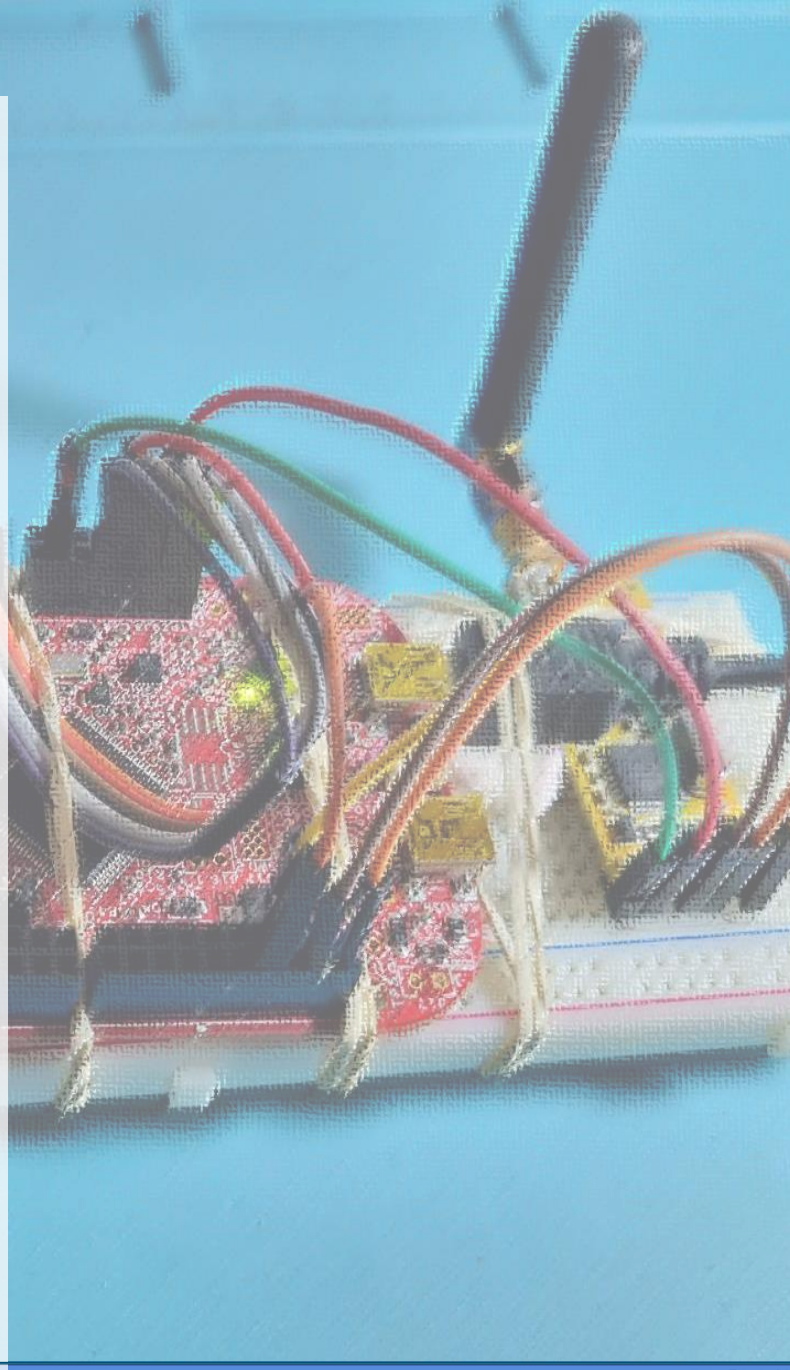
Simposio Argentino de Sistemas Embebidos



GRAFICADOR DE ACELERACIÓN CON ENLACE INALÁMBRICO

Autor:
WAGNER, WALTER S.

Tutor:
ING. MURO, GUSTAVO.

-2023-

ÍNDICE:

1. INTRODUCCIÓN:

En el presente informe se describe el dispositivo desarrollado para ser presentado en el marco del concurso estudiantil, categoría C, del Simposio Argentino de Sistemas Embebidos (S.A.S.E.) en su edición 2023, que se realiza los días 9, 10 y 11 de agosto de 2023 en la ciudad de Bahía Blanca, provincia de Buenos Aires. Este desarrollo surge como parte y continuación del Trabajo Práctico Final (T.P.3) de la asignatura Sistemas Digitales 2 (S.D.2), de la Facultad de Ciencias Exactas, Ingeniería y Agrimensura (F.C.E.I.A.), de la Universidad Nacional de Rosario (U.N.R.).

El trabajo práctico antes mencionado fue llevado a cabo, de manera grupal o individual, por los alumnos que cursaron la materia el primer semestre del corriente año; dirigidos y evaluados por los docentes de la cátedra. En particular, se perseguía el objetivo de desarrollar una aplicación práctica que, si bien fuera de tipo didáctica, sea fácilmente escalable y que les brinde a todos los estudiantes las herramientas para lidiar con cualquier situación problemática que se presenta en un desarrollo comercial. De esta manera se buscó sentar las bases que se utilizan en el ámbito profesional junto con las buenas prácticas y estándares empleados en el modelado y desarrollo de sistemas embebidos.

Este trabajo tenía por consigna general aplicar todos los contenidos temáticos de la asignatura para utilizar el microcontrolador Cortex M0+ de la placa de desarrollo FRDM-KL46Z en una aplicación práctica con el fin de establecer una comunicación serie RS-485 con otro dispositivo. Al mismo tiempo que se buscaba explorar el uso de diversos protocolos, como I2C, SPI y UART, para realizar intercambio de mensajes con periféricos de entrada (sensor de aceleración y pulsadores en este caso) y de salida (pantalla OLED y LEDs).

El funcionamiento del sistema debía ser modelado utilizando el formalismo de Máquina de Estado Finito / Statecharts UML y el código C debía reflejar el modelo propuesto. El intercambio de mensajes entre los dos dispositivos tenía que ser implementado utilizando estructuras de datos de tipo cola circular en la recepción y transacciones a cargo de DMA en la transmisión. En ambos casos el protocolo para comunicarse con el exterior debía ser UART.

El desarrollo de la aplicación podía incorporar funciones de biblioteca provistas por el fabricante. La implementación se tuvo que programar y depurar utilizando el ambiente MCUXpresso.

Se esperaba que el dispositivo desarrollado pudiera comunicarse con la PC por UART/RS485 y debía, además, contar con dos modos de trabajo: en uno de ellos se tenía que interactuar con el software provisto por la cátedra: “SD2_TP3_Master”, que se muestra en la *Figura 1.1*; el mismo es capaz de encuestar el estado de los pulsadores, graficar la aceleración y controlar los LEDs de la placa. Mientras que en el otro modo se debían enviar tramas periódicamente, informando la aceleración en los tres ejes, para ser procesadas en PC mediante un software para animación 3D, que se muestra en la *Figura 1.2* y emula el movimiento de un avión.

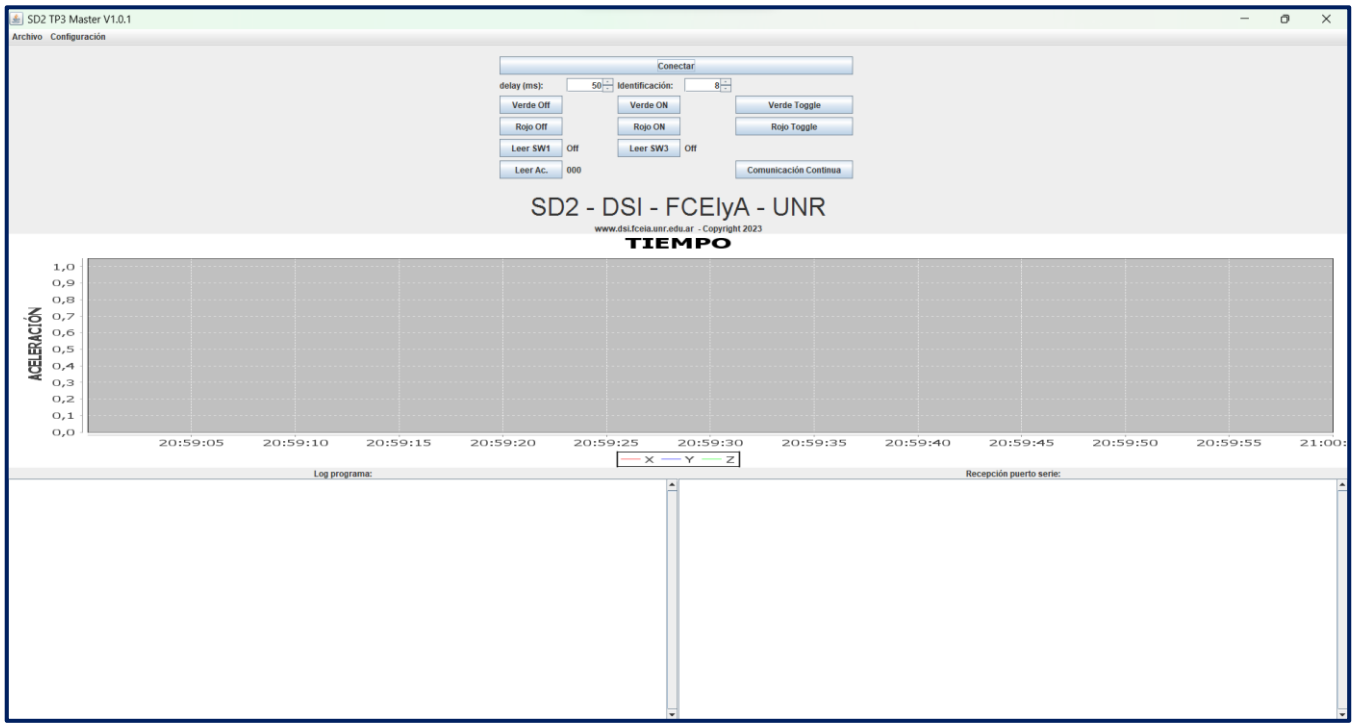


Figura 1.1: Entorno del software "SD2_TP3_Master".

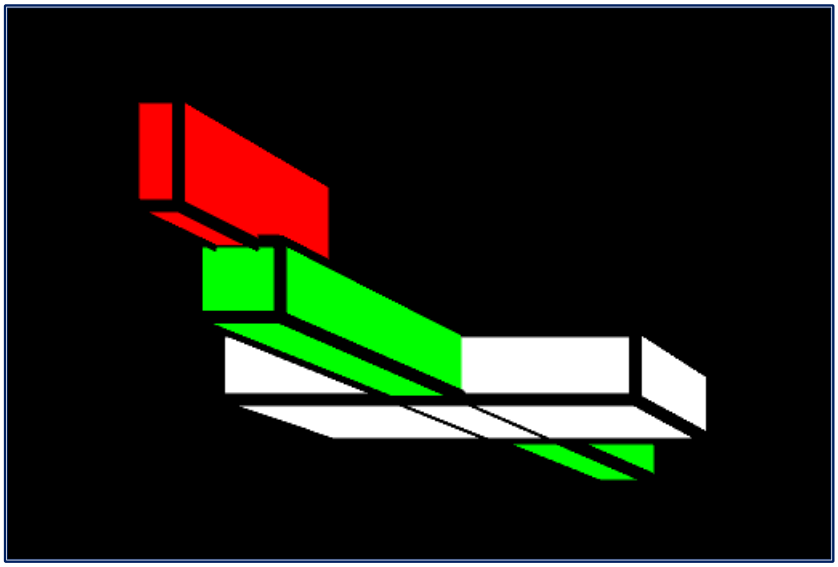


Figura 1.2: Simulación en "Processing_3d".

Mediante el uso de los pulsadores propios de la placa se debía cambiar de un modo de trabajo al otro y se debería indicar, para un correcto funcionamiento, un mensaje que informe el modo operativo actual, en un display OLED comunicado vía protocolo SPI.

Para el primer modo antes mencionado, la placa FRDM-KL46Z debía ser el dispositivo esclavo en la comunicación, recibiendo las peticiones del maestro y dando la respuesta correspondiente a los diferentes mensajes. El dispositivo maestro es siempre la computadora con un adaptador RS485 conectado a uno de los puertos USB.

Cuando en la computadora se ejecuta el programa SD2_TP3_Master, este envía los mensajes hacia el esclavo y recibe las respuestas para poder visualizarlas. Todos los mensajes enviados y recibidos son codificados en caracteres ASCII y comienzan con el carácter ':' (0x3A), y terminan con el carácter LF (0x0A). Los posibles mensajes se detallan a continuación en la *Figura 1.3*. En todos los casos XX representa un número de identificación que va desde "01" a "99", cada grupo utilizó un número asignado por los docentes:

Petición	Respuesta	Descripción
:XX01Y'LF'	:XX01Y'LF'	Acciones sobre el Led Rojo. Valores posibles de Y:
Ejemplo:	Respuesta:	E: encender
:0201E'LF'	:0201E'LF'	A: apagar
	(igual a la petición)	T: toggle
:XX02Y'LF'	:XX02Y'LF'	Acciones sobre el Led Verde. Valores posibles de Y:
Ejemplo:	Respuesta:	E: encender
:0202E'LF'	:0202E'LF'	A: apagar
	(igual a la petición)	T: toggle
:XX11'LF'	:XX11Y'LF'	Leer estado del SW1. Valores posibles de Y:
Ejemplo:	Respuesta:	N: pulsador no presionado
:0211'LF'	:0211P'LF'	P: Pulsador presionado
:XX13'LF'	:XX13Y'LF'	Leer estado del SW3. Valores posibles de Y:
Ejemplo:	Respuesta:	N: pulsador no presionado
:0213'LF'	:0213P'LF'	P: Pulsador presionado
:XX21'LF'	:XX21SXXXSYYYSZZZ'LF'	Transmitir el valor de los 3 ejes de aceleración en centésimas de G. "S" es el signo, puede ser '+' o '-'.
Ejemplo:	Respuesta:	
:0121'LF'	:0121+017-002+101'LF'	

Figura 1.3:Tramas del modo "TP3_MASTER".

En el modo de funcionamiento "Plot3D", desde la placa de desarrollo se deberían enviar los datos de aceleraciones de la siguiente forma: X Y Z (números enteros, separados por un espacio) y al finalizar, un LF, por ejemplo:

-2 21 100'LF'

El trabajo práctico antes descripto fue llevado a cabo en tiempo y forma y aprobado cumpliendo con todos los objetivos propuestos. Así mismo, al alcanzar estas metas se abrieron nuevos horizontes que desplegaron cuestiones que podían ser mejoradas y optimizadas, tanto a nivel de implementación como a nivel de código. La posibilidad de presentar este trabajo en el SASE 2023 fomentó la curiosidad del saber y propuso llevar este desarrollo a un nuevo nivel.

2. OBJETIVOS Y DESAFÍOS:

Como se detalló en la introducción del presente informe, la primera versión del Graficador de Aceleración se desarrolló como Trabajo Práctico final de la asignatura Sistemas Digitales 2. Esta aplicación dio origen a nuevas necesidades e inquietudes que, impulsadas por la posibilidad de ser presentado en el SASE 2023, desencadenaron sinérgicamente en el apremio por desarrollar una versión mejorada del Graficador de Aceleración, esta vez con Enlace Inalámbrico.

Dentro de los objetivos y consignas propuestos en la introducción, se anexaron y desplegaron nuevas demandas, dentro de las cuales se destacan las siguientes:

1. Tomar como punto de partida el trabajo desarrollado, esto es, no modificar el funcionamiento en general y reutilizar todo lo que se había desarrollado ya; aprovechando las virtudes, pero sin perder de vista los defectos, para no cometer los mismos errores. De esta manera poder enfocarse en mejorar cuestiones de implementación, perfeccionar la detección y el procesamiento de las tramas y optimizar el código al mismo tiempo que se documenta y organiza de una manera más profesional, ordenada y escalable todo el proyecto.
2. Reemplazar la comunicación cableada con la PC a través de protocolo RS485 por una comunicación inalámbrica a través de Radio Frecuencia (RF), manteniendo a UART como el protocolo para vincular el dispositivo con el transreceptor.
3. Reducir el tiempo de latencia entre tramas enviadas y recibidas.
4. Agregar la funcionalidad de mostrar en el display OLED, comunicado por protocolo SPI, el módulo de la aceleración con una tasa de refresco adecuada para una correcta visualización, al mismo tiempo que se mejoraban las funciones gráficas del mismo.
5. Explorar el uso del display de cristal líquido presente en la placa de desarrollo y comunicado al micro por protocolo I2C, mostrando los mensajes “SASE” y “2023” de manera alternada.

3. MODELADO:

Para modelar el funcionamiento del Graficador de Aceleración con Enlace Inalámbrico se hizo uso de Máquinas de Estado Finito (MEF) jerárquicas / Statecharts UML, entendiendo a esta técnica de modelado bajo el paradigma de que debe ser una justa representación del comportamiento del sistema, pero con la suficiente simplificación para no entorpecer la lectura ni la comprensión. Es decir, se omiten todas aquellas cuestiones que no hacen al funcionamiento básico y general del sistema, aceptado que si se quiere analizar en detalle se debe acudir a la codificación, realizada en lenguaje C en este caso.

En primer lugar, se tiene una MEF jerárquica de nombre "Modo", que se muestra en *Figura 3.1* y con mayor resolución en el *Anexo I*. Esta MEF es un superestado en sí mismo, en donde se muestra en el display OLED el módulo de la aceleración con una tasa de refresco de 300 milisegundos y en el display LCD los mensajes "SASE" y "2023" de manera alternada cada un segundo. La MEF "Modo" contiene dos estados que se corresponden, como su nombre lo indica, con los dos modos de trabajo del sistema.

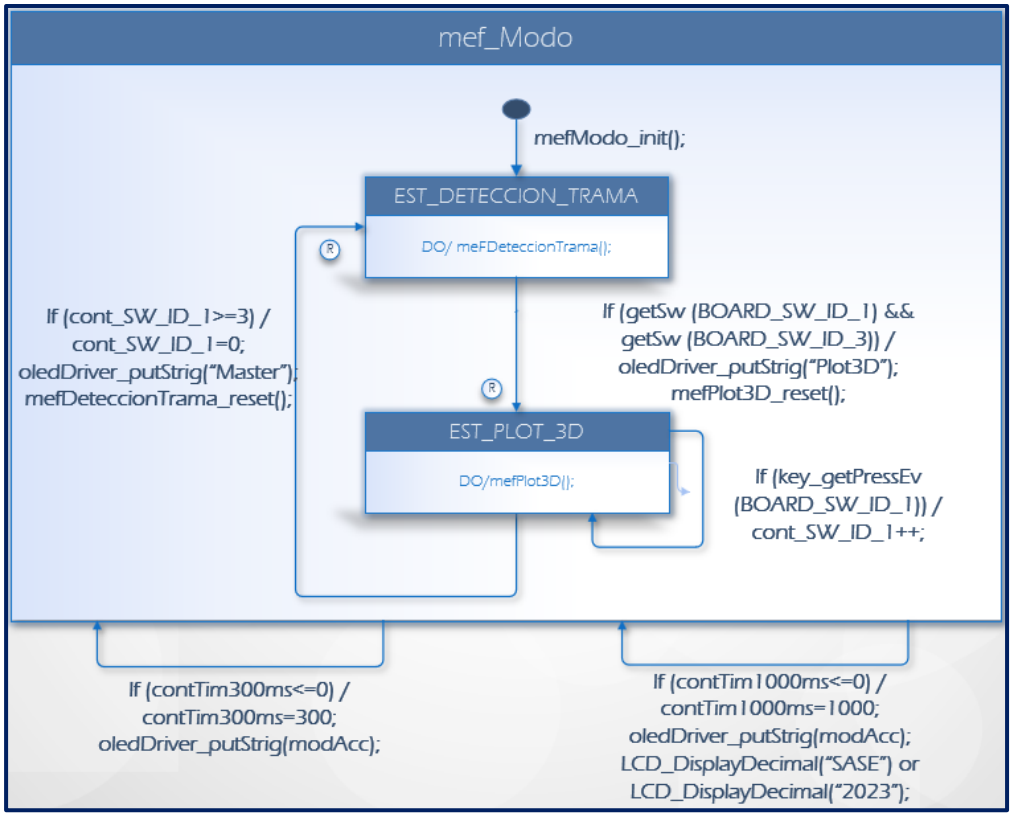


Figura 3.1: MEF MODO (Ver ANEXO I).

Al comienzo, esta máquina de estados realiza una subrutina de inicialización y luego pasa al estado que se corresponde al modo de trabajo en el que el dispositivo se comunica con el software de PC "SD2_TP3_Master". Este es un superestado en el que se correrá la sub máquina de estados "Detección de tramas", que se explicará a continuación. La MEF jerárquica permanecerá en este estado procesando tramas hasta tanto se presionen los dos pulsadores en

simultáneo (SW1 y SW3), en caso de que esto ocurra se evolucionará al siguiente superestado con reset de la subMEF correspondiente.

En el siguiente estado, el dispositivo se vincula con el software “Processing” en el modo Plot3D. En este superestado se corre otra subMEF, que también se explicará cuando sea oportuno, y se saldrá del mismo con 2 eventos de SW1, reseteando a la subMEF correspondiente y regresando al estado de detección de tramas.

En cada estado se muestra el modo de funcionamiento actual en el display OLED.

Para el caso de la subMEF “Detección Trama”, que se muestra en *Figura 3.2* y con mayor resolución en el *Anexo II*, se cuenta con dos estados. Posterior a un reset, esta MEF ejecuta su rutina de inicialización y se posiciona en el estado ESPERANDO, del cual sólo se irá si se recibe un byte de inicio.

Si se reconoce un byte indicando el inicio de trama, entonces se guarda este byte en la primera posición de la trama y se realiza la transición al estado RECIBIENDO, en donde tenemos 4 opciones que deben ser atendidas.

a) En primer lugar, se puede recibir un nuevo byte de inicio, lo cual nos obligaría a limpiar la trama anterior, guardar el byte de inicio y continuar posicionados en el estado RECIBIENDO.
b) En segundo lugar, podemos recibir un byte distinto al inicio y al fin de trama, en ese caso, siempre y cuando el buffer tenga espacio, se debe guardar el byte recibido en la fila.
c) En tercer lugar, puede ser recibido el byte de fin de trama, en ese caso se debe procesar la trama completa para ver si es válida y en cuyo caso tomar la acción correspondiente para luego limpiar el buffer y volver al estado inicial ESPERANDO.
d) En cuarto y último lugar, se puede dar el caso de que se siguen recibiendo datos distintos al byte de inicio y fin, pero simplemente el tamaño excede al máximo que puede tener una trama válida. En este caso es conveniente descartar la trama, limpiar el buffer y volver al estado ESPERANDO.

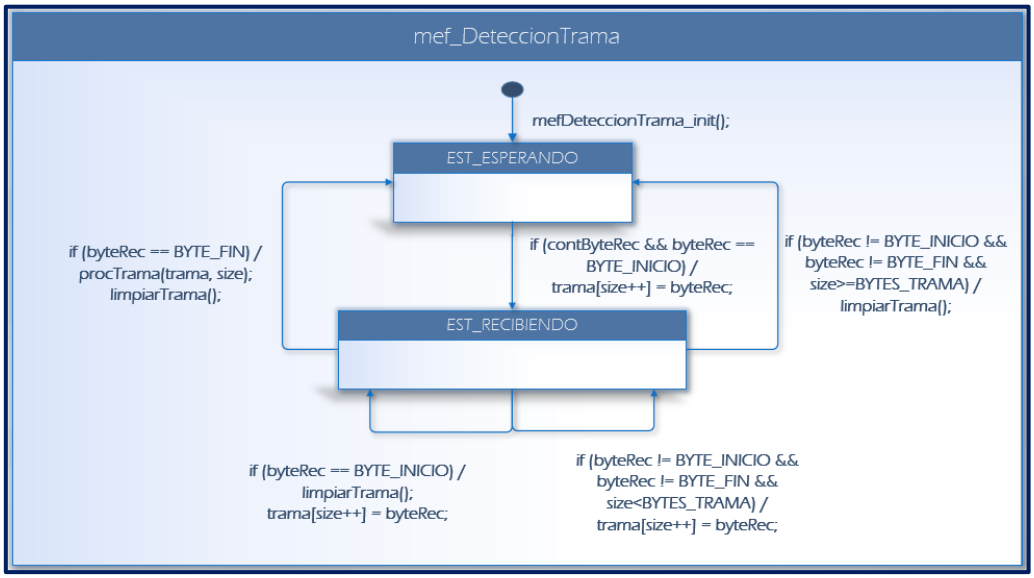


Figura 3.2: MEF DETECCION TRAMA (Ver ANEXO II).

La tercera y última máquina de estados involucrada en la implementación es “Plot3D”, la misma se muestra en la *Figura 3.3* y con mayor resolución en el *ANEXO III*. Esta subMEF también cuenta con dos estados e inicia corriendo su propia subrutina de inicialización.

En el primer estado, de nombre ESPERANDO, permanece generando un delay temporal y una vez que transcurren 100 ms evoluciona al estado ENVIANDO. En este último, cuando se tienen los valores de aceleración, es decir, cuando el acelerómetro interrumpe al microcontrolador informando en sus flags que la excepción fue por “Data Ready”, se envía una trama con la lectura de aceleración en los tres ejes, con un formato preestablecido para que el programa de visualización en 3D pueda procesarlos. Luego se retorna al primer subestado reiniciando la cuenta de tiempo. El delay es necesario para no sobrecargar al puerto serie y permitir un correcto funcionamiento del modo Plot3D.

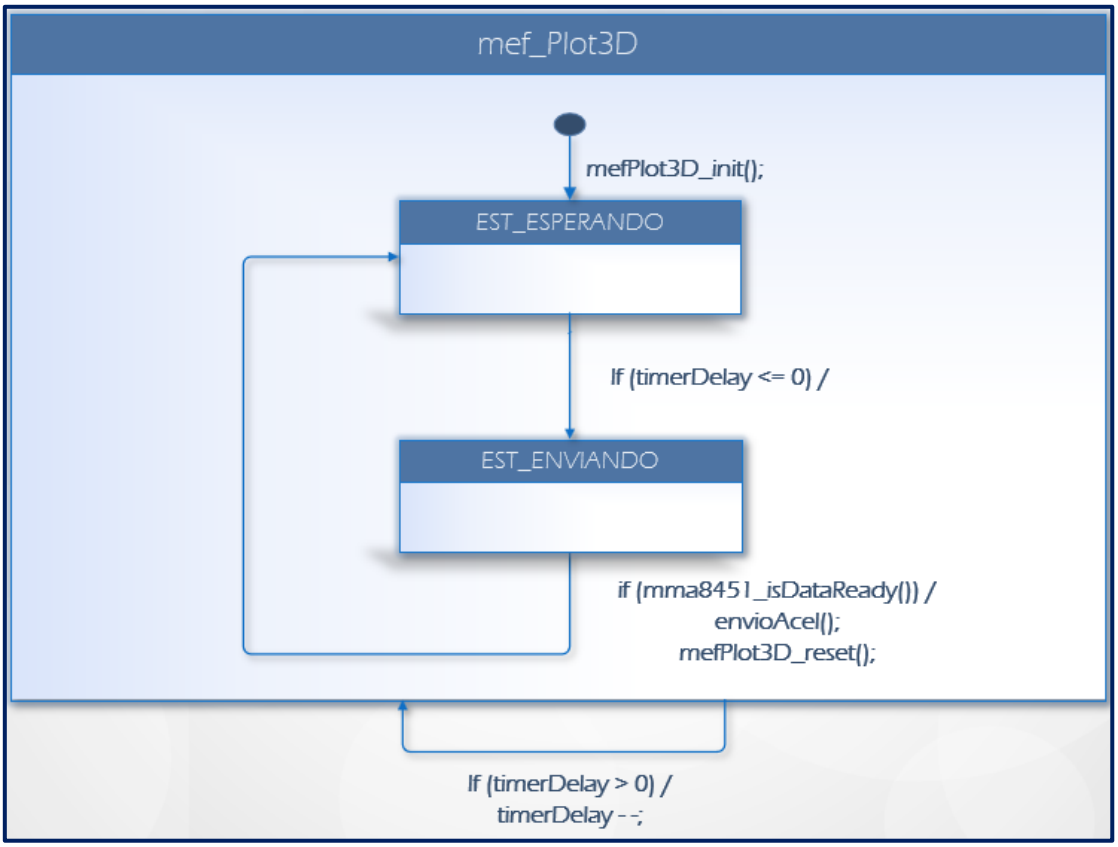


Figura 3.3: MEF PLOT 3D (Ver ANEXO III).

4. IMPLEMENTACIÓN:

HARDWARE:

Para realizar la implementación, la consigna era clara en cuanto al microcontrolador; se debía emplear el kit de desarrollo FRDM-KL46Z[1], que se utiliza a lo largo de todas las materias que componen a la rama de Sistemas Digitales dentro de la carrera de Ingeniería Electrónica. Esta plataforma cuenta con un CORTEX M0+ que corre hasta 48MHz, tiene arquitectura ARM y es de la familia NXP; cuenta además con 256KB de flash y 32KB de SRAM.

Dentro de otras múltiples características la placa posee dos pulsadores, dos diodos LEDs, un sensor de aceleración MMA84511[2] y un display LCD de 7 segmentos, periféricos necesarios para la implementación. Además de los periféricos presentes en la placa, fue necesario un display OLED comunicado por SPI, se optó por uno ampliamente disponible en el mercado de 128x64.

En cuanto a la comunicación con la PC, en la primera versión del Graficador de Aceleración, se hizo uso de adaptadores de UART a RS485 para la conexión con la placa y RS458 a USB para la PC. El esquema de conexiones para esta versión se expone a continuación en la *Figura 4.1*:

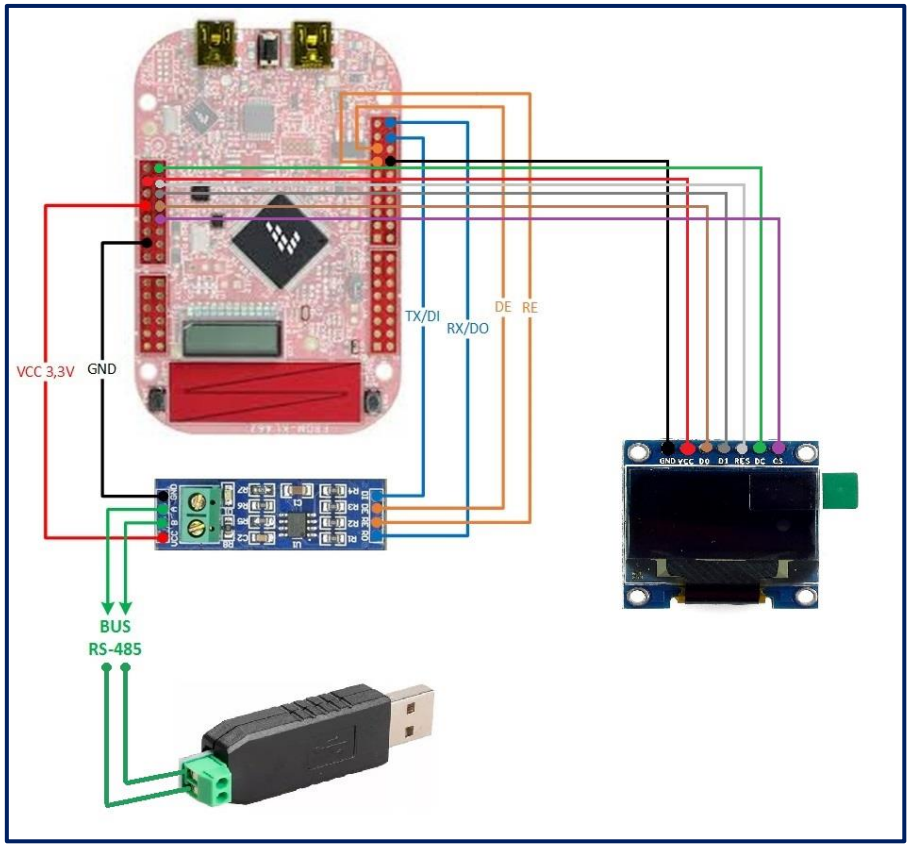


Figura 4.1: Esquema de conexión para la primera versión del Graficador de Aceleración.

[1] https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-kl3x-and-kl4x-mcus:FRDM-KL46Z
[2] https://www.nxp.com/docs/en/data-sheet/MMA8451Q.pdf

En el caso de la versión final del Graficador de Aceleración con Enlace Inalámbrico se utilizaron dos placas APC200A, cuya hoja de datos se encuentra fácilmente en la red o en la referencia citada al pie[3]. Una de las placas se vincula al kit de desarrollo y otra a la PC mediante un adaptador a USB. En ambas fue necesario colocar una antena para establecer el vínculo RF. El esquema de conexiones para esta versión se expone a continuación en la *Figura 4.2*:

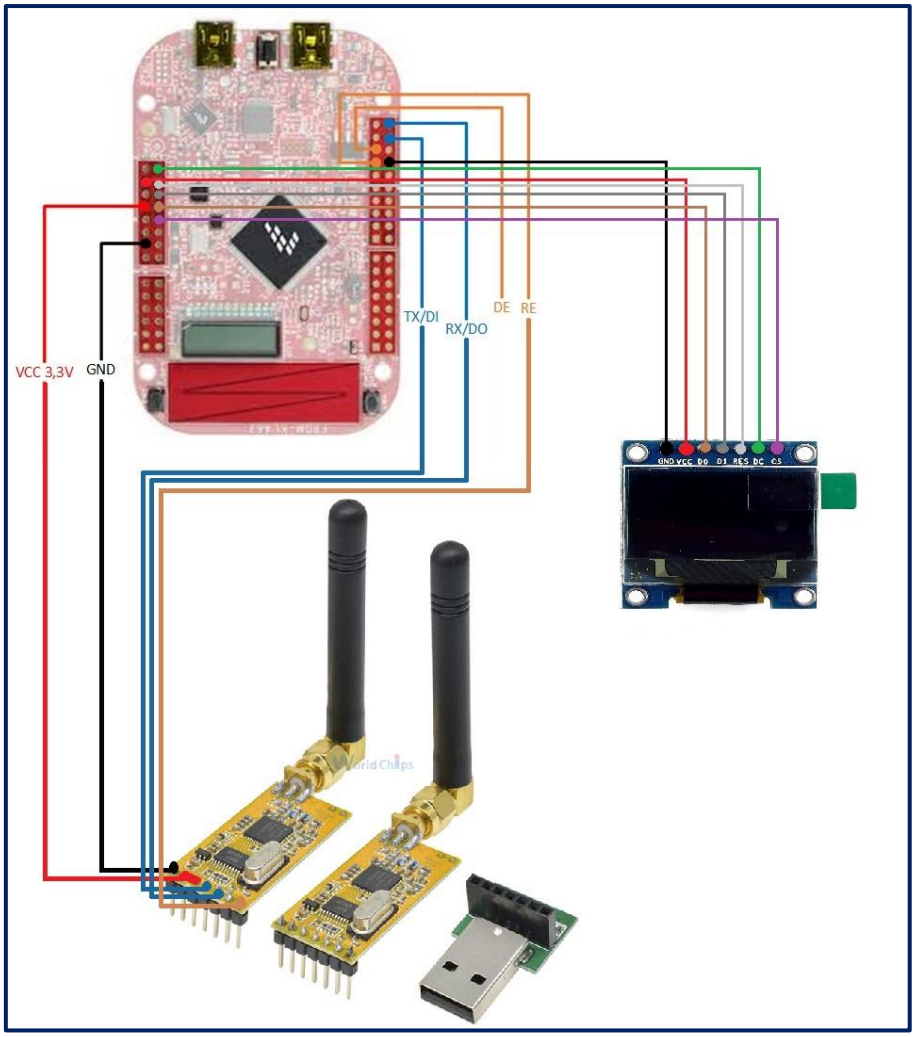


Figura 4.2: Esquema de conexión del Graficador de Aceleración con Enlace Inalámbrico.

Resulta importante resaltar que el Graficador de Aceleración con Enlace Inalámbrico mantiene la característica de poder funcionar de manera cableada por RS485. Fue diseñado de manera tal de trabajar con enlace inalámbrico, pero los módulos APC200 pueden ser reemplazados por los adaptadores a RS485 de forma “Plug&Play”.

[3] https://drive.google.com/file/d/1iPO_oystVQpWW4yRS81OqzHI-dhyYill/view?usp=sharing

SOFTWARE:

A lo largo de la materia Sistemas Digitales 2, en la medida que se dictaba la temática correspondiente a los módulos teóricos, por ejemplo, los protocolos de comunicación SPI, I2C, UART, etcétera, se iban llevando a cabo ejercicios prácticos de aplicación sobre la placa de desarrollo FRDM-KL46Z con el entorno MCUXpresso.

Todos los ejercicios realizados, pensados desde un principio bajo el paradigma y los estándares de la programación modular y la reutilización de código, contribuyeron a construir bibliotecas propias que fueron realmente esenciales para el desarrollo de este trabajo. Esto se complementa con bibliotecas brindadas por el fabricante con ejemplos de aplicación y archivos de ejercicios demostrativos brindados por los docentes de la cátedra. En cada caso, cada vez que se reutilizó o reversionó código, se reconoció debidamente la autoría intelectual del mismo en la documentación correspondiente que acompaña a cada archivo del proyecto.

Mostrar la codificación de todo el proyecto en este informe sería un despropósito, ya que implicaría exhibir miles de líneas de código extendiendo desmesuradamente la presentación. En su lugar, se creyó conveniente hacer énfasis en las cuestiones forma que se contemplaron al desarrollar el código. Si el lector deseara consultar cuestiones en lo particular (*-se recomienda hacerlo-*), **se deja al pie el proyecto publicado en GitHub[4] para que pudiere tenerlo a disposición.**

A lo largo del proceso de convertir el problema modelado por MEFs en una implementación en lenguaje C consistente, se trabajó en capas diferenciando aplicación de drivers, haciendo uso de archivos y carpetas diferentes se logró una organización intuitiva del proyecto, como se puede visualizar en la *Figura 4.1*. Esto permite reutilizar fácilmente código de cualquier capa, pero también simplificó en gran medida la depuración y optimización del proyecto.

Además, se tuvo en cuenta como buena práctica colocar un nombre significativo a cada archivo y a cada función o subrutina del mismo. En adición, el nombre de cada subalgorítmo comienza con el nombre del archivo que lo contiene.

[4] https://github.com/WalyWagner/GraficadorDeAcelaracionConEnlaceInalambrico-SASE2023

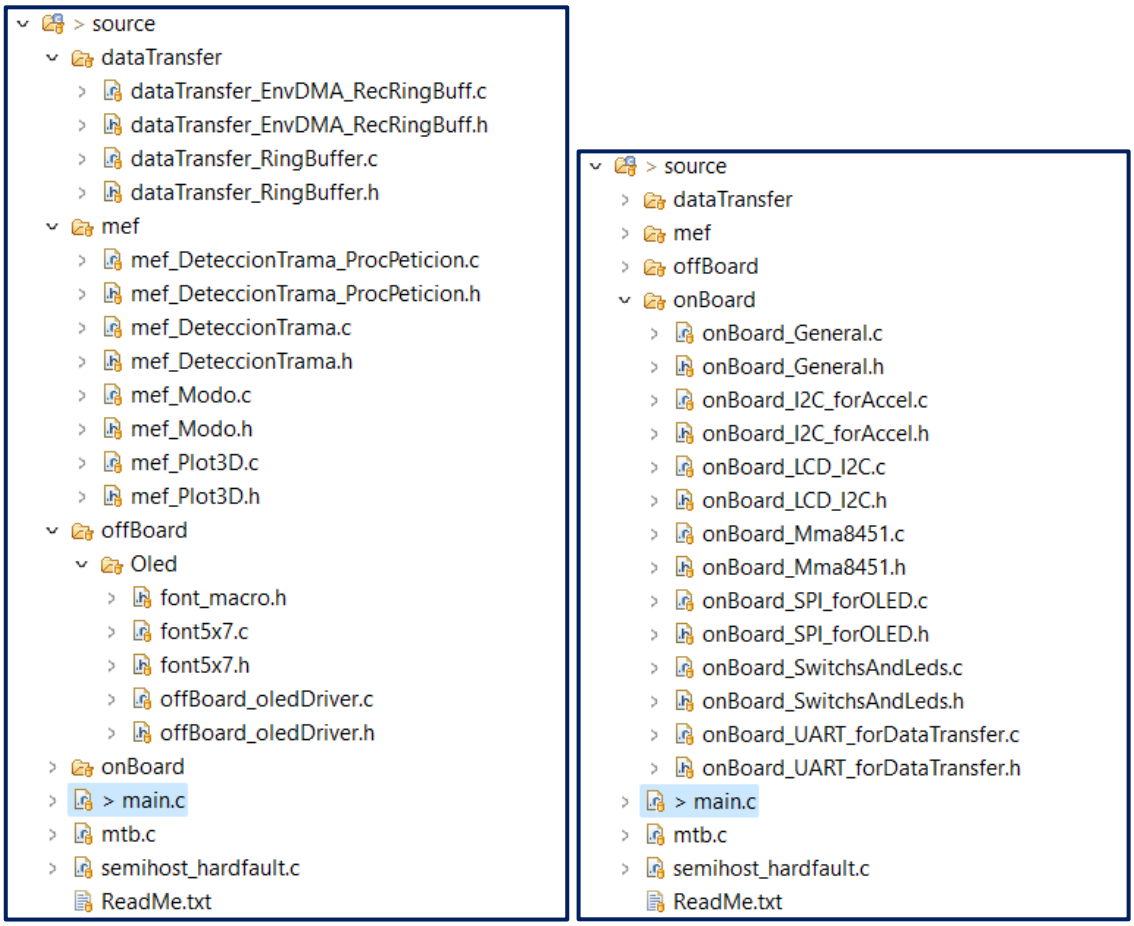


Figura 4.1: Organización de los archivos del proyecto.

En consecuencia, de haber segmentado y modularizado el código se obtuvo un archivo principal "main.c" corto y muy sencillo, como se muestra en la *Figura 4.2*.

main.c

```
**********GRAFICADOR DE ACELERACIÓN CON ENLACE INALÁMBRICO**********

#include <stdio.h>
#include <mef/mef_Modo.h>

int main(void){

    onBoard_General_init();                     //Inicializa la Board + Periféricos
    dataTransfer_EnvDMA_RecRingBuff_Init();     //Inicializa UART+DMA y crea el Buffer Circular
    mefModo_init();                             //Inicializa la Máquina de Estados Finitos (MEF)
    SysTick_Config(SystemCoreClock / 1000U);    //Configura interrupción de systick cada 1 ms

    while(1){
        mefModo();
    }
}

void SysTick_Handler(void) {                    //El SysTick interrumpe cada 1ms
    onBoard_SwitchsAndLeds_key_periodicTask1ms();
    mefModo_task1ms();
}

/*=========================================[end of file]=========================================*/
```

Figura 4.2: Archivo principal "main.c" del proyecto.

5. CONCLUSIÓN:

A la hora de concluir acerca del sistema desarrollado, se cree conveniente analizar si los objetivos fueron cumplidos. Pensando entonces en primer lugar sobre los objetivos funcionales básicos planteados en la introducción podemos decir que todos fueron cumplidos. En las *Figuras 5.1 y 5.2* se observa una prueba realizada sobre los modos de trabajo, TP3_MASTER y PLOT3D respectivamente, con comunicación inalámbrica y funcionalidad exitosa:

Figura 5.1: Software en pc "SD2_TP3_Master" ejecutando una prueba exitosa.

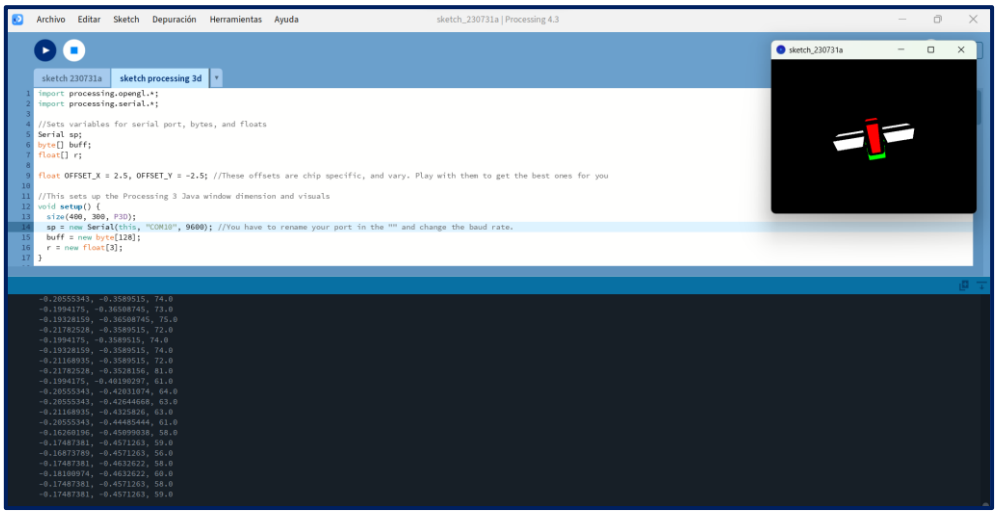


Figura 5.2: Software en pc "Plot3D" ejecutando una prueba exitosa.

Dentro de las problemáticas más desafiantes que se abordaron se destaca el trabajo realizado con el manejo prioritario de excepciones e interrupciones. También el uso de Buffers circulares y los módulos de acceso directo a memoria DMA para la transacción de mensajes.

Respecto a los desafíos propuestos en la segunda sección de este informe se puede objetar:

1. ***Tomar como punto de partida el trabajo ya desarrollado, esto es, no modificar el funcionamiento en general y reutilizar todo lo que se había desarrollado ya; aprovechando las virtudes, pero sin perder de vista los defectos, para no cometer los mismos errores. De esta manera poder enfocarse en mejorar cuestiones de implementación, perfeccionar la detección y el procesamiento de las tramas y optimizar el código al mismo tiempo que se documenta y organiza de una manera más profesional, ordenada y escalable todo el proyecto:***
   - ✓ Sobre este objetivo se trabajó arduamente logrando una implementación más sencilla y eficiente y un código mucho más consistente, ordenado, documentado, intuitivo y escalabre. Lógicamente sería imposible plasmar aquí cada cambio, el resultado final se puede visualizar en el repositorio de GitHub antes mencionado.

2. ***Reemplazar la comunicación cableada con la PC a través de protocolo RS485 por una comunicación inalámbrica a través de Radio Frecuencia (RF), manteniendo a UART como el protocolo para vincular el dispositivo con el transreceptor.***
   - ✓ Este objetivo fue cumplido haciendo uso de dos módulos APC200A de radiofrecuencia, como ya fue detallado en la sección de implementación de hardware.

3. ***Reducir el tiempo de latencia entre tramas enviadas y recibidas.***
   - ✓ Si bien fueron varias las medidas tomadas para optimizar el funcionamiento en este sentido, se cree conveniente hacer énfasis en una mejora particular que marcó una gran diferencia en cuanto al tiempo de respuesta del dispositivo. Se logró optimizar la lectura del acelerómetro aprovechando que los registros en donde se guardan las mediciones de los 3 ejes son contiguos. Antes se iniciaba un intercambio I2C para leer la medición de cada eje, haciendo uso de la función desarrollada para tal fin que se muestra a continuación en el *Código 5.1:*

```
static uint8_t onBoard_Mma8451_read_reg(uint8_t addr){
        i2c_master_transfer_t masterXfer;
        uint8_t ret;
        memset(&masterXfer, 0, sizeof(masterXfer));
        masterXfer.slaveAddress = MMA8451_I2C_ADDRESS;
        masterXfer.direction = kI2C_Read;
        masterXfer.subaddress = addr;
        masterXfer.subaddressSize = 1;
        masterXfer.data = &ret;
        masterXfer.dataSize = 1;
        masterXfer.flags = kI2C_TransferDefaultFlag;
        I2C_MasterTransferBlocking(I2C0, &masterXfer);
        return ret;}
```

Código 5.1: Función onBoard_Mma8451_read_reg( ).

En su lugar, se utiliza ahora una función optimizada, que iniciando una sola transacción I2C es capaz de recibir y montar en memoria la lectura de los 3 ejes, esto se muestra a continuación en el Código 5.2:

```
static void onBoard_Mma8451_read_multyReg(uint8_t addr, uint8_t *buff,
uint8_t size){
        i2c_master_transfer_t masterXfer;
        memset(&masterXfer, 0, sizeof(masterXfer));
        masterXfer.slaveAddress = MMA8451_I2C_ADDRESS;
        masterXfer.direction = kI2C_Read;
        masterXfer.subaddress = addr;
        masterXfer.subaddressSize = 1;
        masterXfer.data = buff;
        masterXfer.dataSize = size;
        masterXfer.flags = kI2C_TransferDefaultFlag;
        I2C_MasterTransferBlocking(I2C0, &masterXfer);
}
```

Código 5.2: Subrutina onBoard_Mma8451_read_multyReg( ).

4. ***Agregar la funcionalidad de mostrar en el display OLED, comunicado por protocolo SPI, el módulo de la aceleración con una tasa de refresco adecuada para una correcta visualización, al mismo tiempo que se mejoraban las funciones gráficas del mismo.***
   - ✓ Este objetivo fue cubierto haciendo uso de una biblioteca abierta que presta servicios gráficos para trabajar con displays de este tipo. De esta manera ahora se muestra en el display un mensaje inicial "ACELEROMETRO SD2" y se imprime el módulo de la aceleración (la raíz cuadrada de la suma de los cuadrados de sus componentes), con una tasa de refresco de 300ms. Además, abajo a la derecha se muestra el modo de operación actual, Master o Plot3D. Todo esto se puede visualizar a continuación en la *Figura 5.3.*

Figura 5.3: Salidas en display OLED indicando la aceleración y el modo de trabajo.

5. ***Explorar el uso del display de cristal líquido presente en la placa de desarrollo y comunicado al micro por protocolo I2C, mostrando los mensajes "SASE" y "2023" de manera alternada.***
   - ✓ Este objetivo fue alcanzado haciendo uso de algunas funciones de biblioteca, imprimiendo el hexadecimal "5A5E" en lugar de "SASE" y luego "2023" de manera alternada cada 1 segundo, como se muestra en la *Figura 5.4.*

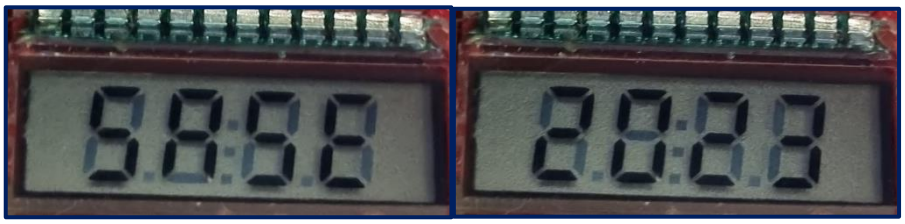


Figura 5.4: Salidas en display LCD con los mensajes "SASE" y "2023"

Se han manejado diversos protocolos de comunicación "onboard" y "offboard", cableados e inalámbricos. Se ha controlado múltiples periféricos de entrada y de salida, imprimiendo por pantalla en displays de distintas tecnologías. Se han procesado tramas, haciendo uso de colas circulares y módulos DMA, con mensajes para comandar o encuestar el estado de los periféricos. Con todos los objetivos propuestos cumplidos, se concluye que el proyecto ha sido llevado a cabo con rotundo éxito.

-Para finalizar, resulta valedero realizar un balance personal del camio recorrido a lo largo de todo el proyecto. La creación del dispositivo ha sido sin duda un logro significativo, pero su valor va más allá de su materialidad. La verdadera esencia de esta experiencia radica en el proceso de desarrollo, el cual ha sido una etapa llena de aprendizaje y crecimiento.

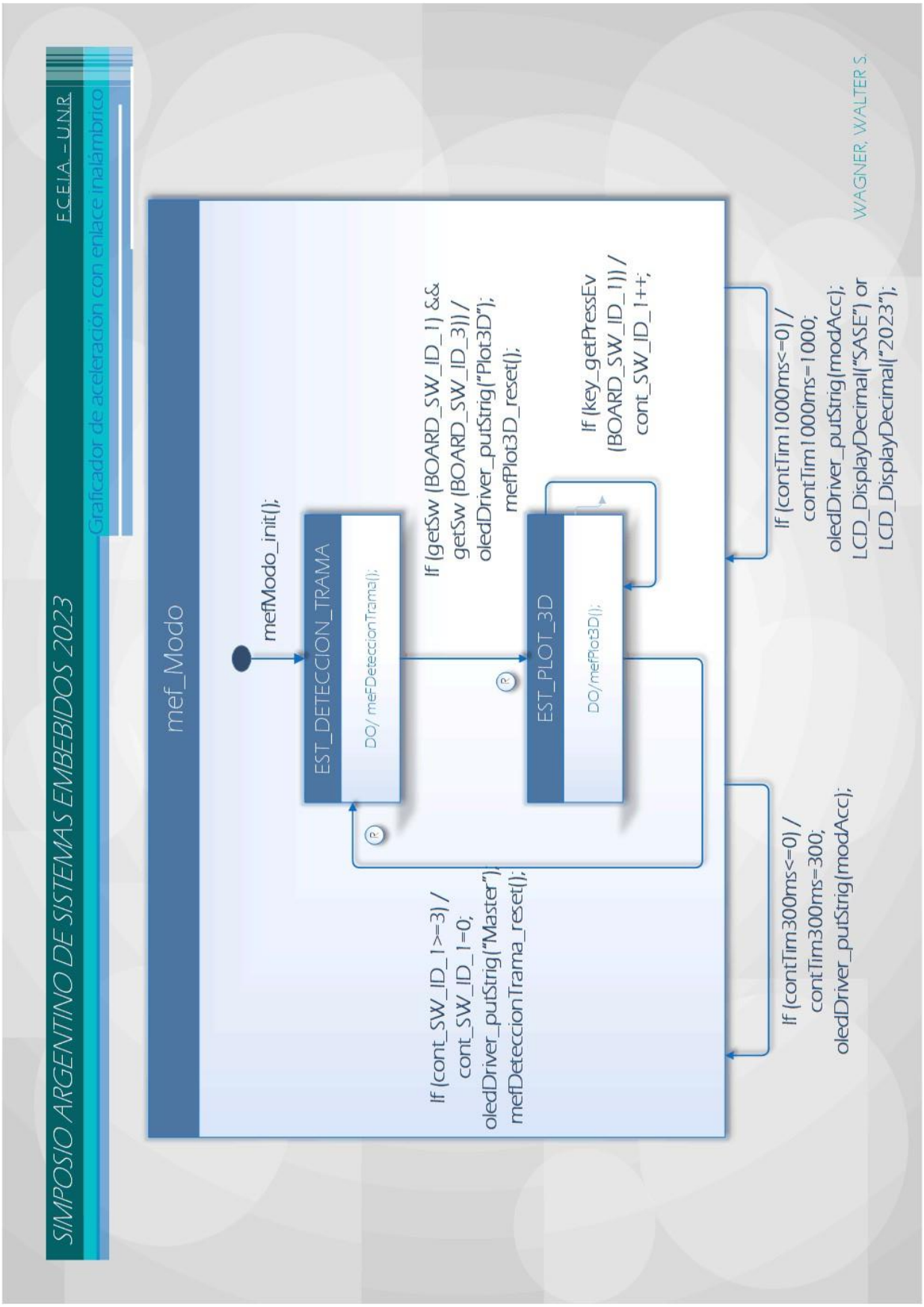
Desde sus inicios en la materia de Sistemas Digitales 2, donde el proyecto comenzó a tomar forma, hasta su fase de optimización para ser presentado en el prestigioso Simposio Argentino de Sistemas Embebidos 2023, cada paso fue un desafío que permitió adquirir nuevos conocimientos. Durante todo este trayecto, tuve la invaluable oportunidad de acercarme al ámbito profesional, guiado por un docente tutor excepcional como el Ing. Gustavo Muro, quien no dudó en destinar de su tiempo para compartir sus conocimientos y experiencia.

Sin dudas uno de los momentos más enriquecedores aún está por venir y será la participación en la competencia estudiantil dentro del SASE 2023. Esta oportunidad no solo me permitirá mostrar este trabajo, sino también aprender de otros proyectos y perspectivas. "¡Nunca dejemos de aprender y superarnos en el fascinante universo de los Sistemas Embebidos!" -

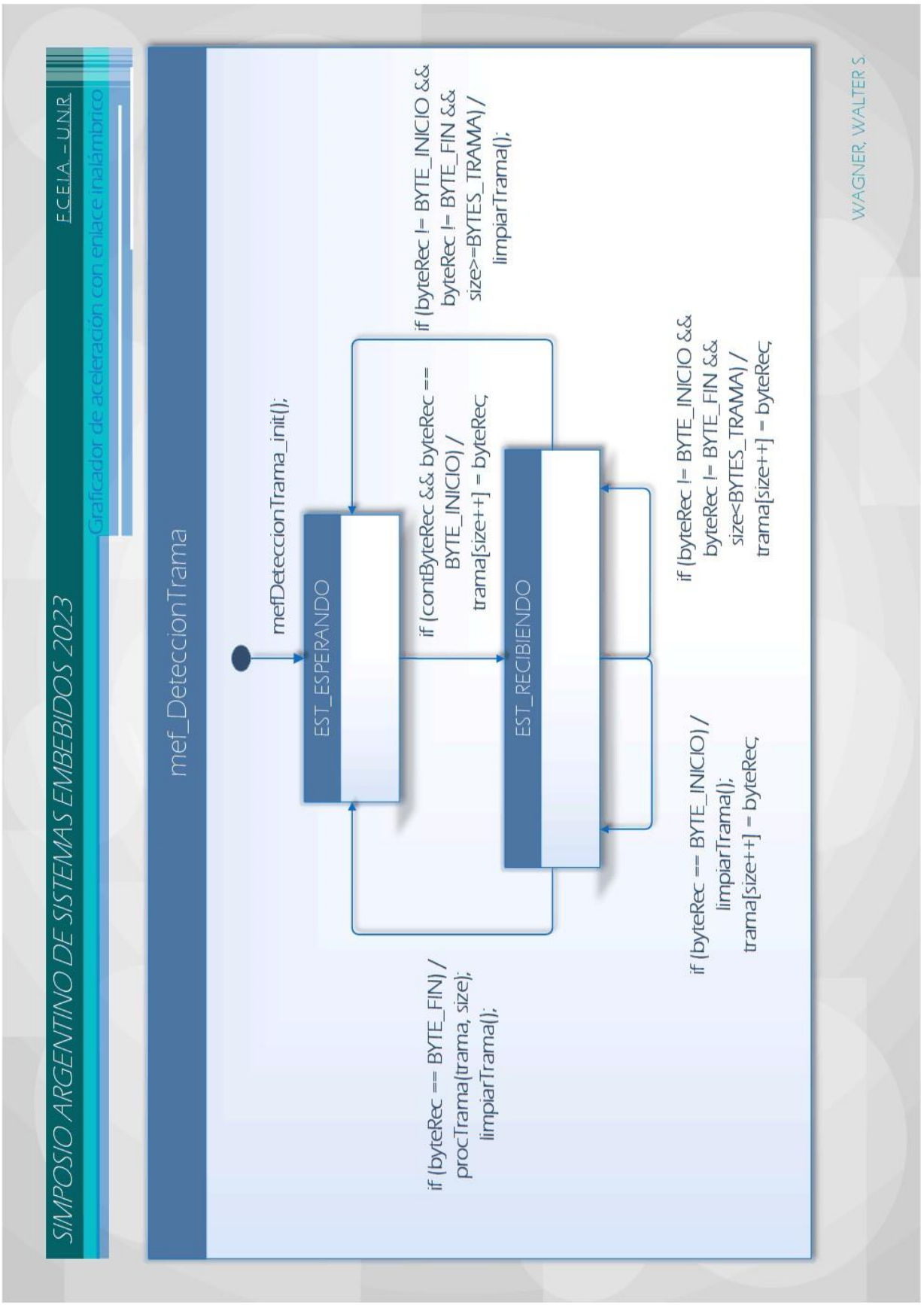
6. BIBLIOGRAFÍA:

- The definitive guide to CORTEX-M0 and CORTEX-M0+ processors. Joseph Yiu. Elsevier Inc. 2° Edición, 2015.
- Fundamentals of System-on-Chip Design on Arm Cortex-M Microcontrollers. René Beuchat et al. ARM Education Media. 1ª Edición, 2021.
- Desarrollo con microcontroladores ARM. CORTEX-M3. Sergio Caprile. Cika Electrónica. 1ª Edición, 2013.
- Practical UML Statecharts in C/C++. Event-Driven Programming for Embedded Systems. Miro Samek. Elsevier. 2° Edición, 2009
- Manual y esquemático del microcontrolador FRDM-KL46z extraídos del sitio del fabricante.
- Hoja de datos TTL to RS485.

7. ANEXO I: MEF Modo:



8. ANEXO II: MEF DeteccionTrama



9. ANEXO III: MEF Plot3D