**Parcial#1- Juan David Cárdenas y Samuel Di Nicola**

**Links:**

* **Video:** https://youtu.be/8BMl453E6I0?si=r5c51LYeV7AR\_1xZ
* **Tinkercad:** <https://www.tinkercad.com/things/0yv24uUowD2-parcial-1-info-ii/editel?sharecode=3MCif6SJYjiq0OOrcRstLz9UMXmuDb-cKlRUL5rPmtI>

**--------------------------------------------------------------------------------------------------------------**

**Análisis del problema.**

Después de una lectura y compresión individual al documento del primer parcial, los integrantes del grupo se reunieron para discutir el documento y cómo afrontar los problemas.

Tinkercad: Sobre esta página en la cual es donde se llevará a cabo la simulación del sistema principal, se discutió dos problemáticas:

1) Como hacer los repositorios en Github: se propuso y al momento de esta redacción es lo que se llevara a cabo es, que en un archivo tipo .txt se pegara los comandos creados Tinkercad, para así poder llevar a cabo el repositorio.

2) Si era posible trabajar de forma cooperativa: se encontró la opción después de una ligera búsqueda se encontró dicha opción y se le aplico al proyecto de Tinkercad.

3) Si era posible hacer una clonación del proyecto (si es posible), esta duda surgió ya que era posible que mediante pruebas de código se dañara las conexiones de alguna forma, por ende, se acordó que cada uno trabajase en un archivo clonado y que si probando el programa se encontraba algún problema con el cableado esto debía de informarse de inmediato para ser corregido, además de crear una nueva clonación.

Queda aclarar que al momento de la reunión solo se tenía conocimiento básico de Tinkercad.

Video: no se ha planeado nada todavía, debido a que aún no se había resuelto el problema.

Sobre la matriz se recalcó que se debía crear la matriz hecha totalmente de leds, los cuales al ser puestos debían cablearse al instante, para si evitar confusión más adelante cuando llegase el momento de programar.

Para el tema de las funciones, se hablará en paralelo de otras, para así ilustrar de mejor manera la idea que se tiene.

Para la función verificación, se recalcó dos cosas:

1) Que no necesariamente se deben de encender los leds al mismo tiempo todos, por ende, un encendido secuencia es posible.

2) Que el usuario puede manipular el tiempo que transcurrirá entre el momento en que los leds se enciendan y se apaguen, además de decir cuantas veces se debe repetir la comprobación.

Para el primer objetivo de la función verificación se descartó la opción de apagado y encendido de forma secuencial, ya que si se usa un 74HC595 se puede encender todos los leds al mismo tiempo si se cargan de antemano la salida, hacer lo mismo para el apagado, aunque aún no se ha probado, pero se conoce un poco sobre el funcionamiento de este integrado debido a una búsqueda en YouTube y Google, por ende se sabe que es viable hasta cierto punto, ya que no se permite soluciones triviales, no se puede tener dos arreglos ya cargados con los estados. Pero algo que no se cambio fue la necesidad de los dos arreglos uno para cada estado, los cuales se cargarán al momento de llamar la función, estos arreglos no serán globales. La forma que se cargaran por el momento es por medio del uso de ciclos.

Para el segundo tema, se piensa que estas opciones sean globales, para poner todo de la función verificación dentro de un while, para que cuando el contador termine se salga y vuelva al menú “principal” que sería la función publik.

Para la función de los patrones fue algo que se a bordo de forma ligera ya que aún no se ha acoplado ni determinado los patrones, lo que se dijo fue: los patrones son simétricos, por ende, basta con hacer la mitad, y así terminar de formar el patrón en su totalidad; se conectara en cascada alrededor de 4 o más (lo que sea necesario) 74HC595, los cuales tiene la peculiaridad de que el primer “bit” que sea ingresado terminara en la posición 8x8 y el ultimo en la 1x1, según se vio en una demostración, por ende, hay que tener en cuenta eso.

Se piensa ingresar funciones dentro de esta función, las cuales contendrá de forma individual los patrones, para así revisar mejor el código y poder usar cada patrón de forma individual, aun no sea ha concretado.

Se dividió los patrones, de tal forma que a cada participante le toque dos.

Función publik: Esta es considerada el menú de inicio en las cuales se presentarán 4 opciones, queda aclarar que esta función estará dentro de la función loop para que se muestre cada vez que se complete o termine algunas de las opciones seleccionadas y en esta función es posible que estén las variables que se usaran para las demás.

Las cuatro opciones son las siguientes (si no se han mencionado se dará a conocer de forma breve lo que se planea hacer):

1). La función verificación de leds.

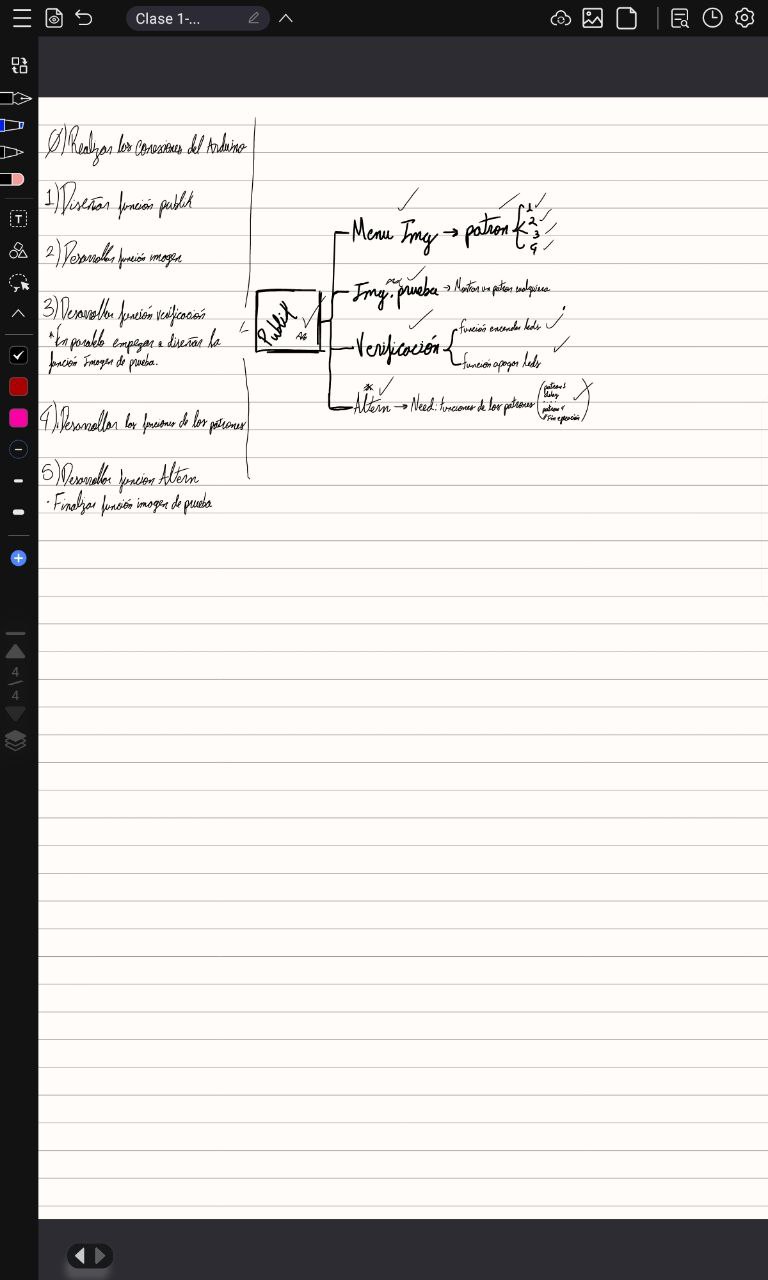
2). Imagen de prueba: En esta opción siempre saldrá el mismo patrón, sin importar cuantas veces se seleccione, además el usuario podrá decir la duración del estado encendido y el de apagado, es posible que el patrón se muestre un par de veces más antes de mostrar a publik nuevamente (al momento de escribirse se tomó como lo más probable)

3). Alternar los patrones: se mostrará en secuencia los mismo cuatro patrones en el mismo orden sin importa las iteraciones realizadas, en este no habrá momento en que los leds estén apagados (a menos que en el patrón sea así), también estará en un while, para que muestre nuevamente el publik.

4). En esta opción estará la función imagen.

Por el momento esta es la forma en la que se desarrollara el trabajo, la información de este documento puede cambiar en su totalidad, todo dependiendo del desarrollo del proyecto.

**Esquema de desarrollo:**



**Algoritmos implementados:**

|  |  |
| --- | --- |
| Funciones | Descripción |
| imagen () | Menú en donde se seleccionan los patrones a imprimir en la matriz |
| verificación () | Permite encender y apagar los leds, además recibe del usuario el tiempo que hay entre apagado y encendido y las veces que se repetirá. |
| Apagado () | Función que apaga todos los leds de la matriz |
| encendido () | Función que enciende todos los leds de la matriz |
| patron\_1() | Función que genera el patrón 1 |
| patron\_2() | Función que genera el patrón 2 |
| patron3() | Función que genera el patrón 3 |
| patron4() | Función que genera el patrón 4 |
| imagen prueba () | Función que toma un patrón predefinido y lo imprime con intervalos de tiempo ingresadas por el usuario |
| Altern () | Alterna entre los 4 patrones en orden con un delay ingresado por el usuario |

**Problemas en el desarrollo:**

Inicialmente la principal dificultad fue averiguar el funcionamiento del integrado 74HC595, viendo cómo funcionaba la configuración del reloj, la forma en que se ingresarían los datos y la forma de hacer que se muestren dichos datos por la matriz. Probamos distintas funciones que estaban relacionadas con el integrado tales como shiftout () o shiftin (), pero a pesar que entendimos en cierta medida su funcionamiento, no supimos como finalizar su implementación para la resolución de los problemas planteados, entonces lo que hicimos fue realizar las manipulaciones de los registros manualmente, es decir, las activaciones del reloj, el latch y las entradas por medio del método “digitalWrite ()” y ciclos.

Otro problema que se nos presento fue el hecho de que en la función “verificación” al enviar la señal para encender todos los leds, los integrados se sometían a una corriente superior a la que soportaban, haciendo que, a pesar que si se enciendan los leds, los integrados no permitían seguir el flujo de la ejecución. Lo que tuvimos que hacer fue variar el valor de las resistencias, porque intentamos utilizar la salida de voltaje de “3.3V” del Arduino, pero el problema se seguía presentando, entonces concluimos con variar los valores de las resistencias en un punto en que los integrados no se perjudicaran y que los leds obtuvieran suficiente luminosidad.

Una dificultad que tuvimos fue, de qué forma construiríamos la estructura de datos para variar los valores de los leds. Concluimos que la mejor manera seria que en vez de usa una matriz 8x8 para todo el problema sería utilizar arreglos individuales y especializados para cada patrón, así permitiendo un desarrollo más cómodo y adaptado a la lógica planteada para cada uno de los patrones, esto debido a las peculiaridades que cada uno presentaba.

**Evolución de la solución y consideraciones de la implementación:**

Una de nuestras principales dudas al inicio fue el hecho del manejo del código, porque ya que Tinkercad no trabaja con archivos si no en la nube, el control iba a ser muy indirecto, pero lo que nosotros decidimos fue manejar el control del código por medio de archivos de texto donde copiábamos el código y lo guardábamos en el repositorio. Y la pregunta seria ¿Por qué no desarrollar primero el código en Qt y luego hacer un port a Tinkercad y adaptarlo al Arduino? Porque sería más esfuerzo del necesario ¿Cuál fue nuestra idea? Adaptar la lógica que pudimos haber creado directamente en Qt, a Arduino directamente, para realizar de una vez toda la programación directamente sobre las funcionalidades de Arduino, haciendo que también nos viéramos a la tarea de desarrollar primero la circuitería para evitar posibles conflictos entre la lógica y las funcionalidades disponibles en Arduino.

A medida del desarrollo nos fuimos dando cuenta de las peculiaridades que se nos presentaban, inicialmente en las conexiones. Decidimos plantear una conexión en cascada de los leds, basándonos en un diseño en tinkercad en el cual se tenían una fila de leds conectados a tres integrados **74HC595** en donde se configuro para prenderlos de forma secuencial, decidimos basarnos en eso para armar cada fila de la matriz haciendo que nuestras entradas vayan ingresando de forma secuencial el valor atribuido a cada led, esto con el fin de tener un mejor control de los mismos y los valores que ingresaban a la matriz.

A continuación, empezamos con la construcción de la función imagen, en donde se nos presentó el hecho de hacer inputs por el monitor serial, al inicio utilizamos el método “Serial.read()” pero sucedía que daban valores los cuales no coincidían con los ingresados, y lo que sucedía era que tomaba los valores ingresados como tipo ”char” pero debido a que la variable de control estaba definida como entero se realizaba una conversión de entero a char, y luego de char a entero dando valores distintos a lo ingresado por el usuario, entonces utilizamos el método “Serial.parseInt()” el cual, sí toma directamente el valor ingresado como entero, así permitiendo una relación más directa entre lo que establecíamos para el “switch” y lo que ingresaba el usuario, la única peculiaridad es que solo toma valores enteros entonces no toma o reconoce entradas que no sean enteros, retornando un valor de 0 en dicho caso, permitiéndonos así un mejor control de los menús desarrollados.

Principalmente lo que cambio en nuestro desarrollo fue la forma en la que íbamos a manejar la estructura de los datos, es decir, la forma en la que como íbamos a configurar los leds. Lo que decidimos en vez de realizar una sola matriz única para generar todos los patrones en base a ella fue usar arreglos personalizados para cada patrón, refiriéndome no a que utilizar arreglos preestablecidos para cada uno, sino a que dependiendo de la peculiaridad del patrón adaptar un arreglo a el mismo. Por ejemplo, en el patrón 4 que se puede ver notoriamente el desplazamiento de una línea de 4 leds encendidos contiguos, en vez de ir iterando sobre una matriz e ir replicando y modificando un elemento, se usó un solo arreglo, con 2 punteros asociados, haciendo en la medida que se necesitaba se modificaban los valores del arreglo por medio de los punteros y estos mismos se desplazaban sobre el mismo para mostrar ese desplazamiento en el patrón. Y así con la misma filosofía aplicada a los 4 patrones, cada uno con su propia necesidad y variabilidad distinta.

Finalmente hay algo que aclarar respecto a las funcionalidades utilizadas. Como se podrá ver en nuestro código, las variaciones del reloj, del latch y las entradas de datos se realizaron de forma manual con “digitalWrite ()” aun teniendo el método “shiftout ()” el cual no utilizamos por la razón de que no logramos comprender su uso del todo. Intentamos experimentar con él al inicio y descubrimos las peculiaridades que albergaba, más que todo en la forma en la que se le debe indicar para encender los leds, pero a la final vimos inviable este camino por la complejidad. Siendo más ameno el desarrollo por la vía que implementamos, haciéndola más intuitiva de comprender en cierta medida.