Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Eléctrica

Máquinas de Estado

Tarea 4

Integrantes: Leslie Cárdenas
Nicolás Ruminot
Profesor: José González
Fecha de realización: 9 de julio de 2023
Fecha de entrega: 9 de julio de 2023
Santiago de Chile

Índice de Contenidos

Índice de Figuras

1. Diagrama de Capas

En esta sección, se definen las funcionalidades correspondientes a cada capa según el modelo de capas OSI mostrado en la Figura 1. Se detallan las funciones que se encuentran en cada capa, teniendo en cuenta que algunas funciones pueden abarcar más de una capa debido a su naturaleza transversal.

Además, se mencionan funciones de nivel más bajo, como la asignación de intensidades de corrientes bajas para las señales y la definición de niveles de voltaje para representar un bit, entre otras. Estas funciones se consideran fundamentales en el funcionamiento del sistema, aunque no se profundizará en su descripción en este momento.

Es importante destacar que las funciones que "vive" en más de una capa desempeñan un papel crucial en la comunicación y el procesamiento de datos entre las diferentes capas. Estas funciones actúan como enlaces entre las capas, permitiendo la transferencia de información y asegurando la correcta interacción entre ellas.

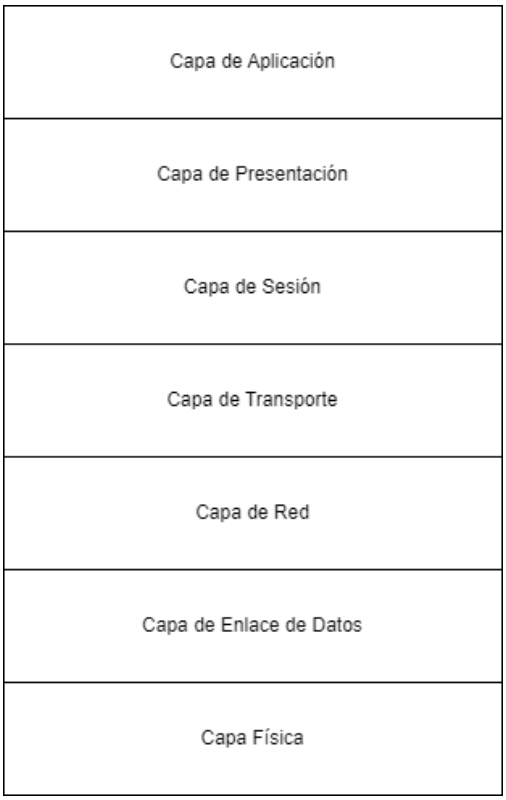

Figura 1: Diagrama de capas.

A continuación, se presentaron las funciones principales que se encuentran en cada una de las capas del modelo OSI, así como las funciones específicas que se utilizan en nuestra aplicación y en qué capa se emplean. Esto nos permitió comprender mejor la distribución de las funciones en el contexto del modelo OSI y cómo se aplican en nuestro sistema.
Especificaciones generales del modelo OSI por capas:

- Capa 1: Capa física
  La capa física se encarga de la transmisión y recepción de bits sin procesar a través del medio de comunicación físico. Define las características eléctricas, mecánicas y funcionales de los dispositivos de red, como cables y tarjetas de red.

- Capa 2: Capa de enlace de datos
  La capa de enlace de datos se encarga de la transferencia de datos confiable entre dos nodos adyacentes en la red. Proporciona detección y corrección de errores, control de flujo y control de acceso al medio.

- Capa 3: Capa de red
  La capa de red se encarga del enrutamiento de los paquetes de datos a través de la red. Es responsable de establecer rutas y determinar la mejor manera de enviar los datos desde el origen al destino.

- Capa 4: Capa de transporte
  La capa de transporte proporciona servicios de transporte confiables y transparentes para los datos. Se encarga de dividir los datos en segmentos, establecer conexiones y asegurar la entrega confiable de los datos.

- Capa 5: Capa de sesión
  La capa de sesión establece, administra y finaliza las conexiones de comunicación entre las aplicaciones. Proporciona servicios de control de diálogo y sincronización para garantizar una comunicación adecuada entre las aplicaciones.

- Capa 6: Capa de presentación
  La capa de presentación se encarga de la representación y el formato de los datos transmitidos. Realiza tareas como la compresión y el cifrado de los datos para garantizar una correcta interpretación por parte de las aplicaciones receptoras.

- Capa 7: Capa de aplicación
  La capa de aplicación proporciona servicios de red a las aplicaciones finales. Incluye protocolos para servicios como el correo electrónico, transferencia de archivos, acceso remoto y navegación web.

Especificaciones de nuestra aplicación por capas del modelo OSI:

- Capa 1: Capa Física
  La mayoría de las funciones trabaja en más de una capa. Funciones de alto nivel, como llamar a Watch Dog Timer (LowPower.powerDown()) para entrar y salir del modo sleep, así como funciones como pinMode(), Serial.write(), getDistance(), entre otros. Principalmente en el estado Lora descrito en la siguiente sección se hace uso de funciones de más bajo nivel que interactúan con la capa física y en los demás estados se hace uso de funciones de alto nivel para controlar los Led, guardar datos en la memoria y utilizar el modo sleep.

- Capa 2: Enlace de Datos
  En esta capa encontramos funciones de alto nivel como Serial.write(), y funciones de más bajo nivel como la resolución de colisiones proporcionada por el chip LoRaWAN, acceso al medio y control de errores.

- Capa 3: Capa de Red
  En esta capa se define la topología de la red, en este caso punto a punto y las funciones son de bajo nivel.

- Capa 5: Capa de Sesión:
  Aquí se establece la conexión entre los dispositivos y se sincronizan entre sí. Las funciones utilizadas aquí son proporcionadas por LoRaWAN y son una mezcla entre alto y bajo nivel.

- Capa 6: Capa de Presentación
  Es utilizada por funciones de alto nivel que permiten la interpretación de los datos, así como funciones de bajo nivel como los cifrados incorporados por LoRaWAN (AES128). Se interpretan las alarmas levantadas por los errores para comunicarlos a través de la capa de aplicación.

- Capa 7: Capa de Aplicación
  Aquí encontramos las funciones de más alto nivel que permiten visualizar los datos por parte del usuario, como formatos de bases de datos o la interacción con los LEDs de la placa, para que un técnico pueda entender lo que está ocurriendo dentro del hardware.

2. Máquina de Estados

En esta sección se presentan los diagramas de estado utilizados para desarrollar la implementación en software. Contamos con una máquina de estados principal que consta de 6 estados y varios subestados.

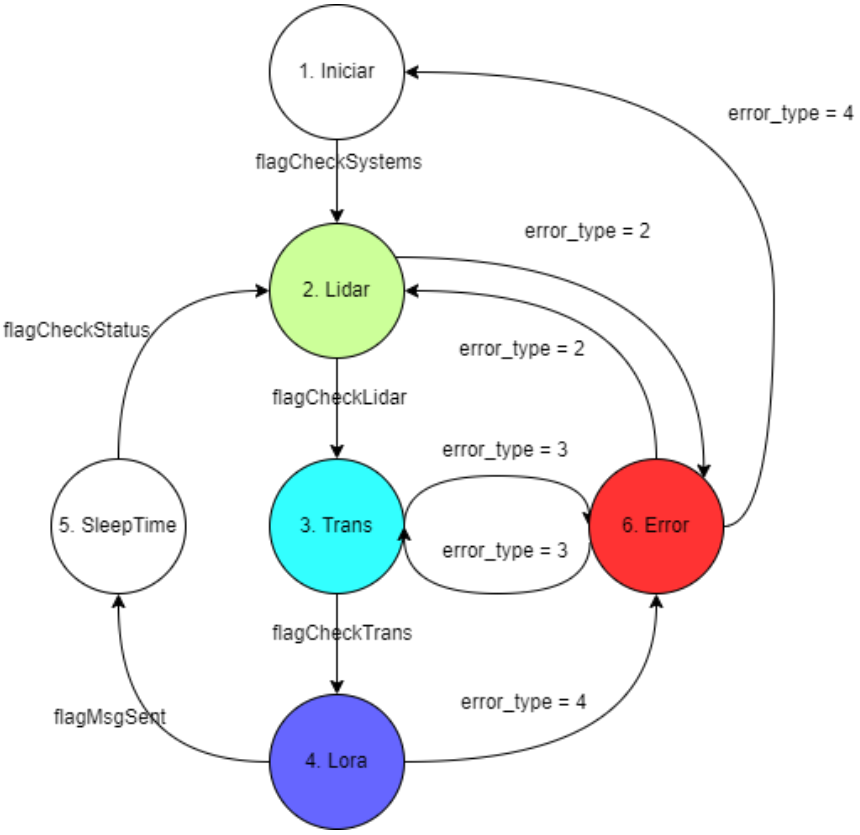

Figura 2: Máquina de estados principal.

2.1. Máquina Lidar

Esta máquina de estados muestra el comportamiento del software encargado de controlar los sensores Lidar. Principalmente se declaran las variables en Iniciar, se le pide la información sobre el nivel del combustible a los sensores Lidar1 y Lidar2 en los estados consecutivos “Midiendo 1 y 2", se calcula la diferencia entre estas distancias y se pasa al siguiente estado intermedio “Guardar1". Para esto se hace uso principalmente de la función getDistance() perteneciente a la librería “TFMini.h” utilizada para el control de estos sensores.

Además se tiene que si las mediciones retornadas por estos sensores son -1, eso se interpreta como un error de medición. Tras 5 errores de medición consecutivos del mismo sensor se manda un mensaje de error.

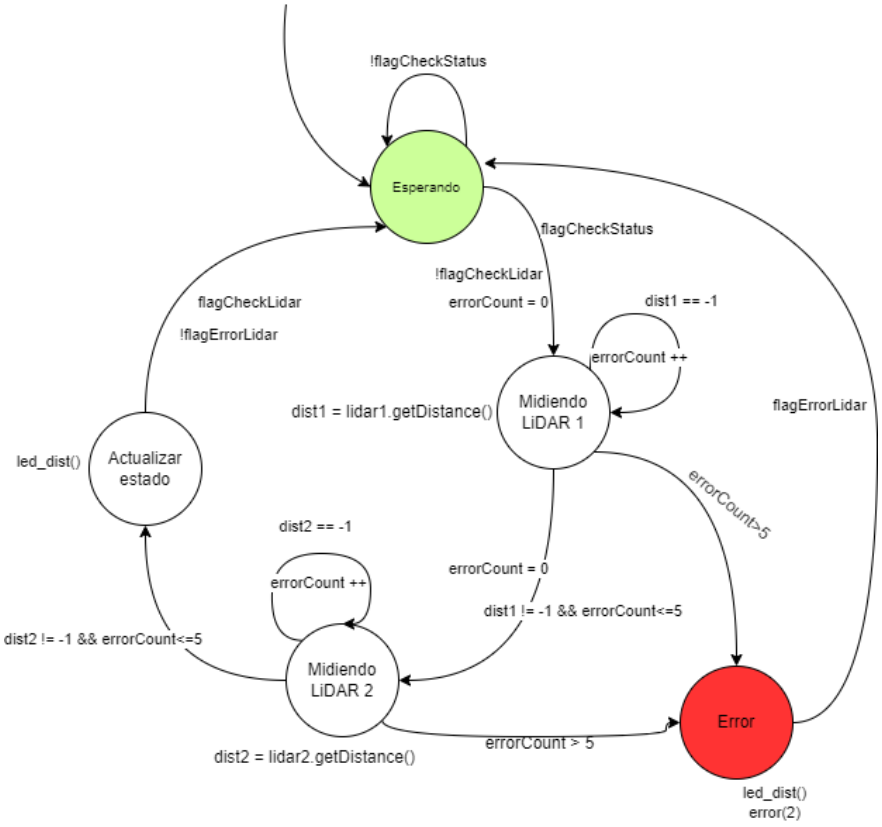

Figura 3: Máquina de estados para los sensores Lidar.

2.2. Máquina de Led LiDAR

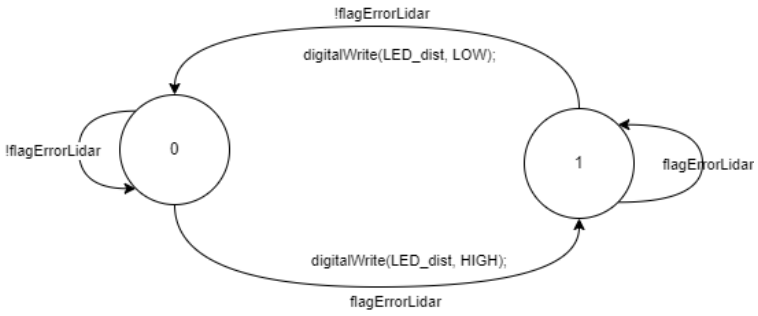

Figura 4: Máquina de estados para los led asociados al LiDAR.

2.3. Máquina Trans

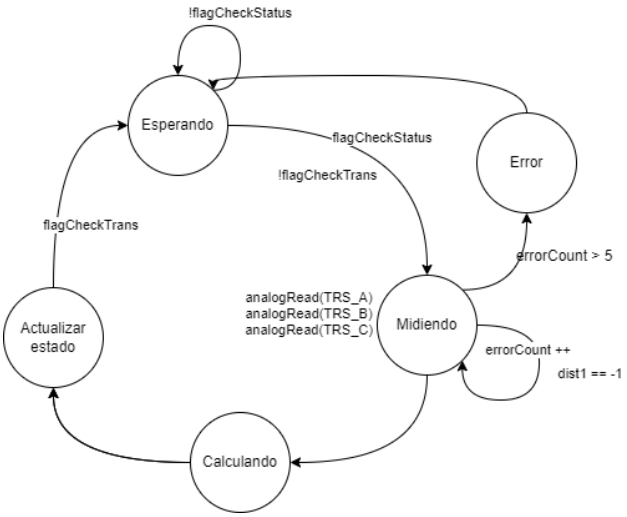

Figura 5: Máquina de estados para el estado Trans.

2.4. Máquina Lora

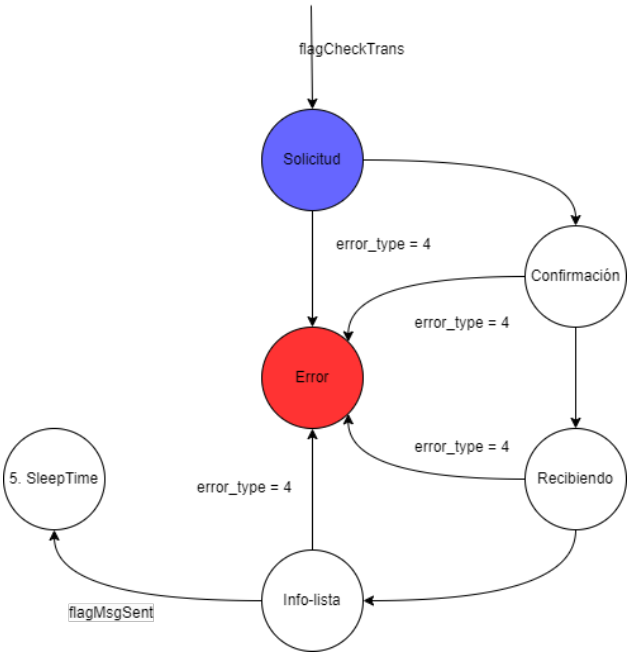

Figura 6: Máquina de estados para el estado Lora.

2.5. Máquina Error

La siguiente máquina de estados recibe como entrada la ubicación del error y lo guarda en la memoria EEPROM (memoria no volátil). Esto se hace con el propósito de permitir que, en caso de una falla en la comunicación a través de LoRaWAN, el técnico pueda acceder al registro de errores incluso si el dispositivo se queda sin energía.

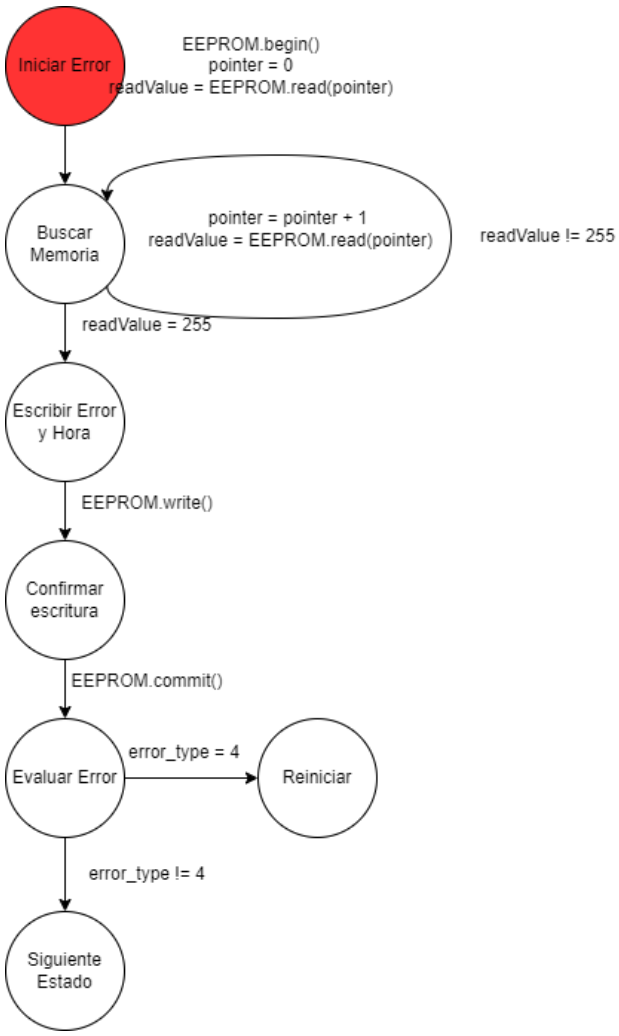

Figura 7: Máquina de estados general para el estado de Error.

2.6. Máquina SleepMode

Esta máquina se ejecuta después de cada ciclo de mediciones, poniendo al procesador en un estado de bajo consumo durante 1 minuto.

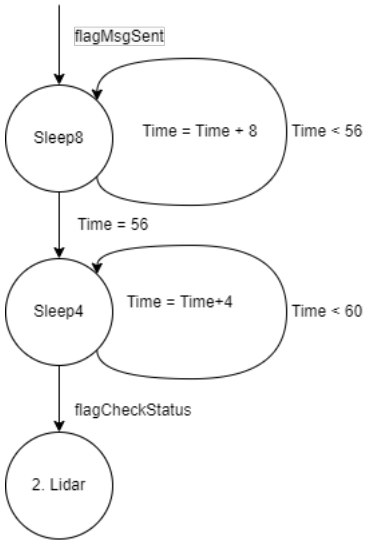

Figura 8: Máquina de estados para el estado Sleep.

3. Diagrama de bloques principal

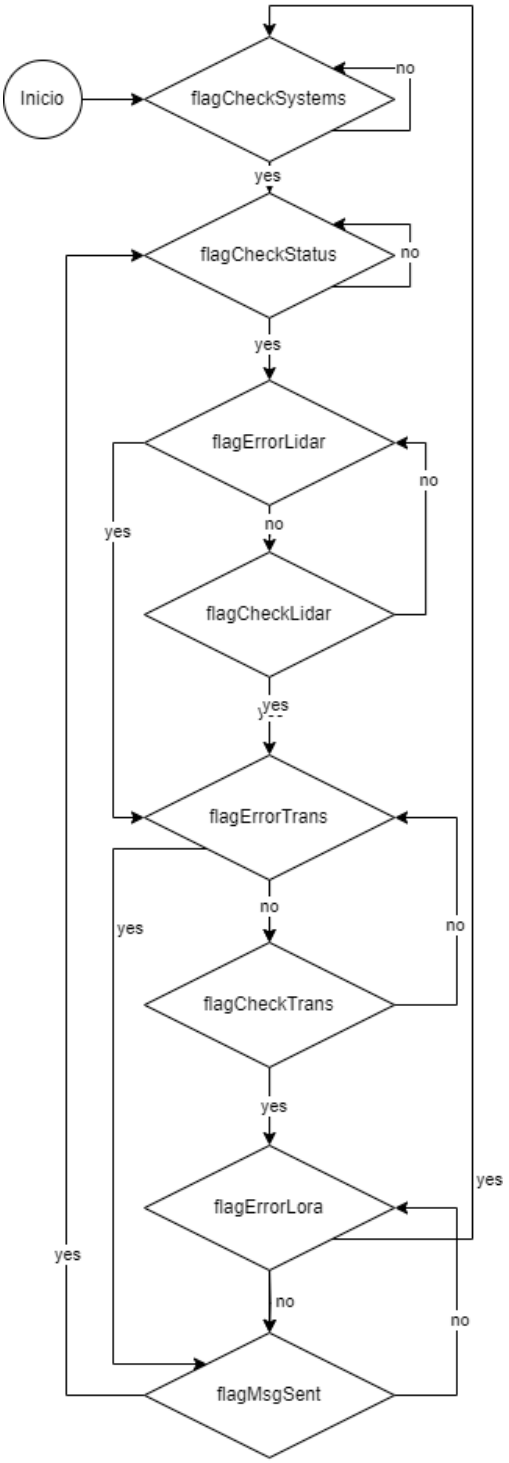

Figura 9: Diagrama de bloques principal.

4. Tiempo invertido en la Tarea

La tarea de programación resultó ser bastante extensa y compleja. Aunque comenzamos por crear los diagramas de bloque con la intención de aclarar nuestras ideas, descubrimos que no siempre reflejaban fielmente lo que terminábamos programando. Esto nos llevó a tener que alternar constantemente entre las máquinas de estado y el código, lo que resultaba en un proceso ineficiente.

En promedio, dedicamos aproximadamente 2 horas para desarrollar cada máquina de estado, pero esto estaba sujeto a posteriores modificaciones. En total, invertimos 18 horas por persona en el desarrollo del código, además de 2 horas adicionales para asegurarnos de que todo estuviera correcto.

Esta experiencia nos permitió aprender que la planificación y la adaptación son fundamentales en el proceso de programación, ya que las ideas iniciales pueden cambiar a medida que nos adentramos en el desarrollo del código.

Máquina de Estado	**Integrante 1**	**Integrante 2**
Main	2	2
Iniciar	-	-
Lidar	4	1
Trans	2	1
Lora	1	2
SleepTime	-	1
Error	1	3
LedLidar	2	1