nos ofrece retos a tener en cuenta en una posible implementación:

**Retos:**

* **Consumo de Memoria:** El almacenamiento de múltiples patrones en arreglos bidimensionales puede ocupar una cantidad significativa de memoria del Arduino, especialmente si los patrones son grandes o si se requiere un almacenamiento extenso.
* **Interfaz de Usuario:** Para que el usuario pueda seleccionar patrones y controlar su visualización, es necesario desarrollar una interfaz de usuario adecuada que permita la selección de patrones y control de tiempo.

# **ESQUEMA DE TAREAS**

Por cada una de las dos alternativas planteadas, elaboramos un esquema muy general, pero que nos será de gran ayuda, para organizar las tareas que se deben realizar en cada alternativa de solución.

**Alternativa 1: Control Multiplexado de LEDs**

|  |  |
| --- | --- |
| **Tarea** | **Descripción Detallada** |
| **Configuración de Conexiones** | - Establecer conexiones físicas entre los 64 LEDs y los pines digitales del Arduino siguiendo un esquema específico. Esto incluye conectar las filas y columnas de la matriz a los pines correspondientes. |
| **Desarrollo de la Función de Control** | - Crear una función que controlará de manera secuencial las filas y columnas de la matriz de LEDs. Esta función deberá activar cada fila y columna en un ciclo, encendiendo y apagando los LEDs correspondientes según el patrón deseado. |
| **Generación de Patrones Dinámicos** | - Diseñar algoritmos que generen patrones de LEDs de manera dinámica. Los algoritmos deben definir qué LEDs se encienden y apagan en cada ciclo de actualización para crear patrones visuales variados. Esto puede implicar el uso de bucles y estructuras de datos para almacenar los patrones. |
| **Función de Usuario (Control Serial)** | - Implementar una función que interprete comandos enviados por el usuario a través del puerto serial. Estos comandos pueden incluir acciones como mostrar un patrón específico, realizar verificaciones de LEDs o ajustar los tiempos de encendido y apagado. |
| **Optimización de Tiempos de Refresco** | - Ajustar los tiempos de encendido y apagado de los LEDs para evitar el parpadeo perceptible por el ojo humano. Esto implica determinar el intervalo de tiempo óptimo para cada ciclo de actualización. |

**Alternativa 2: Almacenamiento de Patrones en Arreglos**

|  |  |
| --- | --- |
| **Tarea** | **Descripción Detallada** |
| **Definición de Arreglos de Patrones** | - Crear arreglos bidimensionales que representen patrones de LEDs. Cada arreglo contendrá información sobre qué LEDs se encienden y apagan en cada fila y columna para un patrón específico. Estos arreglos se definen previamente y pueden ser de diferentes tamaños y formas para representar diversos patrones. |
| **Interfaz de Usuario** | - Diseñar una interfaz que permita al usuario interactuar con el sistema. Esto puede implicar la creación de un menú en el monitor serial donde el usuario pueda seleccionar patrones, ajustar tiempos de visualización y enviar comandos. La interfaz debe ser intuitiva y facilitar la selección de acciones. |
| **Programación de Arduino** | - Desarrollar algoritmos que permitan al Arduino acceder a los arreglos de patrones y mostrarlos en la matriz de LEDs. Esto implica la activación secuencial de filas y columnas según los datos del patrón seleccionado, así como el control del tiempo de visualización. La programación también debe interpretar los comandos del usuario y actuar en consecuencia. |

# **PROBLEMAS DE DESARROLLO QUE AFRONTÓ.**

Durante el desarrollo de un proyecto como este, podrías enfrentar varios problemas, como:

* **Errores de sintaxis y declaración de variables:** Los errores de sintaxis son comunes en la programación y pueden ser difíciles de detectar. Tuvimos que asegurarnos de que todas las variables estén declaradas y utilizadas correctamente e para evitar estos problemas.
* **Problemas de conexión física:** La conexión incorrecta de componentes puede ser un problema importante. Verificar las conexiones físicas es crucial para garantizar que los LEDs se enciendan y apaguen como se espera.
* **Problemas de lógica:** Los errores en la lógica del programa pueden causar resultados inesperados. Es importante revisar y depurar el código para garantizar que las condiciones y bucles se ejecuten correctamente.
* **Problemas de temporización:** Los tiempos de encendido y apagado de los LEDs deben ser precisos para lograr los patrones deseados. Un error en la temporización puede hacer que los patrones se vean incorrectos.
* **Problemas con la matriz de LEDs:** Asegurarse de que la matriz de LEDs esté conectada correctamente y que los LEDs funcionen correctamente puede ser un desafío. Los LEDs dañados o conexiones defectuosas pueden causar problemas en la visualización.

# **EVOLUCIÓN DE LA SOLUCIÓN Y CONSIDERACIONES PARA TENER EN CUENTA EN LA IMPLEMENTACIÓN**

La evolución de la solución incluyó entre otros aspectos los siguientes:

* **Optimización del código:** A medida que el proyecto creció, se volvió importante optimizar el código para reducir el uso de memoria y mejorar la eficiencia. La reutilización de funciones y la eliminación de código duplicado nos fue muy útil.
* **Interfaz de usuario mejorada:** Esto fue todo un reto hacerlo, pero sabemos que con una interfaz mejorada se hace mucho más fácil el uso de programa, valiéndose de la intuición del usuario.
* **Manejo de errores:** Se hizo necesario un manejo de errores robusto. Es decir, hubo que lidiar con situaciones inesperadas, como una conexión de hardware defectuosa o entradas incorrectas del usuario. Esto particularmente nos tomó gran parte del tiempo, nosotros desarrollamos la lógica del código en Qt y luego hubo que pasarla para Tinkercad para Arduino, lo que implicó un reto para nosotros.
* **Pruebas exhaustivas:** Hubo que realizar muchas pruebas del circuito implementado como del código que se hizo. Esto con el fin de optimizar y/o mejorar lo que se estaba haciendo. Aquí nos encontramos con situaciones muy comunes en la implementación de circuitos como corrientes muy altas para los componentes y componentes no conectados.

# **BIBLIOGRAFÍA**

<https://stackoverflow.com/questions/63331870/how-do-i-use-an-arduino-as-an-led-controller>

<https://programarfacil.com/blog/arduino-blog/matriz-led-arduino-max7219/>

[Explicación: Arrays en programación | Arduino Secuencia de LED's.](https://www.youtube.com/watch?v=80tCUyfmwF0&t=53s)

[](https://www.youtube.com/watch?v=80tCUyfmwF0&t=53s)