UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Cartel Led Programable

Trabajo práctico integrador

Grupo N° 2

Materia: Técnicas digitales II
Profesor: Ing. Juan Alarcón
Ayudante: Ing. Marcelo Crivelli

Integrantes	Legajo
Jonathan Yujra	163647-9
Ovidio Argani	137715-2
Cristian Torres	151015-0

Índice

1. Descripción General

El proyecto se basa en un cartel iluminado programable por el usuario, que contará con distintas funcionalidades y con diferentes tecnologías. El cartel contará con 256 leds que se iluminarán de acuerdo a las peticiones del usuario.
El usuario podrá:

- Establecer y mostrar la fecha
- Establecer y mostrar la hora
- Mostrar la temperatura del ambiente
- Establecer, mostrar y guardar un mensaje.
- Variar la velocidad del mensaje mostrado por el cartel

Para ello, el usuario podrá comunicarse con el cartel a través de comunicación por Bluetooth o con pulsadores que formarán parte del dispositivo.

Entre las tecnologías con las que contará el cartel se encuentran:

- Comunicación Bluetooth
- Reloj en tiempo real
- Sensor de temperatura
- Tarjeta de memoria en formato SD
- Sensor de intensidad lumínica

1.1. Introducción

Los carteles siempre han formado y formarán parte de la sociedad. Su utilidad para enunciar un evento, alertar sobre algún peligro, informar alguna novedad son de extrema necesidad para los habitantes que pueden recurrir a ellos de forma práctica y casi al instante.

Para el proyecto de Técnicas Digitales II se propuso la creación de un cartel programable que pueda ser utilizado para satisfacer las necesidades básicas de un usuario, como conocer la fecha y hora actual, la temperatura en el ambiente, o para enunciar a través de él algún mensaje para alertar o informar alguna novedad.

1.2. Interfaz con el usuario

Se buscó que la interacción entre el usuario y el dispositivo sea la más práctica y cómoda posible y por eso se recurrió a una comunicación ultra instalada en el ambiente tecnológico como es el bluetooth. El usuario a través de una aplicación podrá conectarse al dispositivo e interactuar con él, ya sea para establecer algún tipo de dato (como la hora y fecha actual o

algún mensaje) como también para poder pedirle al cartel que muestre esos datos, la temperatura o algún mensaje.
También se tuvo en cuenta un usuario más acostumbrado a una comunicación más analógica como pueden ser unos pulsadores. Los mismos estarán para cumplir las exigencias mínimas de las personas, como mostrar hora, fecha, temperatura o los mensajes.
El cartel estará conformado por 256 leds ubicados en forma de matriz(8x32) que mostrarán el mensaje de derecha a izquierda.
El cartel también cuenta con un sensor de luz que hará de acompañante visual, ya que elevará la iluminación del cartel si se encuentra en un ambiente demasiado iluminado (exteriores) como también reducirá la iluminación del mismo si el ambiente se encuentra oscuro (lugares cerrados, subsuelos).

1.3. Operación

El modo de operación es sencillo. El usuario a través de una aplicación podrá comunicarse con el cartel y enviar datos o mensajes que se reproducirá en el dispositivo. Tendrá una opción usuario normal y una pensada para trabajos de mantenimiento, donde a través de unas macro se podrán testear diferentes funciones y controlar el buen funcionamiento del mismo.

El usuario también podrá optar por usar unos pulsadores que forman parte del dispositivo para también comunicarse con él.

2. Hardware

Para la construcción del cartel se utilizó una placa física donde se interconectaron los diferentes sensores y módulos para el armado final del dispositivo. Entre los módulos y sensores utilizados se encuentran:

- 4x Matrix led de 8x8 puntos
- STM32 NUCLEOF401-RE
- Módulo Bluetooth HC06
- Sensor de luz BH1750
- Sensor de temperatura DHT11
- Lector de tarjetas SD

La placa física será alimentada con 5V y con un regulador de voltaje de 5V a 3,3V para alimentar algunos sensores.

2.1. Diagrama en Bloques

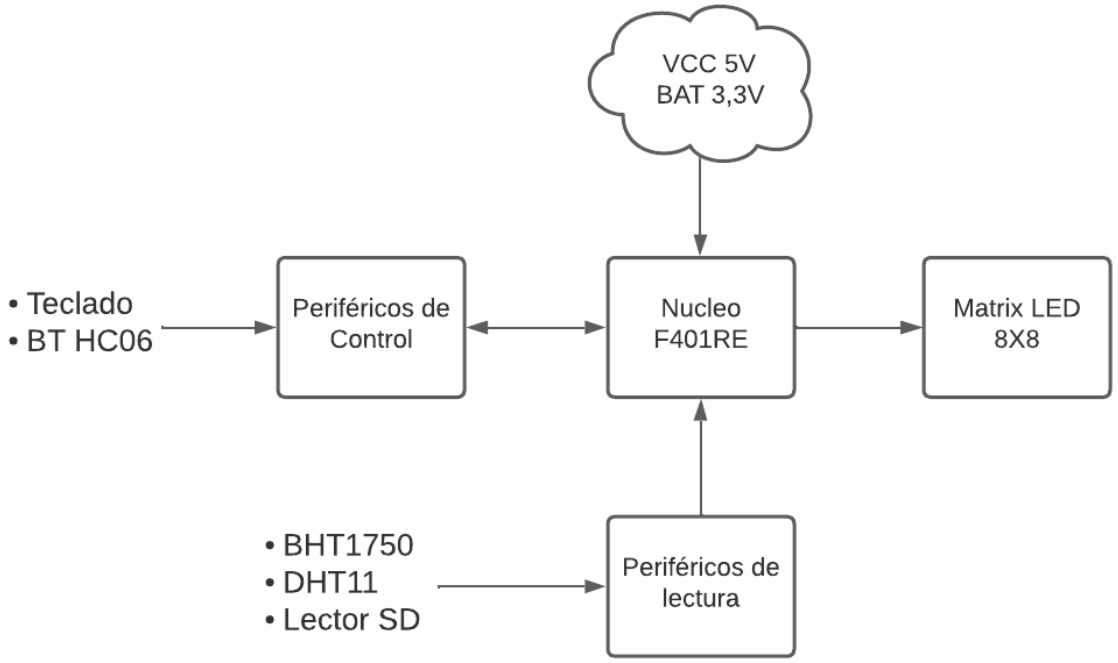

Fig. 1: Diagrama de bloques

2.2. Circuito esquemático

2.2.1 Alimentación

Alimentación 5V

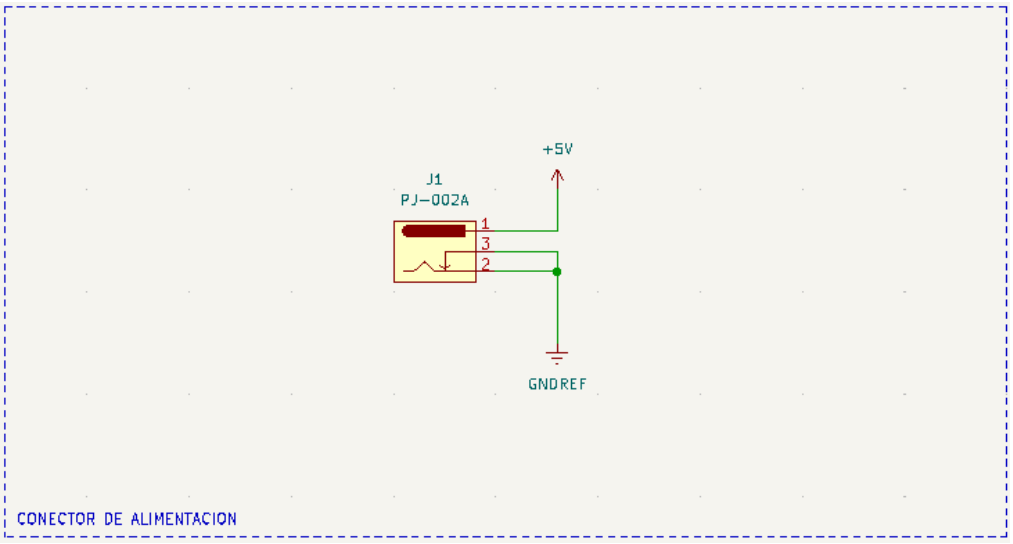

Fig.2: Conector de alimentación 5V

Regulador de voltaje de 5V - 3,3V

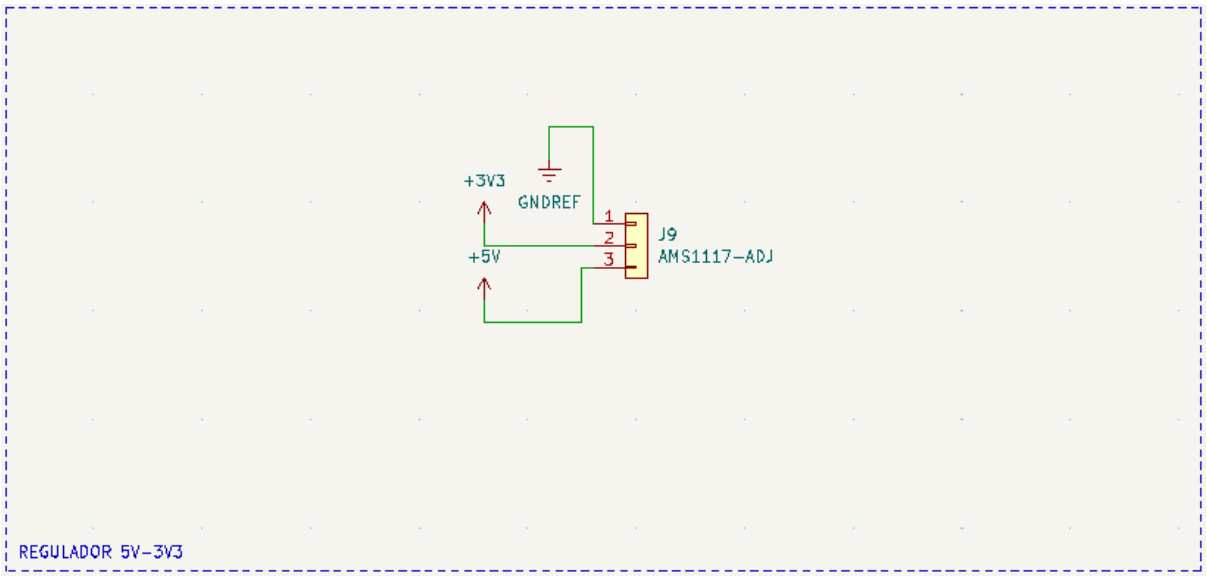

Fig.3: Regulador de voltaje 3,3V

Batería de 3,3V

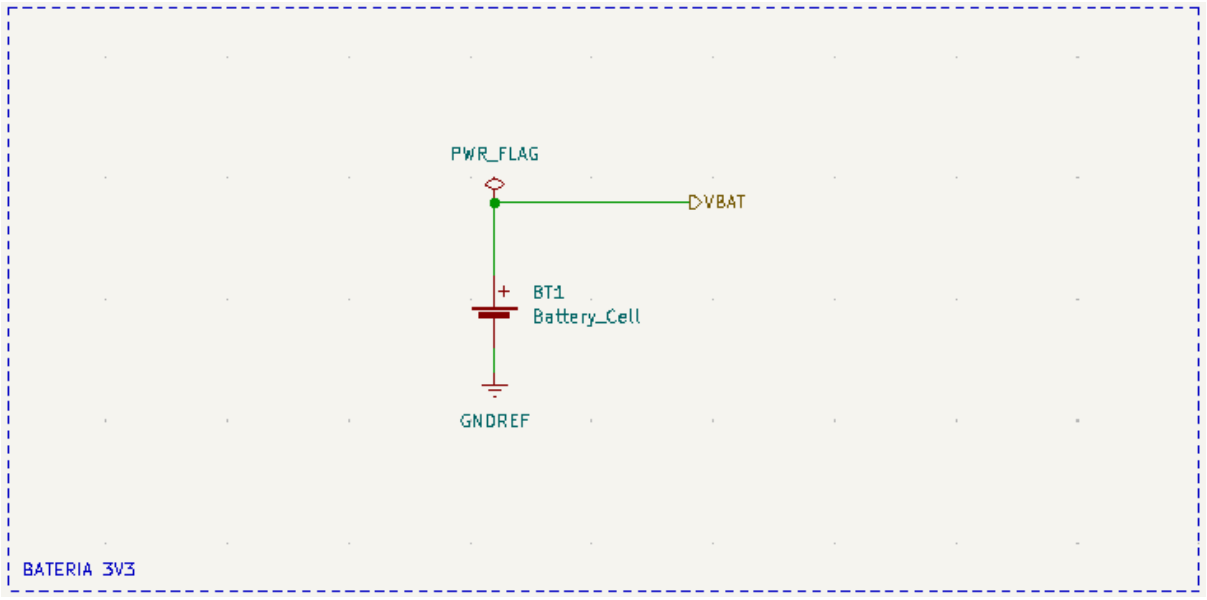

Fig.4: Batería CR2032 de 3V

2.2.2 STM32 NUCLEO F401RE

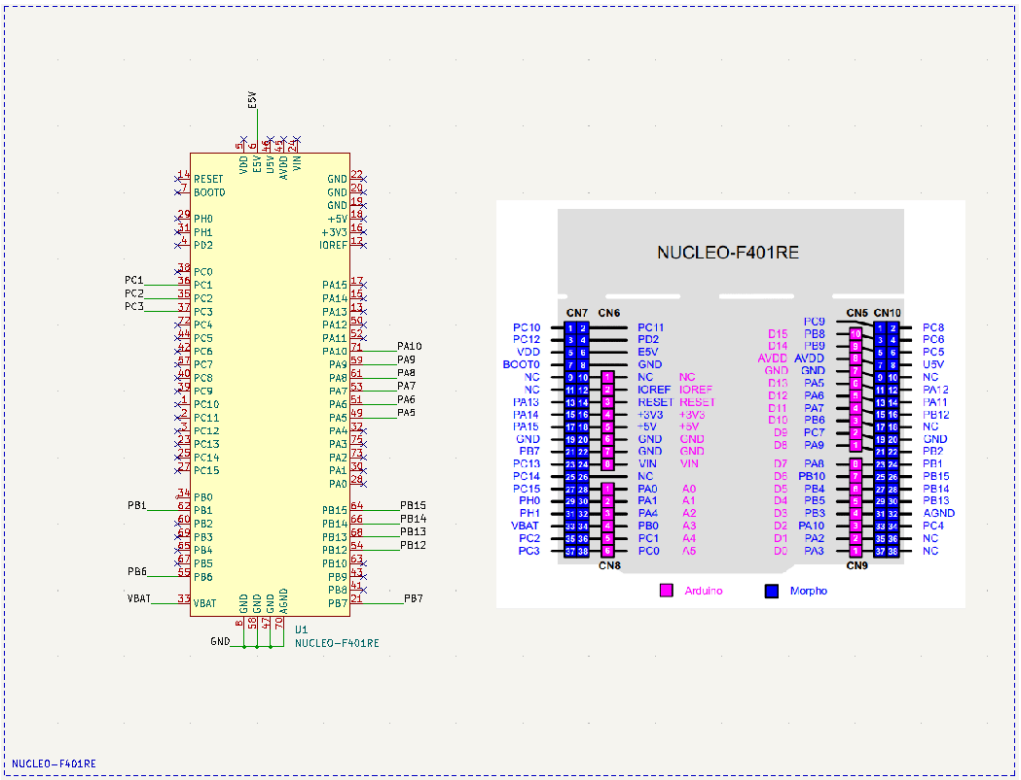

Fig.5: STM32 NUCLEO-F401RE

2.2.3 Entradas y Salidas

Nombre de los pines de E/S

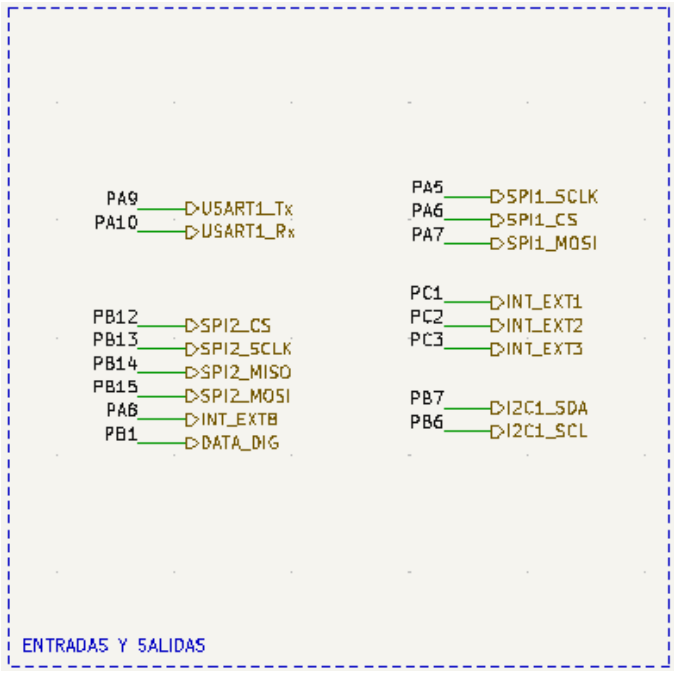

Fig.6: Nombre de los pines de E/S

Sensor de temperatura DHT11

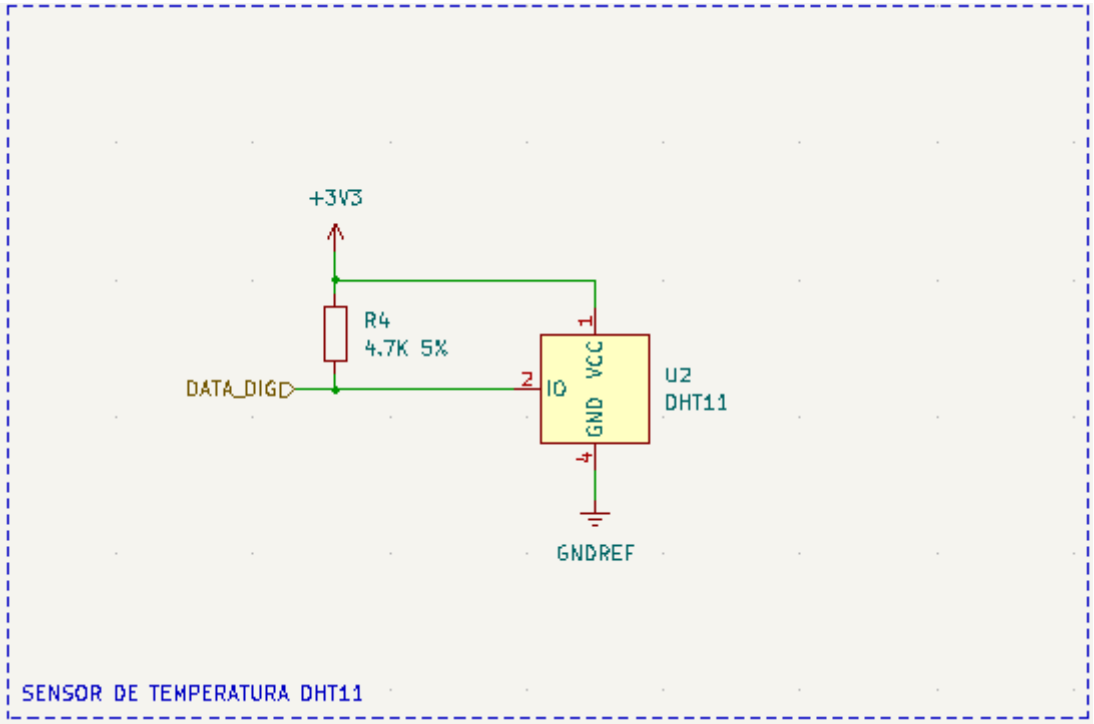

Fig.7: Sensor de temperatura DHT11

Módulo de Bluetooth HC-06

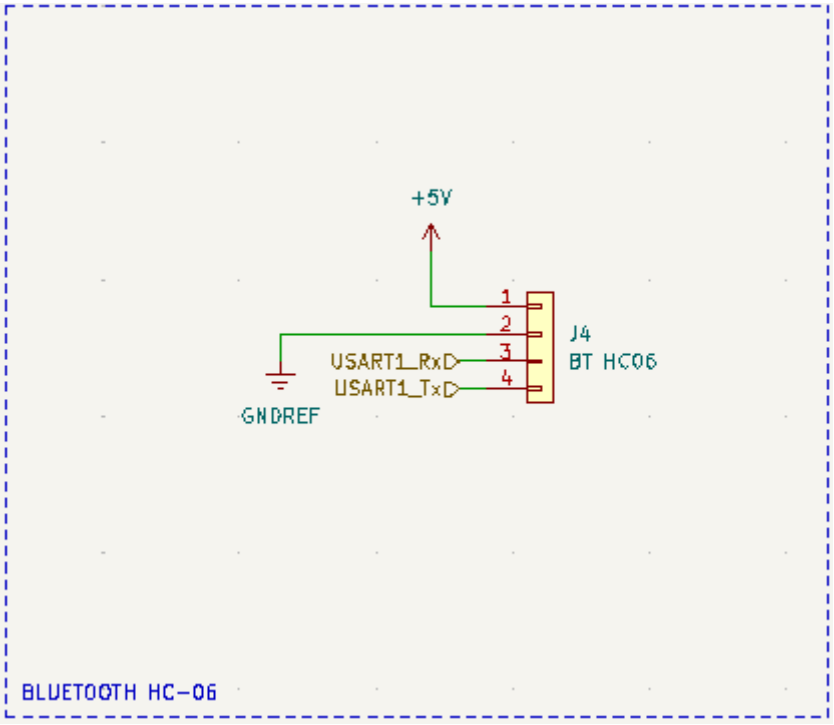

Fig.8: Módulo Bluetooth HC-06

Matriz LED 8x8

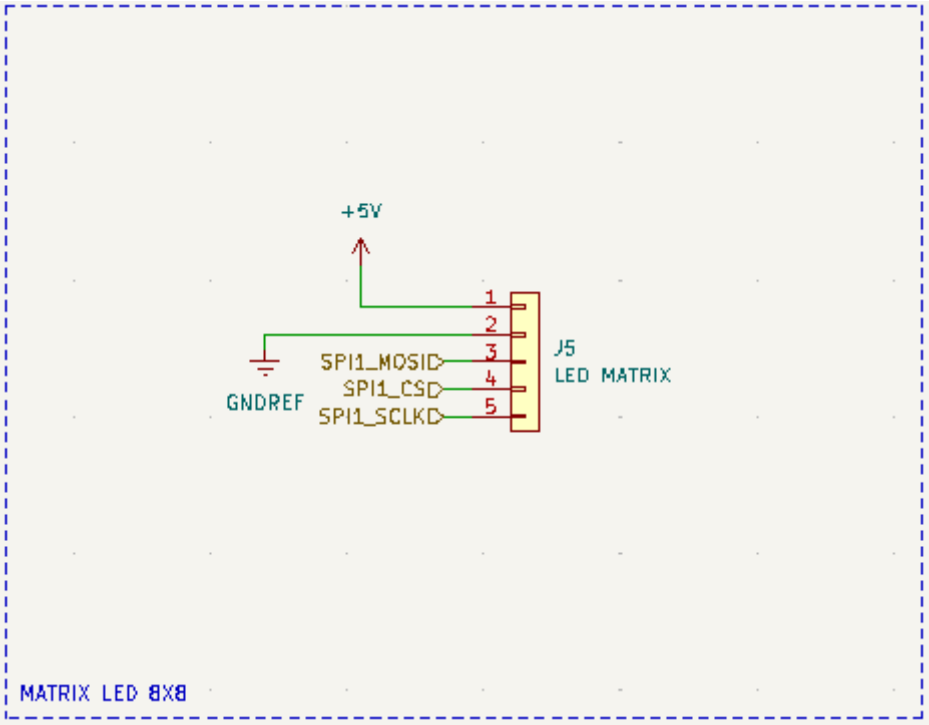

Fig.9: Matriz LED 8X8

Sensor de Luz BH1750

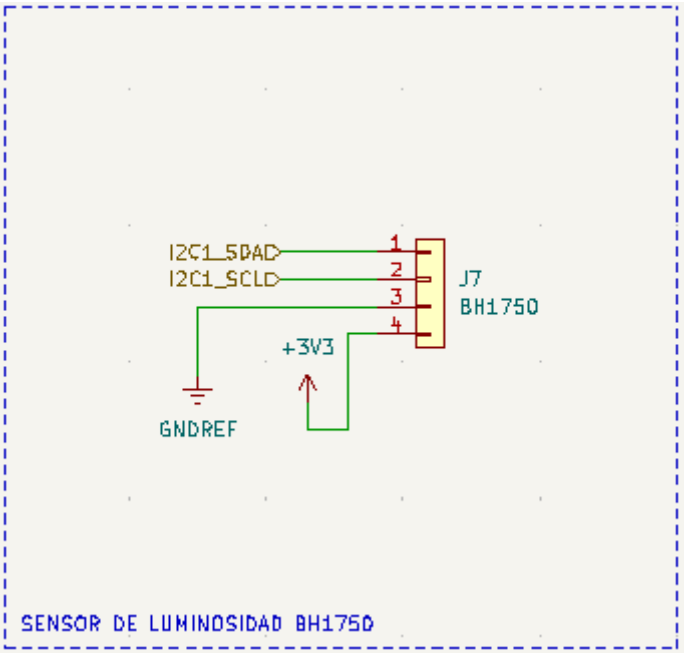

Fig.10: Sensor de luz BH1750

Lector de tarjeta SD

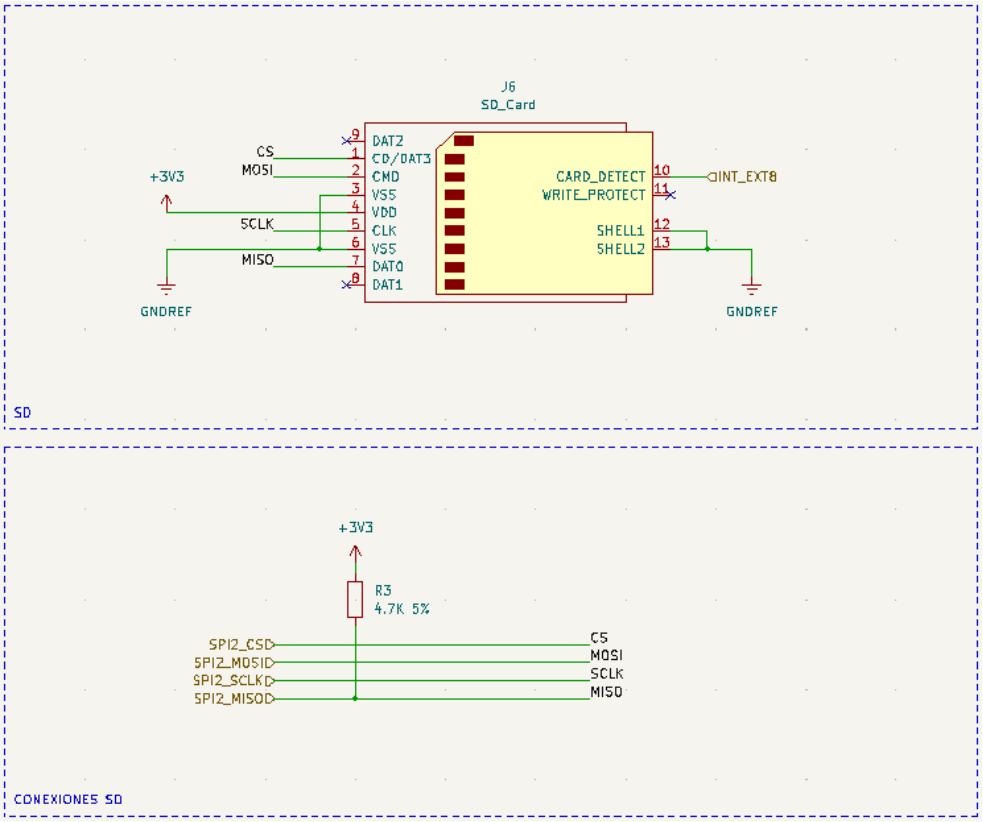

Fig.11: Lector de Tarjeta SD

Teclado

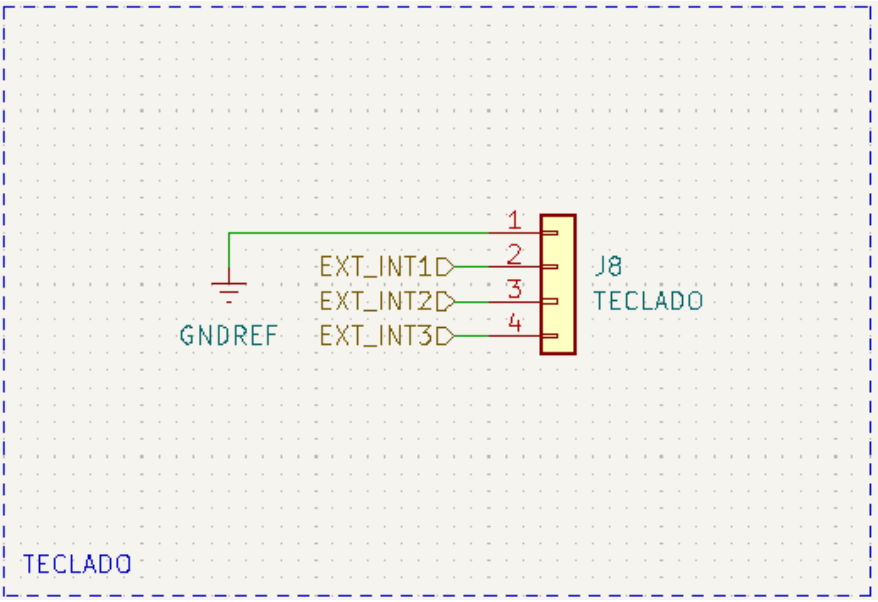

Fig.12: Teclado

2.3. Descripción del Circuito

El circuito está alimentado por una tensión continua de 5V y también cuenta con un regulador de voltaje de 5V a 3,3V. La NUCLEO F401RE además estará alimentada por una batería de 3,3V CR2032 que servirá para alimentar las funciones básicas de la placa como el reloj en tiempo real y que de ese modo no pierda la cuenta.

La distribución de la alimentación en los componentes es la siguiente:

Los componentes alimentados a 5V son los siguientes:

- NUCLEO F401RE
- BT HC-06
- 4x Matriz LED 8X8

Los componentes alimentados a 3,3V son los siguientes:

- Lector SD
- BH1750
- DHT11

2.4. Circuito Impreso

Diseño de PCB

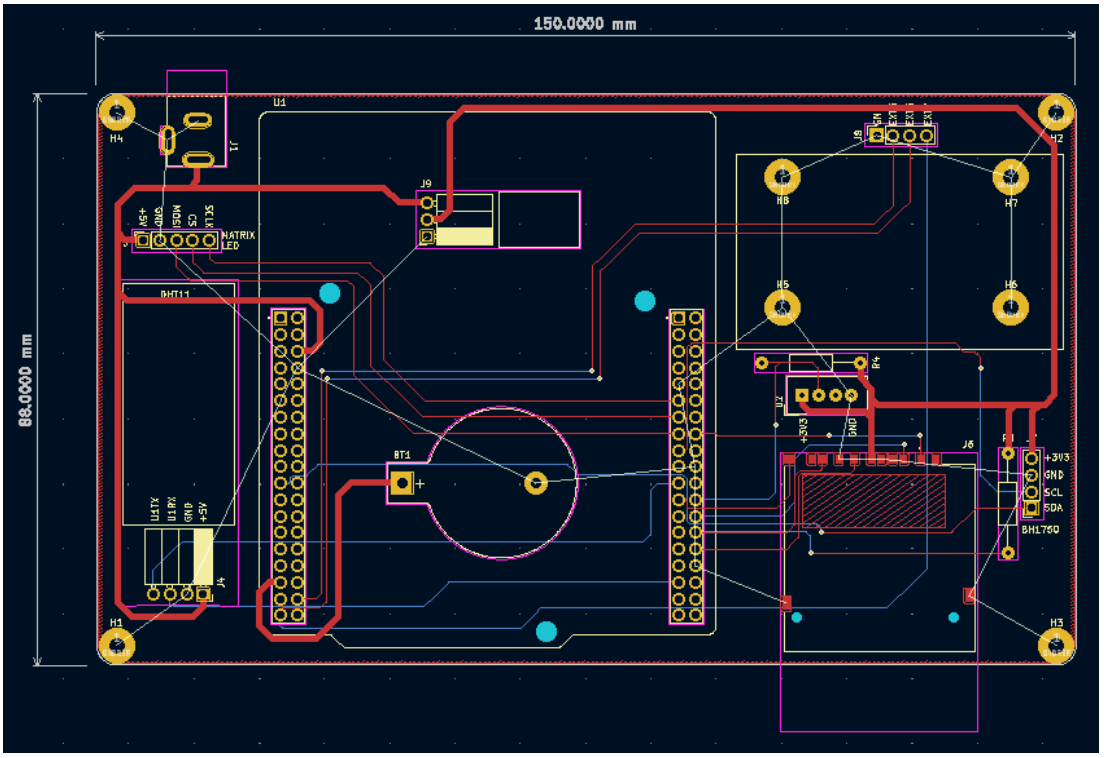

Fig.13: Diseño de PCB

Ubicación de los componentes en el PCB

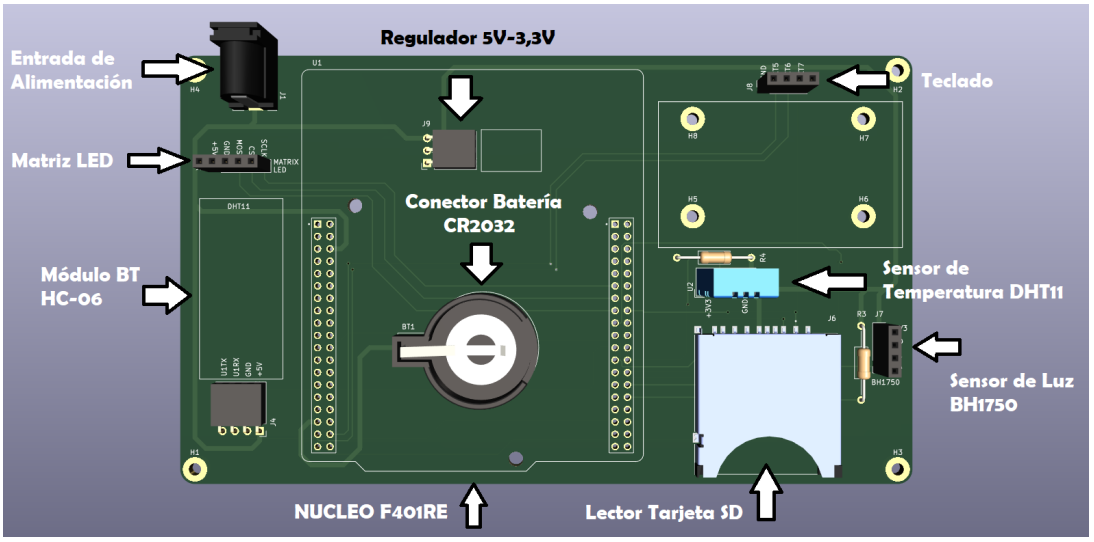

Fig.14: Ubicación de componentes

3. Software

3.1. Software Microcontrolador

En el proyecto se utilizaron los siguientes protocolos de comunicación:

- SPI para la comunicación con la SD y la matrix LED
- I2C para la comunicación con el sensor de luz BH1750 y el sensor de temperatura DHT11
- Comunicación Serie para la comunicación con el módulo de BT HC06.

Cada uno de los módulos tuvo sus pines dedicados para asegurar la correcta transferencia de información.

Para el manejo de tiempo se utilizaron timers y RTC. El RTC para guardar la fecha y hora en el micro, teniendo en cuenta de que estas no se pierdan en caso de que el micro sea desconectador la tensión de alimentación ya que está también alimentado por una batería externa.

Se utilizaron Timers para controlar:

- Sistema operativo (FreeRTOS)
- DHT11
- Rutina de la Matrix Led

El teclado cuenta con 3 botones que fueron configurados como interrupciones externas.

El diagrama principal se encuentra en el apartado 3.1.4.

3.1.1. Entorno de desarrollo

El proyecto fue enteramente desarrollado en el IDE que nos provee ST, el STM32CUBE IDE y el lenguaje de programación utilizado fue en su totalidad el C.

3.1.2. Sistema Operativo

Se utilizó un sistema operativo en tiempo real (FREERTOS) por su practicidad con los manejos de tarea ya que su planificador es del tipo preemtive, los que nos permite organizar las tareas según su prioridad.

El RTOS (y su planificador) cuenta con un timer dedicado a que se ejecute su planificación.

3.1.3. Librerías

DEEP BLUE

Se utilizaron librerías externas para el módulo de matriz Led 8x8 y el IC MAX7219, dichas librerías fueron realizadas para la gente de DEEP BLUE.

DOT_MATRIX.h

```
// The Number OF Separate DOT MATRIX Units (Whether Single or Cascaded) To Be Used In The Project
#define DOT_MATRIX_UNITS  1
// The Matrix Units Count Of The Longest Cascaded String In The System
#define MAX_CASCADED_NUM  4

// DOT Matrix SPI Options
#define DOT_MATRIX_SPI    SPI1

// DOT Matrix Timer Base Options
#define MATRIX_TIMER_EN   1
#define MATRIX_TIMER      TIM4
#define MATRIX_TIMER_CLK  72
#define MATRIX_TIME_BASE  1

// DOT Matrix Other Definitions
#define STATIC_MODE       0
#define SCROLL_MODE       1
#define MATRIX_DISPLAY_UNIT1  0
```

Fig.15: Bloque de código DOT_MATRIX.h

En la imagen de arriba se observan los diferentes defines donde se configura la cantidad de módulos en cascada, el número de SPI de la placa NUCLEO y el timer de interrupción.

```
// DOT MATRIX Configuration Parameter Structure
typedef struct
{
    GPIO_TypeDef * SS_GPIO;
    /*
     * GPIOA or GPIOB ...
     * */
    uint16_t       SS_PIN;
    /*
     * GPIO_PIN_0 or GPIO_PIN_1 ...
     * */
    uint16_t       SCROLL_SPEED;
    /*
     * Any Value [0-65535]
     * The Bigger This Number, The Slower Scrolling Becomes
     * */
    uint8_t        CASCADED_DEVICES;
    /*
     * Any Number of Devices [ 1 - 255 ]
     * */
    uint8_t        BRIGHTNESS;
    /*
     * Any Value [ 0 -> 15 ]
     * */
    uint8_t        SCROLL_Mode;
    /*
     * STATIC_MODE or SCROLL_MODE
     * */
}DOT_MATRIX_CfgType;
```

Fig.16: Struct de configuración de los módulos Matriz Led

Dentro de la estructura DOT_MATRIX_CfgType se pueden administrar la velocidad de corrimiento, la iluminación y los pines dedicados.

Las prototipos de funciones de la librería encargada administrar la funciones internas de del módulo son las siguientes:

```
// Init Functions Without & With Dedicated Timer
void DOT_MATRIX_Clear(SPI_HandleTypeDef * hspi);
void DOT_MATRIX_Brightness(uint8_t b);
void DOT_MATRIX_Init(SPI_HandleTypeDef * hspi);
void DOT_MATRIX_Init_TMR(SPI_HandleTypeDef * hspi, TIM_HandleTypeDef* TMR_Handle);
// STATOC MODE Functions
void MATRIX_CLEAR(uint8_t au8_MATRIX_Instance);
void MATRIX_Write_Char(uint8_t au8_MATRIX_Instance, uint8_t myChar);
// SCROLL MODE Functions
void DOT_MATRIX_Main(void);
void MATRIX_TMR_OVF_ISR(TIM_HandleTypeDef* htim);
void MATRIX_SCROLL_SetSpeed(uint8_t au8_MATRIX_Instance, uint16_t au16_SPEED);
void MATRIX_DisplayMessage(uint8_t au8_MATRIX_Instance, char* ArrayPointer, uint16_t ArraySize);
```

Fig.17: Prototipos de funciones utilizados en la librería

En la siguiente variable del tipo struct perteneciente a DOT_MATRIX_cfg.c se define el pin de Chip Select, la velocidad inicial (en milisegundos) y el brillo por defecto.

```
const DOT_MATRIX_CfgType DOT_MATRIX_CfgParam[DOT_MATRIX_UNITS] =
{
    // DOT_MATRIX Display Unit1 Configurations
    {
        GPIOA,
        GPIO_PIN_6,
        100,   /* Scroll Speed*/
        4,    /* Number Of Cascaded Devices*/
        3,    /* Brightness Level */
        SCROLL_MODE
    }
};
```

Fig.18: Configuración Default de los módulos Matriz Led

Los enlaces a las librerías se encuentran en el apartado de referencias.

Alexander Hoffmanes

La librería externa del BH1750 creada por Alexander Hoffmanes, es sencilla. De ella usamos la inicialización del sensor de luz digital:

```
HAL_StatusTypeDef BH1750_init_dev(BH1750_device_t* dev)
{
    BH1750_send_command(dev, CMD_POWER_ON);
    BH1750_send_command(dev, CMD_RESET);
    BH1750_send_command(dev, CMD_H_RES_MODE);

    return HAL_OK;
}
```

En donde se envían comandos por I2C para configurar el sensor a una resolución de 1 lux.

Luego para las mediciones usamos la función:

```
HAL_StatusTypeDef BH1750_get_lumen(BH1750_device_t* dev)
{
    BH1750_read_dev(dev);
    BH1750_convert(dev);
    return HAL_OK;
}
```

La cual lee el valor sensado por el módulo BH1750, luego lo convierte a unidades de lux.

FatFS - Generic FAT Filesystem Module

FatFs es un módulo de sistema de archivos FAT/exFAT genérico para pequeños sistemas integrados. Está escrito totalmente en C y en una capa completamente separada de la capa de E/S del disco. Fue desarrollada por chaN.

FatFs controla los dispositivos de almacenamiento a través de una sencilla interfaz de acceso a los medios que se muestra a continuación.

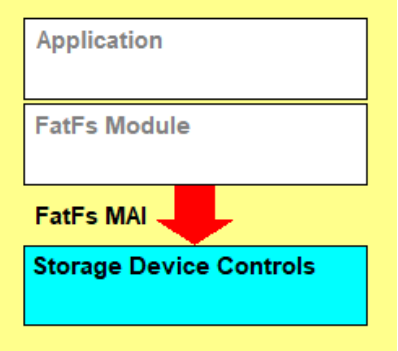

Fig.19: Abstracción de la librería

Las librerías que incluye el paquete son:

- **diskio.h:** Encargada de la interfaz de disco de bajo nivel (E/S)
- **ff.h:** módulo de archivos de sistema FAT
- **ffconf.h:** Configuraciones funcionales FatFs

Los archivos .C que incluye el módulo son:

- **ff.c:** código del módulo de archivos de sistema FAT
- **ffsystem.c:** código de funciones dependientes del SO para FATFs
- **ffunicode.c:** tablas de conversión para diferentes lenguajes.
- **mmc_stm32f1_spi.c:** código de control del modo SPI

El enlace a la librería se encuentra en el apartado de referencias.

3.1.4. Programa Principal

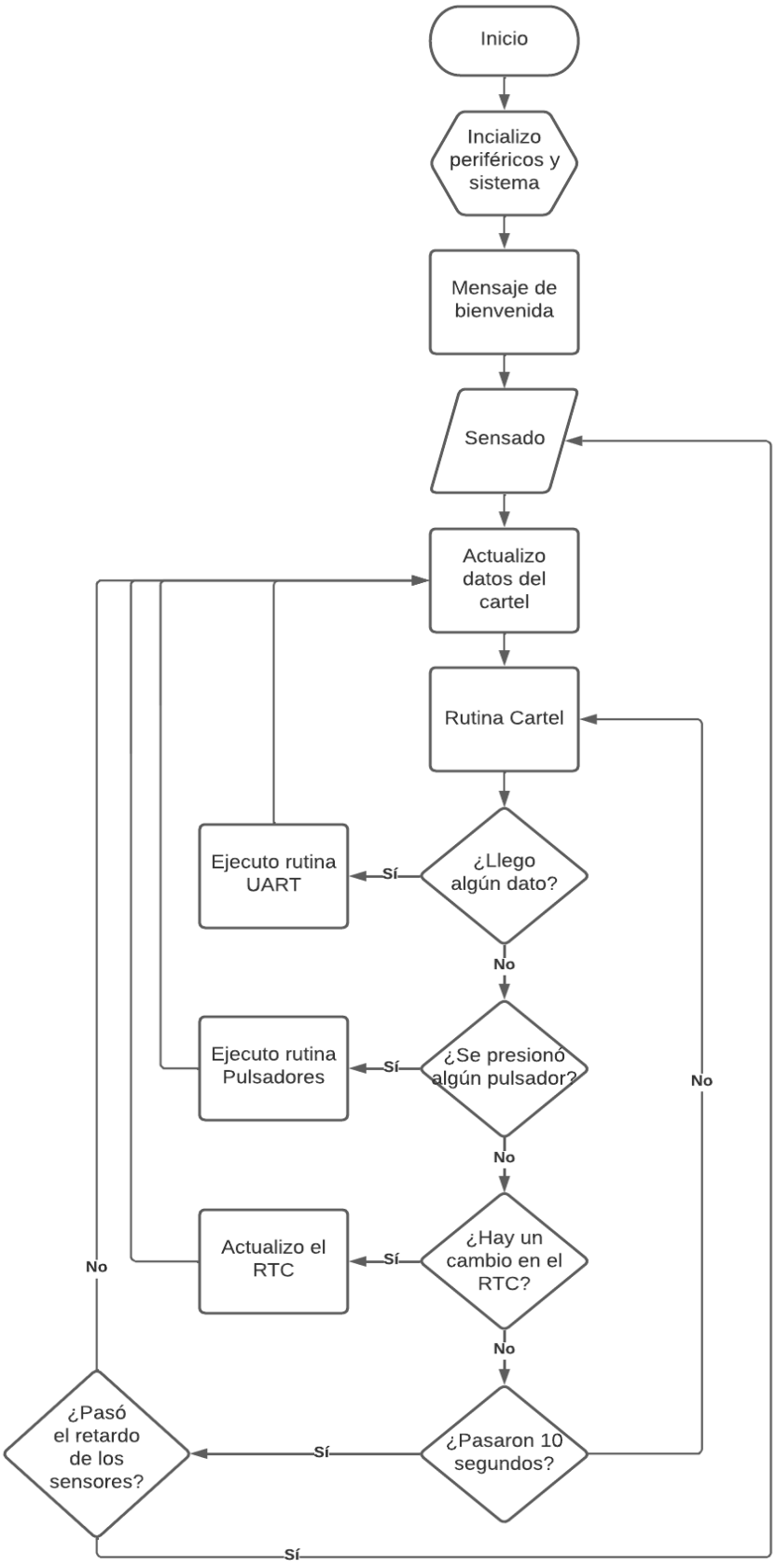

Fig.20: Diagrama de flujo general del sistema

El programa está enteramente diseñado en freertos, que tiene la particularidad de ejecutar un número prediseñado de tareas al mismo tiempo, por lo que el diagrama de flujo de arriba es una aproximación de lo que sucede en realidad. Al ejecutarse las tareas por orden de prioridad y al tener cada una un tiempo de ejecución distinto es difícil pensar en que el sistema tenga algún tipo de previsibilidad.

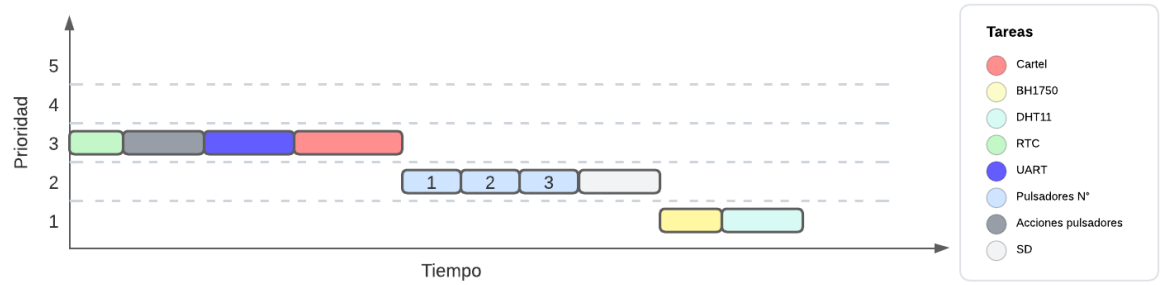

Fig.21: Diagrama de prioridad

En la imagen de arriba se ve como fue diseñado el planificador y la prioridad que se le dio a las tareas. Cabe destacar que la más importantes para nuestro criterio fueron:

- Tarea Cartel: donde se encarga de la manipulación de los mensajes en la matriz led
- Tarea UART: donde una vez recibido el mensaje por bluetooth se procede a separarlo y darle una función dependiendo del mensaje recibido.
- Tarea RTC: donde el reloj actualiza la hora y fecha actual.

```
void inicializar_tareas (void)
{
    xTaskCreate( tarea_cartel, "tarea_cartel", configMINIMAL_STACK_SIZE, NULL, 2, NULL );
    xTaskCreate( tarea_bh1750, "tarea_bh1750", configMINIMAL_STACK_SIZE, NULL, 2, NULL );
    xTaskCreate( tarea_DHT11, "tarea_DHT11", configMINIMAL_STACK_SIZE, NULL, 4, NULL );
    xTaskCreate(tarea_uart,"uart2",configMINIMAL_STACK_SIZE,NULL,1,NULL);
    xTaskCreate(tarea_actualizarRTC,"actualizarRTC",configMINIMAL_STACK_SIZE,NULL,tskIDLE_PRIORITY+2,NULL);
    xTaskCreate(tarea_pulsador1,"Pulsador1",configMINIMAL_STACK_SIZE,NULL,2,NULL);
    xTaskCreate(tarea_pulsador2,"Pulsador2",configMINIMAL_STACK_SIZE,NULL,2,NULL);
    xTaskCreate(tarea_pulsador3,"Pulsador3",configMINIMAL_STACK_SIZE,NULL,2,NULL);
    xTaskCreate(tarea_sd,"SD",configMINIMAL_STACK_SIZE,NULL,3,NULL);
    xTaskCreate(tarea_acciones_pulsador,"AccionesPulsador",configMINIMAL_STACK_SIZE,NULL,1,NULL);

}
```

Fig.22: Inicialización de tareas

En la imagen de arriba se puede observar la inicialización de tareas del RTOS cuyo orden de prioridad decrece a medida que el número de prioridad crece.

```
void inicializar_perifericos(void)
{
    inicializar_semaforos ();
    inicializar_cartel();
    inicializar_BH1750();
    inicializar_Matrix();
    HAL_TIM_Base_Start(&htim3);
    inicializar_SD();

}
```

Fig.23: Inicialización de periféricos

3.1.5. Rutinas de Interrupción

Rutina de recepción/transmisión por Bluetooth

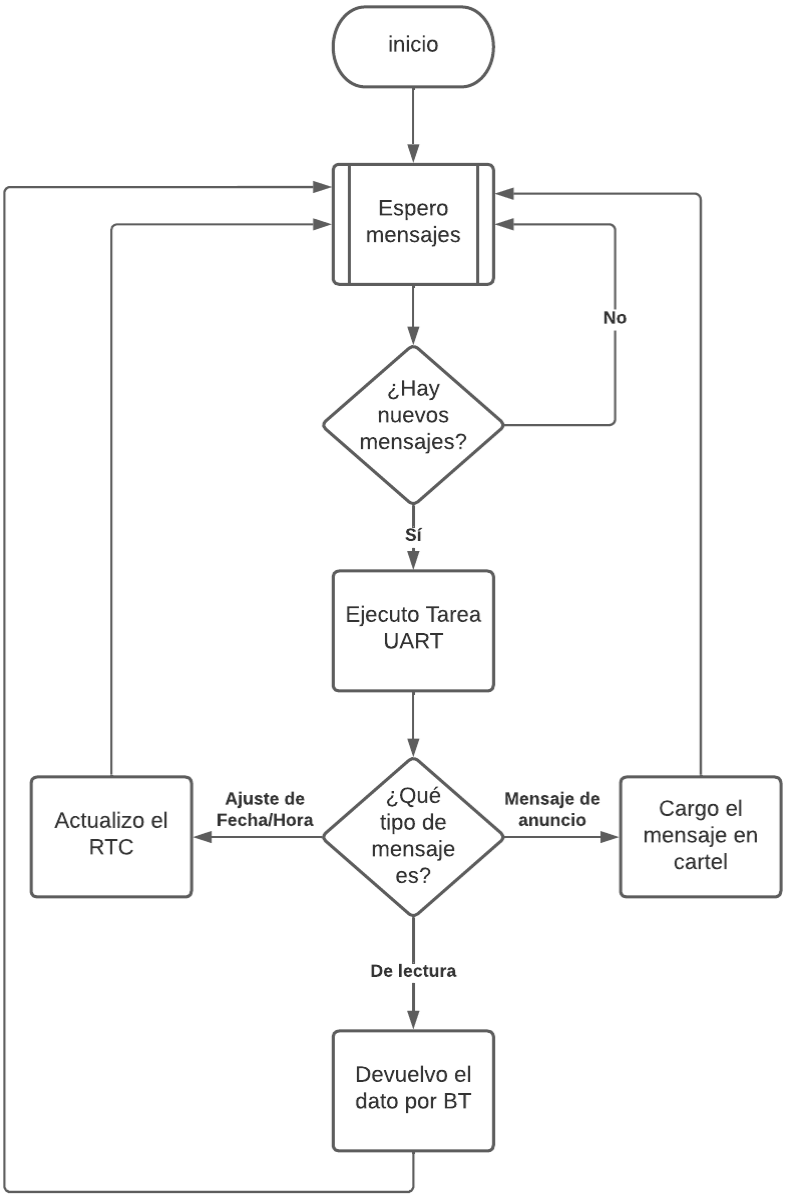

Fig.24: Diagrama de flujo tarea Bluetooth

La rutina de recepción por Bluetooth se hace en forma de interrupción y a través de comunicación serie. Cuando llega un nuevo mensaje el programa deja lo que está haciendo y va a atender el pedido de iniciar la recepción del mensaje. Mientras recibe, sigue ocupándose en las demás tareas. La HAL encargada de administrar eso es:

```
HAL_StatusTypeDef HAL_UARTEx_ReceiveToIdle_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
```

La HAL recibe 3 parámetros:

- El puntero a la uart
- El vector donde se van a guardar los datos
- Y el valor máximo de recepción.

HAL_UARTEx_ReceiveToIdle_IT permite recibir una cantidad máxima de datos predefinida en el parámetro Size, como también si llega a un estado Idle (es decir, porque el largo de la trama era mucho más chico que el valor máximo propuesto y encontró un fin de cadena en el medio), también puede cortar la recepción y avisar que terminó de recibir.

Una vez terminado de recibir los datos, dependiendo del tipo de mensaje recibido, el micro puede:

- Actualizar el valor de la hora y fecha en el RTC.
- Devolver un mensaje si lo que se está pidiendo es algún dato.
- Guardar un mensaje enviado por el usuario.

Rutina del teclado

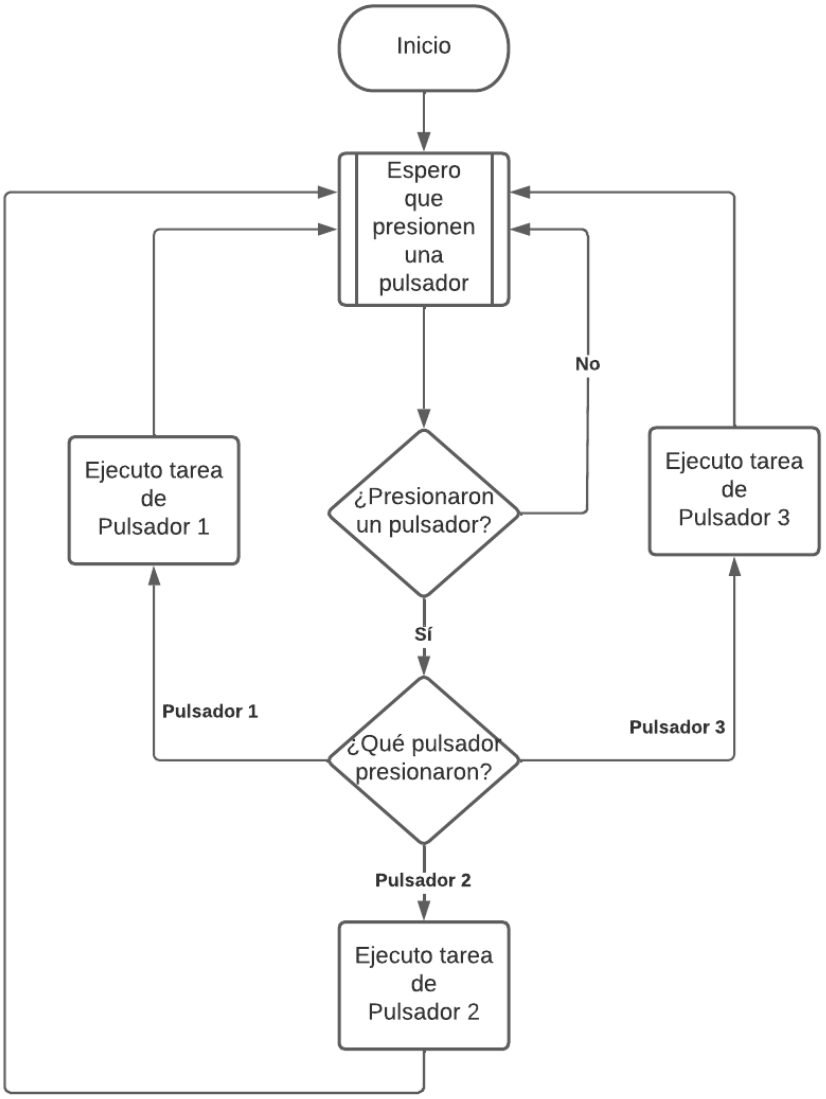

Fig.25: Diagrama de flujo tarea teclados

El teclado consiste en tres pulsadores conectados a la NUCLEO F401RE en forma de interrupción externa. Cada pulsador tiene su propia tarea y sub rutina encargado de modificar algún aspecto que afecte a lo que se muestra en la matriz de led.

Subrutina pulsador 1:
El pulsador 1 uno controlará los modos en que se mostrarán los datos en el cartel.

- El modo normal, el cartel mostrará la fecha, la hora y la temperatura cada 10 segundos, cada uno
- Si se presiona una vez, el cartel mostrará solo la fecha ininterrumpidamente.
- Si se presiona por segunda vez, el cartel mostrará solo la hora ininterrumpidamente.
- Si se presiona por tercera vez, el cartel mostrará solo la temperatura ininterrumpidamente.
- Si se vuelve a presionar volverá al modo normal

Subrutina pulsador 2:
El pulsador 2 mostrará el mensaje cargado en el dispositivo a través del bluetooth o del SD.

- Si se presiona una vez, el cartel mostrará el mensaje cargado por el Bluetooth.
- Si se presiona por segunda vez, el cartel mostrará el mensaje cargado por el SD.
- Si se presiona por tercera vez, el cartel volverá al modo normal.

Subrutina pulsador 3:
El pulsador 3 variará la velocidad en la que los datos se desplazan en el cartel.

- Si se presiona una vez, el cartel tendrá una velocidad baja.
- Si se presiona por segunda vez, el cartel tendrá velocidad media.
- Si se presiona por tercera vez, el cartel tendrá velocidad alta.
- Si se vuelve a presionar, volverá a velocidad baja.

3.1.6. Rutinas Generales

Rutina de la cartel led

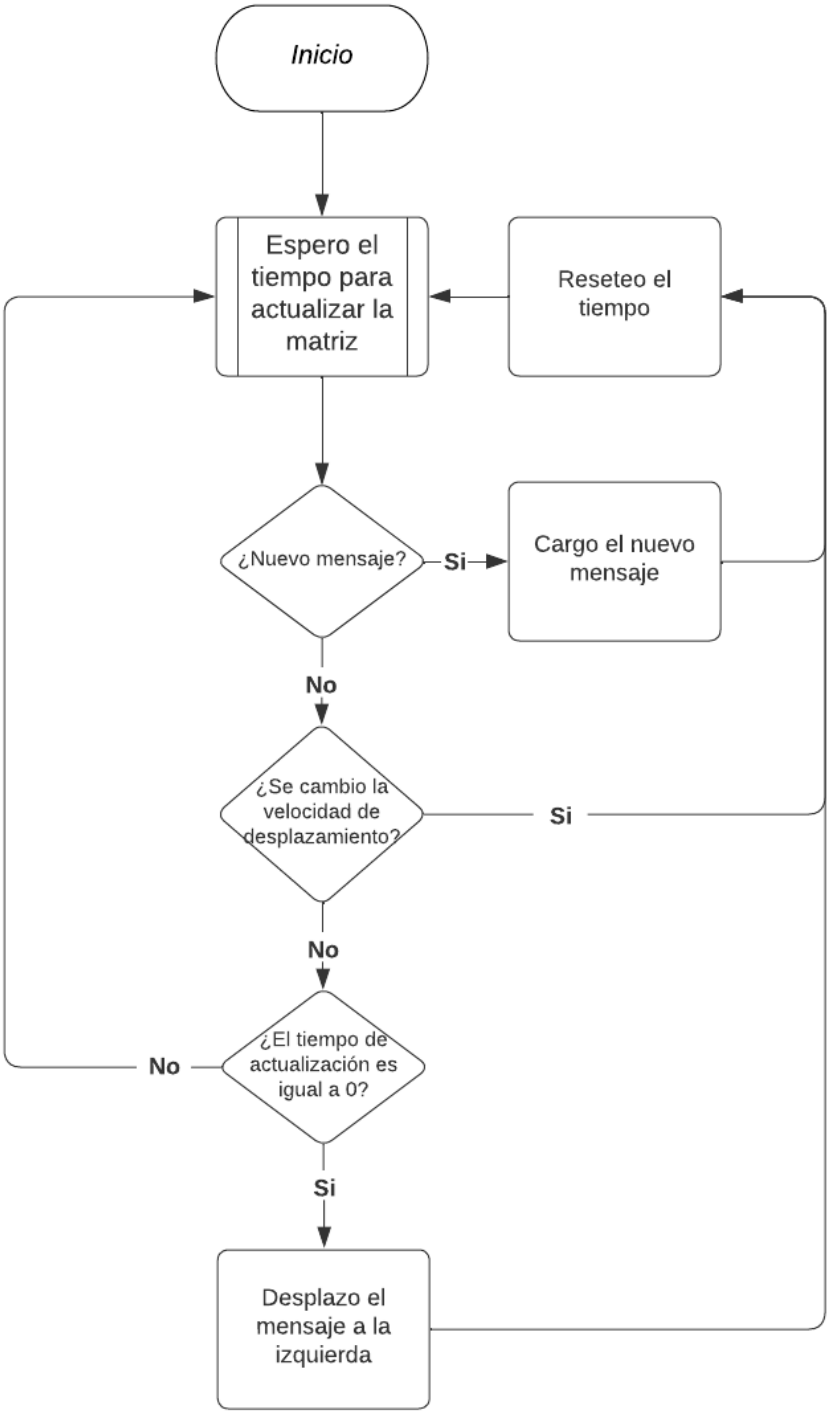

Fig.26: Diagrama de flujo tarea cartel led

El cartel led se actualiza cada cierta cantidad de milisegundos, cuando termina este tiempo, el mensaje del cartel se desplaza una posición a la izquierda. La función encargada de este desplazamiento es:

```
void MATRIX_TMR_OVF_ISR(TIM_HandleTypeDef* htim)
```

Cuando se modifica el mensaje a mostrar se apagan todos los leds de los módulos y empieza a escribir el nuevo mensaje por medio de la función. La función que apaga los leds de los módulos es:

```
void DOT_MATRIX_Clear(SPI_HandleTypeDef * hspi)
```

Si se presiona el pulsador 3 cambia la velocidad del desplazamiento del cartel.

Rutina RTC

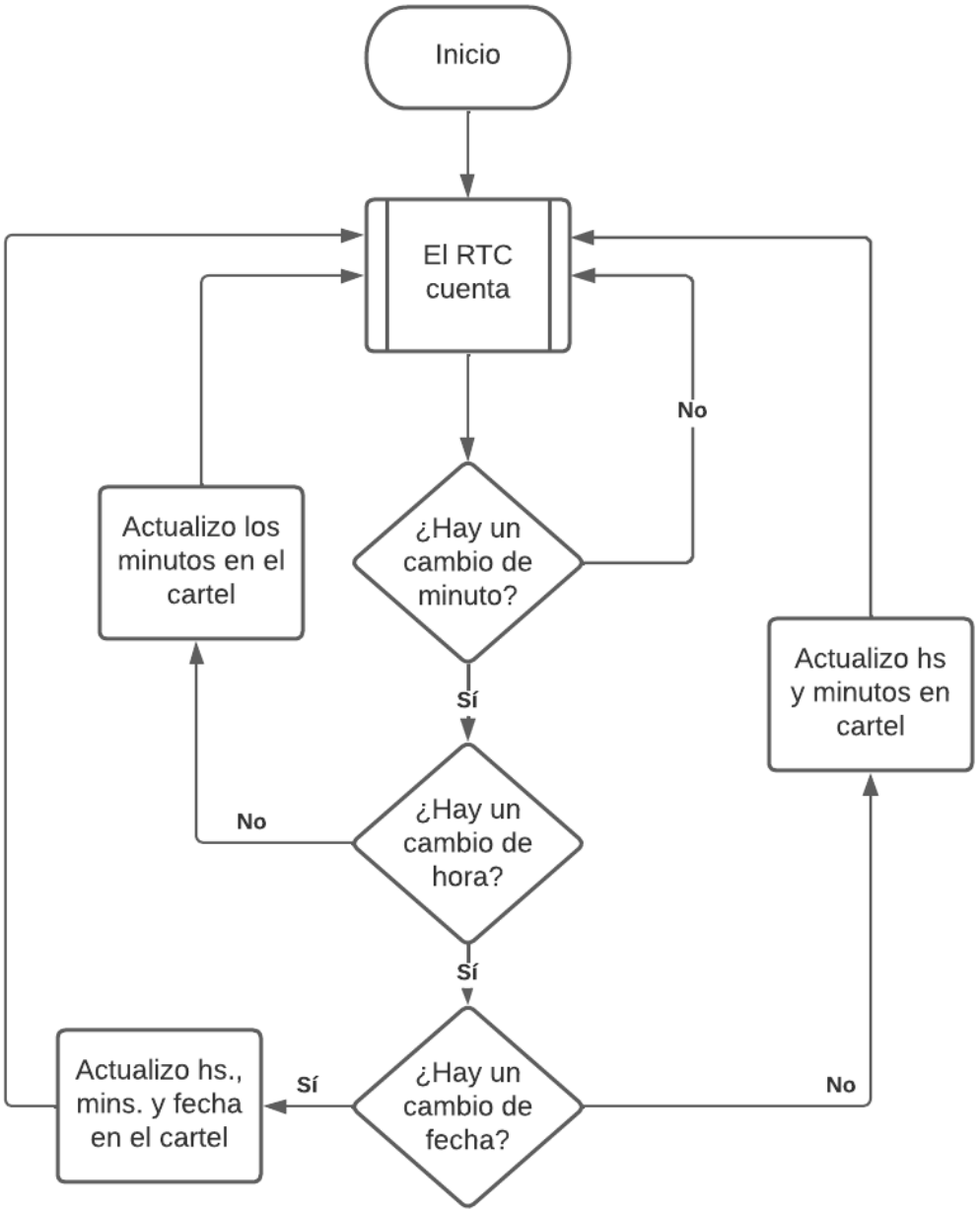

Fig.27: Diagrama de flujo tarea RTC

El RTC se encarga de mantener la cuenta de la hora y fecha actual en todo el sistema. Aún si el dispositivo se encuentra desenchufado, el RTC sigue funcionando debido a que está siendo alimentado por una batería CR2032.

Su funcionamiento es simple y casi imperceptible, cuenta segundos,minutos,horas y fechas. Cuando ocurre algún cambio, actualiza los datos del cartel. Su cuenta puede ser modificada o actualizada a través del Bluetooth.

Rutina Tarjeta Sd

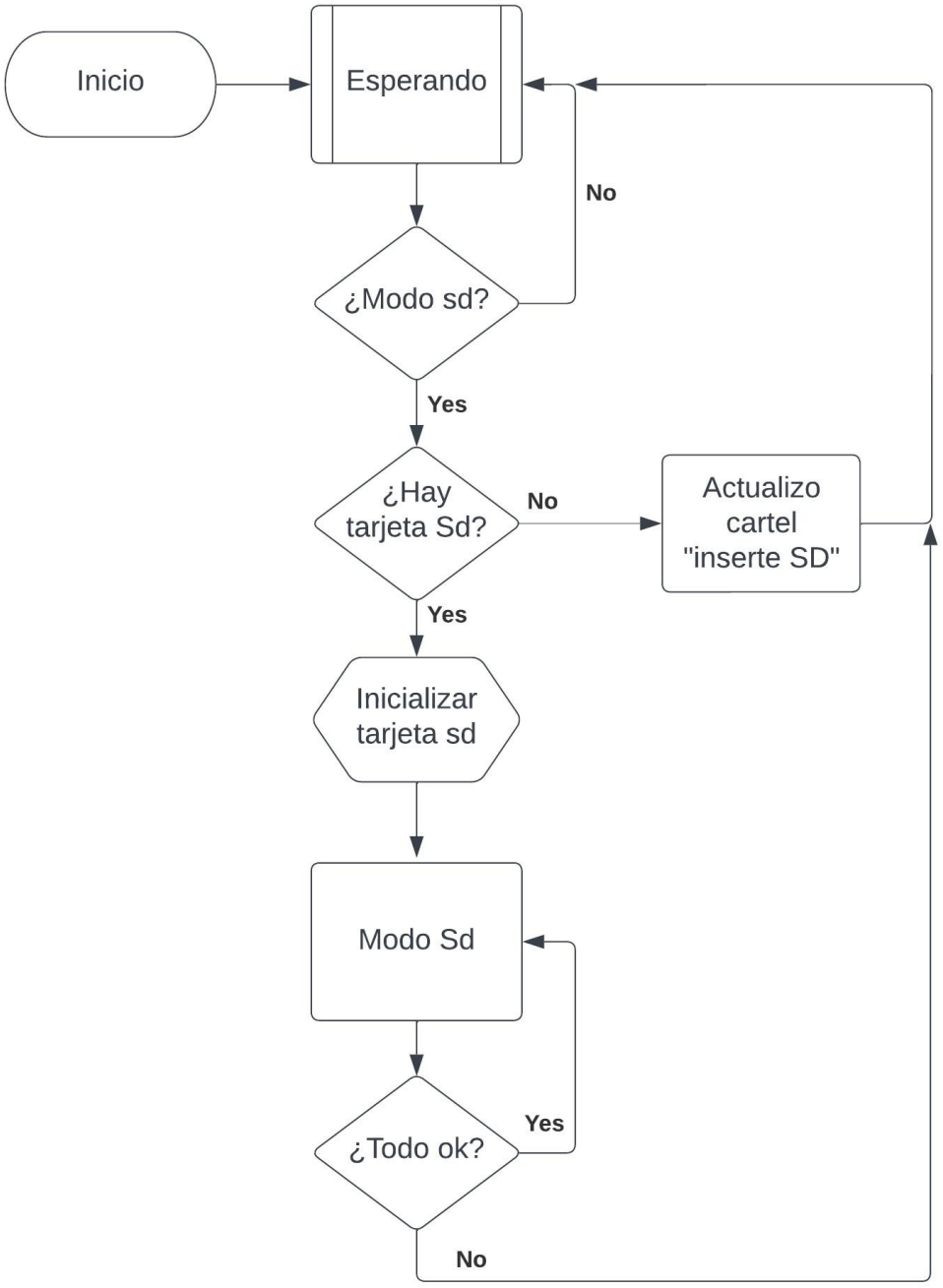

Fig.28: Diagrama de flujo tarea SD

Cuando el usuario seleccione este modo se tendrá que chequear si hay una tarjeta sd insertada, caso contrario se pedirá por el cartel que se inserte una. Si hubiera tarjeta insertada se procede a inicializar y pasar a modo sd.

Cuando se encuentre en el modo sd los mensajes se cargaran a partir de la tarjeta sd. Los mensajes estarán alojados dentro de un archivo llamado "mensajes.txt". La particularidad de este modo es que los mensajes del usuario contenidos dentro del archivo se cargarán de forma cíclica.

Por ejemplo: Si tuviera un solo mensaje, sería el único en mostrarse. Pero, si tuviera dos o más, se mostraran de a uno con una duración predeterminada, y siempre de forma cíclica. Se considera un mensaje a una línea del archivo de texto. El modo sd se encarga automáticamente de actualizar el buffer del cartel.
A continuación un ejemplo del contenido del archivo.

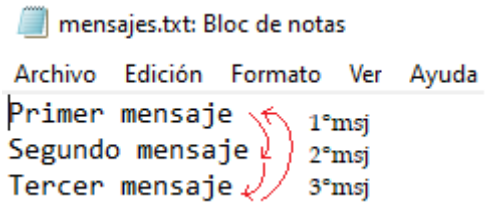

Fig.29: Creación de mensajes

En el caso de que se retire la tarjeta o se cambie de modo se retorna al estado inicial.

Rutina Sensor de Temperatura y Humedad

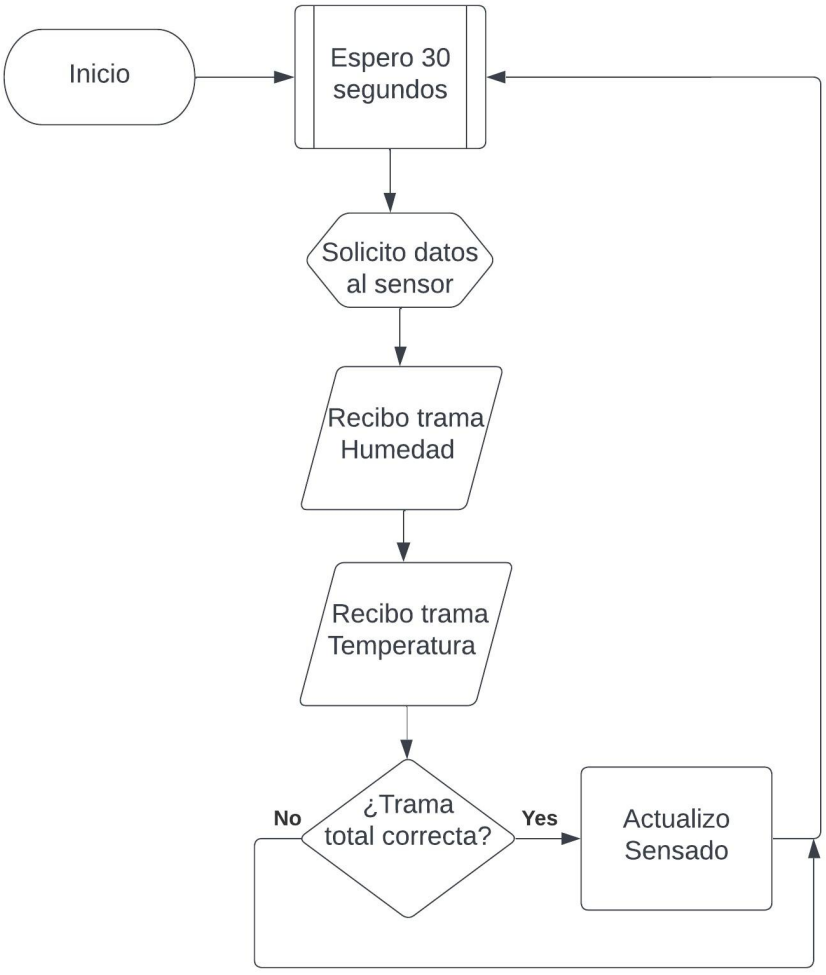

Fig.30: Diagrama de flujo tarea Sensor de temperatura DHT11

El sensor DHT11 es un dispositivo que capta la temperatura y la humedad del ambiente. Su forma de recibir es en formato serie, empezando por la humedad y luego por la temperatura.

La comunicación entre el microprocesador y el DHT11 y su sincronización se hace a través de un solo bus de datos de 40bits.

El formato de datos es:

8 bits de datos de humedad para enteros + 8 bits de datos de humedad para decimales + 8 bits de datos enteros de temperatura + 8 bits de datos decimales de temperatura + 8 bits de paridad.

La inicialización del DHT11 se hace a través de:

```
uint8_t DHT11_Start (void)
```

Las lecturas se hacen a través de de la tarea DHT11 con la función:

```
uint8_t DHT11_Read (void)
```

Rutina de sensor de luz digital

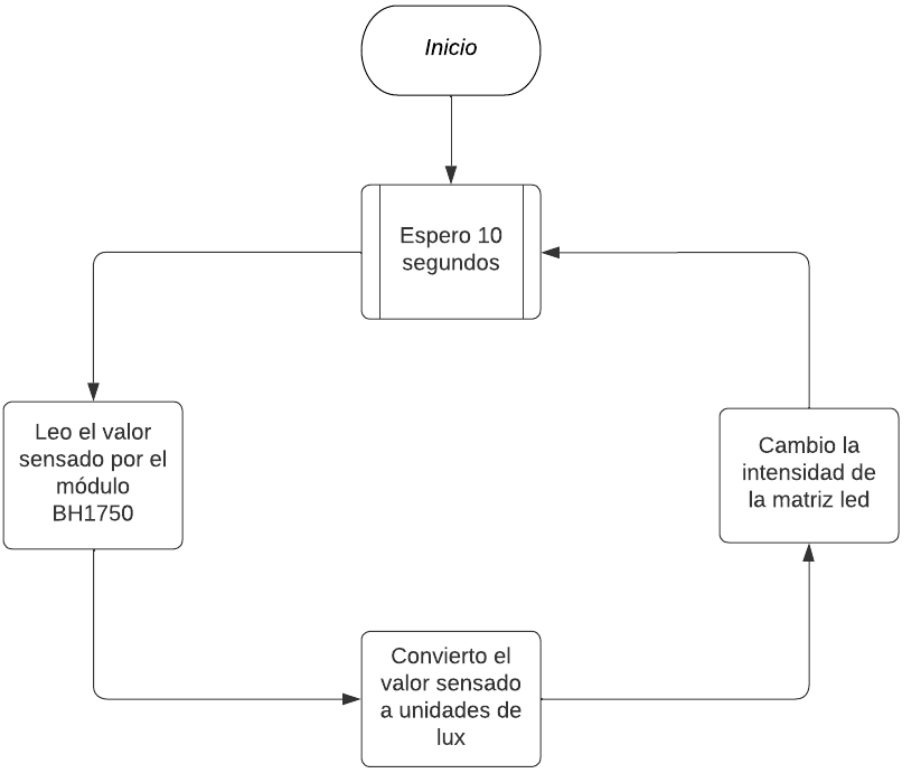

Fig.31: Diagrama de flujo tarea sensor de luz BH1750

La rutina del sensor de luz BH1750 se ejecuta por medio de la comunicación I2C cada 10 segundos, la función encargada de leer el valor y convertirlo a unidades de lux es:

```
HAL_StatusTypeDef BH1750_get_lumen(BH1750_device_t* dev)
```

En esta función se ejecutan 2 funciones más:

- BH1750_read_dev(dev): Se encarga de leer 2 bytes que sensa el módulo BH1750
- BH1750_convert(dev): Se encarga de unir los 2 bytes y dividirlo por 1,2 para convertirlo a lux

Una vez que terminó de convertir a unidades de lux, se determina el brillo de la matriz led:

- Si la iluminación es alta, el brillo de la matriz será alta
- Si la iluminación es baja, el brillo de la matriz será baja

La configuración de la intensidad del brillo se hace mediante la función:

DOT_MATRIX_Brightness(i); // Siendo i un valor del 1 al 10

4. Referencias

Hojas de datos

Sensor de luz BH1750: Link Datasheet

Sensor de temperatura DHT11: Link Datasheet

Módulo Bluetooth: Link Datasheet

Regulador de tensión LM1117: Link Datasheet

Matriz Led 8x8: Link Datasheet

IC Max7219: Link Datasheet

SD: Link Datasheet

Manuales

STM32 NUCLEO F401RE: Reference Manual

STM32 NUCLEO F401RE: User Manual

Librerías

Deep Blue Librería y proyecto: Link WEB
Librería Descarga: Link Descarga
Proyecto ChaN FATFS: Link WEB