DEPARTAMENTO DE
INGENIERÍA
MECÁNICA

UNIVERSIDAD DE SANTIAGO DE CHILE

Facultad de Ingeniería
Departamento de Ingeniería Mecánica
Carrera de Ingeniería Civil Mecatrónica

Proyecto Ingeniería de Software Interdisciplinaria

Sistema de validación de rendimiento de una bobina Tesla

Integrantes:

Nombre	RUT	Correo electrónico
Matías Khol	21.378.326-2	matias.khol@usach.cl
Alejandro Alarcón	21.813.314-2	alejandro.alarcon.p@usach.cl
Pablo Canales	21.405.022-6	pablo.canales.p@usach.cl
Daniel González	20.239.382-9	daniel.gonzalez.i@usach.cl

Profesor: Daniel Gacitúa / Ricardo Hasbún

Asignatura: Ingeniería de Software Interdisciplinaria

Fecha de entrega: 17 de Diciembre de 2025

Índice

1. Introducción

El presente proyecto se desarrolla en el contexto de la asignatura de Ingeniería de Software y corresponde al diseño e implementación de un sistema mecatrónico que integra hardware y software para el monitoreo y análisis de variables físicas.

En particular, el proyecto se enfoca en el estudio experimental de una bobina Tesla, seleccionada como sistema mecatrónico de referencia, con el objetivo de analizar su comportamiento electromagnético mediante la adquisición y visualización de datos obtenidos desde un sistema de monitoreo externo. El sistema propuesto busca facilitar la medición y validación del funcionamiento de este tipo de dispositivos, abordando la dificultad existente para caracterizar experimentalmente bobinas Tesla u otros sistemas electromagnéticos similares.

El desarrollo del proyecto se realiza siguiendo principios de ingeniería de software, utilizando la metodología ágil Scrum y un enfoque incremental. A nivel de diseño del software, se emplea una arquitectura Modelo–Vista–Controlador (MVC), orientada a organizar de forma estructurada la lógica del sistema, la interfaz de usuario y el control de la interacción.

Este informe presenta el desarrollo correspondiente al Sprint 1 del proyecto, en el cual aborda la definición del sistema, la especificación de requisitos, el diseño inicial y la planificación del trabajo. En esta etapa no se considera aún la integración física del hardware, centrándose el avance en la construcción conceptual del sistema y en la generación de la estructura base del software.

2. Objetivo y Alcance del Sprint 1

2.1. Objetivo del Sprint

El objetivo del Sprint 1 es definir y diseñar la base del proyecto, estableciendo los requisitos del sistema, la estructura general del software y la planificación del desarrollo. Este sprint sienta los fundamentos del proyecto desde el punto de vista de la ingeniería de software, sin considerar aún la implementación ni la integración física del hardware.

2.2. Alcance del Sprint

El alcance del Sprint 1 comprende la definición y el diseño inicial del sistema desde el punto de vista de la ingeniería de software. En esta etapa se desarrollan la especificación de requisitos (SRS), los diagramas UML, la planificación del proyecto mediante artefactos Scrum y la definición de la estructura base del repositorio de desarrollo.

Durante este sprint se describe el funcionamiento general del sistema, las variables que serán medidas, las relaciones entre los componentes y las limitaciones del sistema. El hardware se considera únicamente a nivel conceptual, identificando los componentes principales involucrados, tales como sensores, actuadores y la bobina Tesla, sin abordar aún su implementación física ni su conexión eléctrica o electrónica.

Quedan explícitamente fuera del alcance del Sprint 1 la construcción de la maqueta, la integración real del hardware, las pruebas físicas del sistema y la validación experimental, las cuales serán abordadas en sprints posteriores del proyecto.

3. Especificación de Requisitos del Software (SRS)

3.1. Propósito del sistema

El propósito del sistema es proporcionar una herramienta de software capaz de caracterizar el comportamiento de una bobina Tesla mediante la adquisición, visualización y análisis de variables físicas asociadas a su funcionamiento.

El sistema está orientado al monitoreo del entorno de operación de la bobina, permitiendo comparar los datos medidos con valores teóricos esperados, con el objetivo de apoyar procesos de validación experimental y análisis académico.

El software está diseñado para ser utilizado por estudiantes, académicos o investigadores interesados en el estudio experimental de bobinas Tesla u otros sistemas de alta tensión similares, en contextos de laboratorio o investigación.

El sistema no controla directamente la operación de la bobina Tesla. Su funcionamiento es independiente del sistema de generación de la descarga, limitándose a la medición del entorno mediante sensores externos y al procesamiento de dicha información dentro del software.

En el contexto del Sprint 1, el sistema se encuentra en una etapa inicial de definición y estructuración, en la cual se establecen las funcionalidades fundamentales, las variables de interés y el flujo general de operación del software. Esta versión considera solo una parte de las funcionalidades finales del sistema, enfocándose en la definición de las variables relevantes y en la implementación de una visualización básica de los datos medidos, la cual servirá como base para el desarrollo de nuevas funcionalidades en sprints posteriores.

3.2. Alcance del sistema

El sistema corresponde a una solución combinada de hardware y software orientada al monitoreo y análisis de variables físicas asociadas al funcionamiento de una bobina Tesla, con el objetivo de apoyar la validación experimental de su comportamiento.

Dentro de su alcance, el sistema considera la medición de variables como el voltaje asociado a la señal de radiofrecuencia mediante un sensor RF, el voltaje medido por un fotodiodo asociado a la intensidad lumínica de la descarga, el ángulo de un servomotor utilizado para posicionar el sensor RF, y el voltaje de entrada aplicado a la bobina Tesla, adquirido mediante el microcontrolador para su utilización en el análisis teórico del sistema.

El usuario puede interactuar con el sistema físico modificando el ángulo del servomotor para desplazar el sensor RF, y con el sistema de software mediante la visualización de

gráficos en tiempo real de las variables medidas, el acceso al historial de datos y la realización de cálculos básicos, como la comparación entre valores teóricos y experimentales.

Quedan explícitamente fuera del alcance del sistema el control directo de la bobina Tesla y su sistema de alimentación, los cuales operan de manera externa al sistema de monitoreo. El sistema se limita a la adquisición y procesamiento de las variables medidas, sin intervenir en la lógica de encendido, apagado o regulación de la bobina.

En el contexto del Sprint 1, el alcance del proyecto se limita a la definición de los requisitos del sistema, el diseño inicial mediante diagramas UML, la planificación del trabajo mediante metodología Scrum y la construcción de una estructura base de software. La integración con la maqueta física y las pruebas experimentales del sistema de monitoreo se abordan en sprints posteriores.

3.3. Definiciones, acrónimos y abreviaturas

Término	**Definición**
RF	Señal de radiofrecuencia asociada al campo electromagnético generado por la bobina Tesla.
Fotodiodo	Sensor utilizado para convertir la intensidad lumínica de la descarga en una señal de voltaje medible por el sistema.
MVC	Modelo de arquitectura de software utilizado para estructurar el sistema, separando la lógica, la interfaz y el control.
UML	Lenguaje gráfico utilizado para representar el diseño de un sistema orientado a objetos.
SRS	Documento que especifica los requisitos y el alcance del sistema de software.
Sprint	Periodo corto y fijo de tiempo asociado a la metodología Scrum, en el cual se desarrolla un incremento del sistema.

3.4. Descripción general del sistema

3.4.1. Contexto y perspectiva del producto

El sistema corresponde a una solución independiente de monitoreo y análisis, diseñada para operar de manera externa a la bobina Tesla. Si bien el sistema es autónomo en su funcionamiento actual, su diseño permite una posible escalabilidad a futuro.

El sistema se comunica con sensores encargados de medir variables físicas del entorno de la bobina, y con un controlador que procesa las señales adquiridas y envía la información al computador. El software principal se ejecuta en un notebook, desde el cual se visualizan y analizan los datos obtenidos.

3.4.2. Funciones generales del producto

Las funciones generales del sistema son las siguientes:

- Medición de variables físicas asociadas al funcionamiento de la bobina.
- Visualización de gráficos en tiempo real de las variables medidas.
- Control del movimiento de un servomotor para posicionar el sensor RF.
- Registro y almacenamiento de datos de medición.
- Análisis y comparación de los resultados obtenidos.

3.4.3. Actores del sistema

Los actores que interactúan con el sistema son:

- **Usuario**: Persona que utiliza el sistema para realizar mediciones, visualizar resultados y analizar los datos obtenidos.
- **Controlador**: Dispositivo encargado de procesar las señales de voltaje provenientes de los sensores, enviar los datos al notebook y ejecutar el movimiento del servomotor.

3.5. Casos de uso

- **UC-01: Medición de variables**
  - **Actor:** Usuario
  - **Gatillo:** El usuario enciende el sistema de monitoreo.
  - **Resultado esperado:** El sistema mide las variables físicas del entorno de la bobina (voltaje RF y voltaje Fotodiodo, etc.).
- **UC-02: Visualización de gráficos**
  - **Actor:** Usuario
  - **Gatillo:** El usuario selecciona el tipo de gráfico a visualizar.
  - **Resultado esperado:** El sistema genera y muestra el gráfico correspondiente con los datos medidos en tiempo real.
- **UC-03: Registro de datos**
  - **Actor:** Usuario
  - **Gatillo:** El usuario presiona el botón para guardar los datos.
  - **Resultado esperado:** El sistema guarda los datos en un archivo o base de datos, permitiendo su acceso posterior.
- **UC-04: Análisis de datos**
  - **Actor:** Usuario
  - **Gatillo:** El usuario solicita un análisis comparativo entre los valores teóricos y los valores experimentales.
  - **Resultado esperado:** El sistema calcula y muestra el error entre los valores teóricos y experimentales.
- **UC-05: Movimiento del servo**
  - **Actor:** Usuario
  - **Gatillo:** El usuario selecciona un nuevo ángulo para mover el servo.
  - **Resultado esperado:** El sistema mueve el servo a la posición deseada, ajustando la distancia entre el sensor RF y la bobina.

3.6. Requisitos funcionales

RF-1: El sistema debe medir la señal de radiofrecuencia mediante un sensor RF y la intensidad lumínica de la descarga eléctrica mediante un fotodiodo.

RF-2: El sistema debe visualizar en tiempo real los valores medidos de las variables físicas, permitiendo al usuario observar la evolución de los datos durante la operación del sistema.

RF-3: El sistema debe registrar los datos de salida de los sensores, incluyendo los voltajes medidos, el tiempo asociado a cada medición y los valores calculados de intensidad de campo electromagnético e intensidad lumínica.

RF-4: El sistema debe calcular, a partir de los datos medidos, la intensidad del campo electromagnético y la intensidad lumínica de la descarga eléctrica.

RF-5: El sistema debe permitir el control del ángulo de un servomotor para posicionar el sensor RF, posibilitando el acercamiento o alejamiento del loop respecto de la bobina Tesla.

3.7. Requisitos no funcionales

RNF-1: El sistema debe estar desarrollado utilizando el lenguaje de programación Python.

RNF-2: El sistema debe estar estructurado siguiendo una arquitectura Modelo–Vista–Controlador (MVC), separando la lógica del sistema, la interfaz de usuario y el control de la interacción.

RNF-3: El sistema debe actualizar la visualización de los datos medidos con un retardo máximo de un segundo, para permitir un análisis cercano al tiempo real.

RNF-4: La interfaz del sistema debe estar orientada a usuarios con conocimientos técnicos del área, incorporando elementos y conceptos técnicos, pero manteniendo una interacción simple y comprensible.

RNF-5: El sistema debe poder ejecutarse en un computador personal o notebook, sin requerir configuraciones especiales de hardware adicionales al sistema de monitoreo.

3.8. Restricciones

El sistema se encuentra sujeto a las siguientes restricciones técnicas y operacionales:

- El software debe ser desarrollado utilizando el lenguaje de programación Python.
- El sistema depende del uso de sensores específicos, un microcontrolador y un computador personal o notebook para la adquisición, procesamiento y visualización de los datos.

- El sistema no realiza control automático sobre la bobina Tesla. La regulación del voltaje de entrada se efectúa de manera manual mediante un regulador externo, mientras que el sistema adquiere y registra el voltaje aplicado a la bobina para su utilización en el análisis y la validación teórica del comportamiento del sistema.

- El tiempo de operación del sistema de monitoreo se encuentra limitado por el voltaje de alimentación aplicado a la bobina Tesla, de modo de evitar su sobrecalentamiento durante la operación a altos niveles de tensión.

- El sistema debe operar considerando un perímetro de seguridad alrededor de la bobina Tesla, evitando la presencia de dispositivos electrónicos o elementos metálicos que puedan generar interferencias o representar un riesgo durante las mediciones.

- El desarrollo del proyecto debe seguir la metodología ágil Scrum, organizando el trabajo en sprints incrementales.

- Las pruebas experimentales del sistema se encuentran limitadas por condiciones de seguridad, tiempo de exposición y entorno de uso del sistema.

- La validación del sistema se realiza mediante la comparación entre los resultados experimentales obtenidos y el comportamiento teórico esperado de la bobina Tesla, considerando las características conocidas del sistema

- El sistema considera condiciones básicas de seguridad durante su operación, limitando el tiempo de medición y el uso de altos niveles de voltaje en la bobina Tesla. En esta etapa del proyecto, dichas condiciones se definen a nivel conceptual y procedimental, sin implementar aún mecanismos automáticos de protección o bloqueo en el software.

3.9. Criterios de aceptación y verificabilidad

Esta sección tiene como objetivo definir los criterios mediante los cuales se validará que los requisitos y el diseño del sistema han sido correctamente definidos, permitiendo avanzar de manera fundamentada hacia las etapas de diseño detallado e implementación del software.

La verificación del sistema se enfoca en comprobar que el comportamiento esperado, las funciones definidas y las relaciones entre los componentes sean coherentes con las especificaciones establecidas en las secciones anteriores del documento. Para ello, se consideran criterios de aceptación asociados a cada requisito funcional, los cuales permiten evaluar su correcto cumplimiento.

En el caso de la medición de variables físicas, se considera que el sistema cumple con el requisito cuando los valores obtenidos se encuentran dentro de los rangos esperados definidos por las especificaciones técnicas de los sensores y por los modelos teóricos asociados al funcionamiento de la bobina Tesla.

La visualización de los datos se considera válida cuando las curvas generadas por el sistema presentan un comportamiento coherente con el modelo teórico esperado, permitiendo

la comparación visual entre los resultados teóricos y experimentales.

El registro de datos se valida mediante la ejecución de pruebas controladas en las cuales el sistema almacena información durante un periodo definido, verificando posteriormente la integridad y consistencia de los datos guardados.

El análisis y los cálculos realizados por el sistema se consideran correctos cuando las ecuaciones implementadas corresponden a modelos matemáticos validados, extraídos de referencias teóricas o técnicas, y los resultados obtenidos son consistentes con dichos modelos.

Finalmente, el funcionamiento del movimiento del servomotor se valida mediante pruebas realizadas con la bobina Tesla desenergizada, verificando que el sistema permite modificar la posición del sensor de forma controlada y segura antes de operar bajo condiciones reales.

4. Arquitectura del sistema (Sprint 1)

4.1. Flujo Sensor–Controlador–Actuador

El flujo de funcionamiento del sistema comienza cuando se inicia el proceso de medición. En esta etapa, los sensores de radiofrecuencia (RF) y el fotodiodo adquieren las señales físicas del entorno de la bobina Tesla y entregan sus voltajes de salida al microcontrolador. De forma paralela, se registra el voltaje de entrada aplicado a la bobina, el cual es utilizado posteriormente como referencia para los cálculos teóricos del sistema.

El microcontrolador recibe los voltajes correspondientes a las señales medidas y los envía al software que se ejecuta en el notebook mediante un enlace de comunicación por cable. El software recibe estos datos y los transfiere al modelo, donde se procesan las ecuaciones asociadas para calcular las intensidades experimentales a partir de los sensores y las intensidades teóricas a partir del voltaje de entrada de la bobina. Los resultados obtenidos son registrados en archivos para permitir el análisis posterior y la visualización del historial de mediciones.

Una vez procesados los datos, el sistema genera las representaciones gráficas que son mostradas al usuario a través de la interfaz gráfica. Estas gráficas incluyen la comparación entre las curvas teóricas y experimentales, permitiendo analizar la variación de las mediciones en función del tiempo, la distancia u otras variables relevantes.

El actuador del sistema corresponde a un servomotor, cuyo movimiento es controlado desde la interfaz gráfica mediante la selección de un ángulo dentro de un rango definido. Al ingresar un nuevo ángulo, el controlador envía la instrucción correspondiente al microcontrolador, el cual ejecuta el movimiento del servo para acercar o alejar el loop del sensor RF respecto de la bobina Tesla.

El sistema opera bajo un esquema de retroalimentación, ya que los cambios en la posición del sensor RF generados por el servomotor influyen directamente en las mediciones realizadas. Esta información es procesada nuevamente por el sistema, permitiendo observar en tiempo real el efecto del movimiento del actuador sobre las curvas obtenidas.

4.2. Arquitectura MVC

El software del sistema se estructura utilizando el patrón Modelo–Vista–Controlador (MVC) con el objetivo de mantener una separación clara entre la lógica de procesamiento, la interfaz de usuario y la coordinación del flujo de datos. Esta separación facilita la modularidad del proyecto y permite realizar mejoras o cambios en el sistema sin afectar todas las partes del software.

Modelo (Model). El modelo se encarga de gestionar y procesar los datos adquiridos durante las mediciones. En particular, almacena los voltajes de salida de los sensores (RF y fotodiodo) y el voltaje de entrada aplicado a la bobina Tesla. A partir de estos datos, ejecuta las ecuaciones necesarias para calcular las intensidades de forma experimental (usando los voltajes de los sensores) y de forma teórica (usando el voltaje de entrada de la bobina). Además, compara los resultados teóricos y experimentales, calcula el error asociado y registra la información en un archivo para su posterior análisis. Como parte del modelamiento del sistema, también considera el cálculo de la distancia entre la bobina y el loop del sensor RF en función del ángulo del servomotor.

Vista (View). La vista corresponde a la interfaz gráfica utilizada por el usuario para observar el comportamiento del sistema. Su función principal es mostrar gráficos con curvas teóricas y experimentales, visualizar valores relevantes y permitir el acceso a los registros asociados a cada experimento. La vista también entrega controles para interacción del usuario, como el ingreso del ángulo del servomotor, opciones para seleccionar distintos tipos de gráficos y un botón de reinicio para comenzar nuevas tomas de datos. La vista no realiza cálculos ni procesa datos; únicamente presenta la información entregada por el modelo y dirije las acciones del usuario hacia el controlador.

Controlador (Controller). El controlador actúa como intermediario entre la vista y el modelo, gestionando el flujo de información del sistema. Recibe los datos enviados por el microcontrolador y los transfiere al modelo para su procesamiento y registro. Asimismo, ante acciones del usuario (por ejemplo, cambiar el tipo de gráfico, solicitar historial, reiniciar registros o ingresar un ángulo de servo), el controlador coordina las actualizaciones del modelo y de la interfaz gráfica. En el caso del movimiento del servomotor, el controlador envía el comando correspondiente al microcontrolador para ejecutar el ángulo solicitado.

En conjunto, esta organización permite que el procesamiento del sistema pueda ejecutarse de manera independiente a la interfaz gráfica, y facilita la extensión del proyecto en sprints posteriores, por ejemplo, incorporando nuevos sensores, nuevas métricas o nuevos tipos de análisis sin alterar la estructura completa del software.

5. Diseño Orientado a Objetos: UML de clases

5.1. Diagrama de Clases

El diagrama de clases presenta la estructura general del sistema bajo un enfoque de Programación Orientada a Objetos, organizando el software mediante la arquitectura Modelo–Vista–Controlador (MVC). En este diagrama se representan las clases principales del software (Vista, Controlador y Modelo), junto con la abstracción de los componentes físicos (microcontrolador, sensores y servomotor) y sus relaciones. El objetivo es describir la organización y comunicación entre los módulos del sistema.

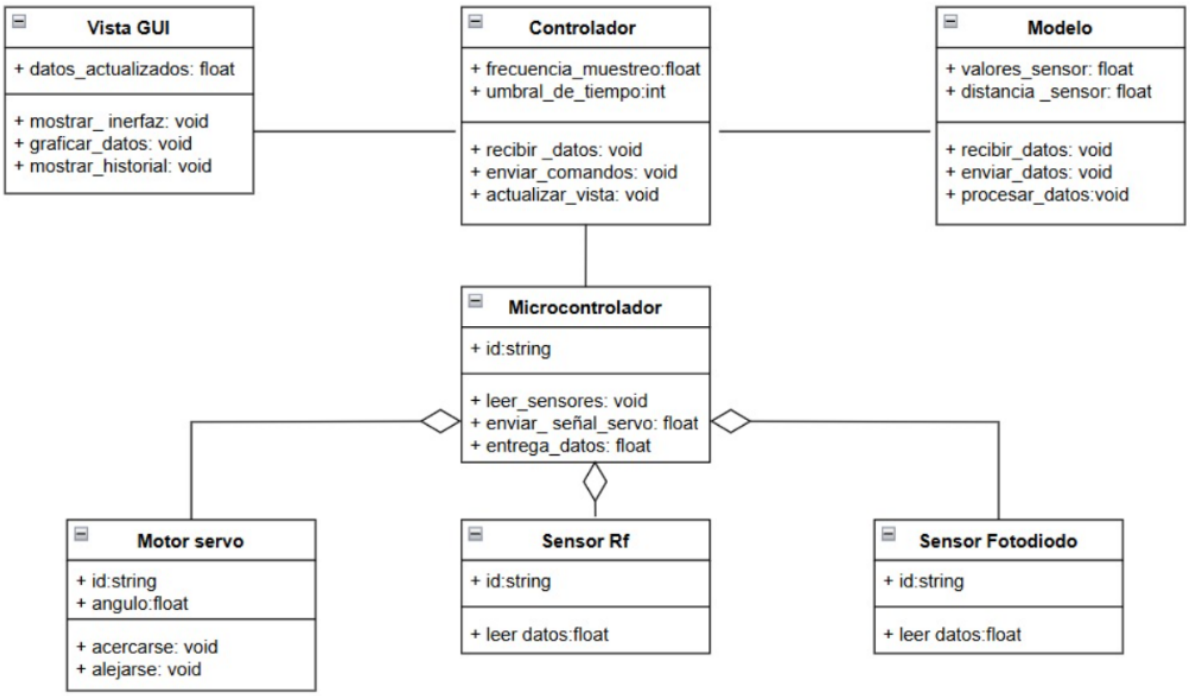

Figura 1: Diagrama UML de clases (Sprint 1).

5.2. Descripción de clases principales

ClaseControlador

La clase Vista (GUI) corresponde a la interfaz gráfica de usuario (*Graphical User Interface*) del sistema y representa el punto de interacción entre el operador y el software. Su función principal es permitir la visualización de la información procesada y la ejecución de acciones por parte del usuario.

Entradas: recibe por una parte, los datos provenientes del microcontrolador (voltajes medidos por los sensores y voltaje de alimentacion de la bobina) y, por otra las acciones solicitadas desde la vista, tales como visualizar datos, reiniciar registros o mover el servomotor.

Procesamiento / decisiones: el controlador interpreta las acciones solicitadas desde la vista y coordina el flujo de información entre el modelo y el microcontrolador. En función de estas acciones, decide cuándo solicitar datos al modelo para actualizar la vista y cuándo ejecutar comandos hacia el microcontrolador, como la rotación del servomotor. Adicionalmente, gestiona el inicio, detención y reinicio del registro de datos durante el experimento.

Salidas: envía datos procesados o consultados hacia la vista para su visualización (valores numéricos, gráficos e historial), y envía comandos hacia el microcontrolador para ejecutar acciones físicas, principalmente el control del servomotor.

Clase Modelo

La clase Modelo es la encargada del procesamiento central de la información del sistema. Su función principal es recibir los datos medidos, realizar los cálculos correspondientes y almacenar los resultados para su posterior consulta.

Entradas: el modelo recibe los voltajes de salida del sensor de radiofrecuencia y del fotodiodo, el voltaje de alimentación de entrada de la bobina Tesla, el ángulo del servomotor y las señales de control enviadas por el controlador según las acciones solicitadas por el usuario.

Procesamiento: a partir de los datos recibidos, el modelo realiza la conversión de voltaje a intensidad tanto para el sensor RF como para el fotodiodo, aplicando las ecuaciones correspondientes. Además, calcula la intensidad teórica de la bobina Tesla utilizando el voltaje de alimentación y los modelos teóricos definidos. El ángulo del servomotor se emplea para calcular la distancia entre la bobina y el loop del sensor RF. Con esta información, el modelo compara las curvas teóricas y experimentales y determina el error asociado. Durante este proceso, se registra el tiempo de medición junto con todas las variables calculadas.

Registro de datos: el modelo almacena de forma estructurada los voltajes medidos, las intensidades calculadas, el ángulo del servomotor, la distancia estimada, el tiempo de medición y el error obtenido, permitiendo construir un historial completo del experimento.

Salidas: cuando el controlador lo solicita, el modelo entrega los valores procesados, las curvas teóricas y experimentales, los errores calculados y el historial de registros para su visualización o análisis.

Responsabilidades excluidas: el modelo no interactúa directamente con el usuario ni con el hardware, ni toma decisiones relacionadas con la interfaz gráfica o el control físico del sistema. Su rol se limita al procesamiento, almacenamiento y entrega de información.

Independencia: el modelo puede operar de manera independiente de la vista, ya que el procesamiento y registro de datos se realiza aun cuando no exista una visualización activa.

Clase Vista (GUI)

La clase Vista corresponde a la interfaz gráfica del sistema y representa el punto de interacción entre el operador y el software. Su función principal es permitir la visualización de la información procesada y la ejecución de acciones por parte del usuario.

Entradas: la vista recibe, a través del controlador, los valores numéricos procesados, las curvas teóricas y experimentales, los errores calculados y el historial de registros asociados a cada experimento.

Interacción del usuario: la vista permite al operador ejecutar acciones como seleccionar el tipo de gráfico a visualizar, ingresar el ángulo del servomotor para modificar la posición del sensor RF, detener el experimento para finalizar la adquisición de datos, reiniciar el sistema mediante un botón de *reset* que elimina los registros almacenados y descargar los archivos de datos generados durante el experimento.

Procesamiento: la vista no realiza cálculos ni procesamiento de datos. Su rol se limita a mostrar la información entregada por el controlador y a capturar las acciones solicitadas por el usuario mediante botones y controles de la interfaz.

Comunicación con el sistema: todas las acciones generadas desde la vista son enviadas al controlador, quien se encarga de gestionar la comunicación con el modelo y el microcontrolador. La vista no recibe información de forma directa desde el modelo ni interactúa directamente con el hardware.

Rol dentro del sistema: la vista existe para proporcionar al operador una visualización clara del comportamiento del sistema y del registro del experimento, facilitando el análisis y la validación del funcionamiento de la bobina Tesla sin intervenir en la lógica interna del sistema.

Clase Sensor RF

La clase Sensor RF representa el sensor encargado de medir la radiofrecuencia generada por el campo electromagnético de la bobina Tesla. Su función principal es convertir la señal asociada a dicho campo en un voltaje medible por el sistema.

Entrada: el sensor recibe como entrada el campo electromagnético producido por la bobina Tesla, cuya intensidad varía segun el funcionamiento del sistema.

Salida: el sensor entrega como salida una señal de voltaje analógica, proporcional a la radiofrecuencia medida. Este voltaje es enviado directamente al microcontrolador para su adquisición.

Comunicación con el sistema: el sensor RF se conecta físicamente al microcontrolador y no interactúa de forma directa con el software. No realiza procesamiento ni toma decisiones, limitándose únicamente a la medición y conversión de la señal física a una señal eléctrica.

Clase Sensor Fotodiodo

La clase Sensor Fotodiodo representa el sensor encargado de medir la intensidad luminosa generada por la descarga eléctrica de la bobina Tesla. Su función principal es convertir la radiación luminosa incidente en una señal eléctrica utilizable por el sistema.

Entrada: el sensor recibe fotones de luz producidos por la descarga eléctrica de la bobina Tesla, cuya intensidad varía en función del funcionamiento del sistema.

Salida: el fotodiodo entrega una señal de voltaje analógica proporcional a la intensidad luminosa detectada. Esta señal es enviada directamente al microcontrolador para su adquisición.

Comunicación con el sistema: el sensor fotodiodo se conecta físicamente al microcontrolador y no interactúa de manera directa con el software. Su rol se limita a la medición de la señal luminosa y a su conversión en una señal eléctrica.

Clase Motor Servo

La clase Motor Servo representa el actuador encargado de modificar la posición del loop del sensor de radiofrecuencia dentro del sistema. Su función principal es controlar la distancia entre el sensor RF y la bobina Tesla, permitiendo acercar o alejar el sensor durante el experimento.

Entrada: el servomotor recibe una señal de control proveniente del microcontrolador, la cual indica el ángulo de rotación que debe ejecutar. Este ángulo es definido por el usuario desde la vista, gestionado por el controlador y finalmente enviado al microcontrolador.

Salida: como resultado de la señal recibida, el servomotor realiza una rotación mecánica que modifica la posición del loop del sensor RF, generando una variación en la distancia respecto a la bobina Tesla.

Comunicación con el sistema: el servomotor se comunica únicamente con el microcontrolador y no interactúa de forma directa con el software. Su rol se limita a ejecutar las órdenes de movimiento indicadas por el microcontrolador.

6. Planificación Scrum

6.1. Metodología Scrum aplicada al proyecto

Para el desarrollo del proyecto se adopta la metodología ágil Scrum, con el objetivo de mantener un mejor control del proceso, permitiendo flexibilidad y adaptación a medida que el proyecto avanza. Scrum facilita la división del desarrollo en fases incrementales denominadas sprints, lo que permite evaluar el progreso de forma periódica y realizar ajustes cuando sea necesario.

El proyecto se organiza en tres sprints, de acuerdo con lo establecido en la asignatura, con una duración aproximada de dos a tres semanas cada uno. A través de esta metodología se busca lograr una adecuada organización del trabajo, mejorar la coordinación del equipo y asegurar avances incrementales en la construcción del sistema.

6.2. Roles Scrum

El equipo de trabajo está conformado por cuatro integrantes, quienes asumen los roles definidos por la metodología Scrum, adaptados al contexto del proyecto académico.

El rol de *Product Owner* es asumido por Matías, quien representa al cliente y es responsable de priorizar las tareas del proyecto y definir los objetivos a alcanzar en cada sprint.

El rol de *Scrum Master* es desempeñado por Alejandro, encargado de facilitar el desarrollo del proyecto, promover buenas prácticas de trabajo y asegurar el correcto uso de la metodología Scrum dentro del equipo.

El equipo de desarrollo está conformado por Pablo y Daniel. Pablo es responsable del diseño del software, enfocándose en la modularidad del sistema y la correcta aplicación de la arquitectura definida. Daniel se encarga del desarrollo del hardware y de su integración con el software, considerando la lógica de control del sistema.

6.3. Product Backlog

El Product Backlog del proyecto reúne todas las tareas identificadas para el desarrollo completo del sistema, desde la definición de requisitos y el diseño conceptual hasta la implementación, integración y validación del sistema final. Cada tarea fue estimada utilizando Story Points bajo la escala de Fibonacci, considerando su complejidad, nivel de conocimiento requerido y criticidad dentro del proyecto.

Tarea	**Story Points**	**Criticidad**	**Justificación**
Elaboración del documento SRS	8	Alta	Define los requisitos y alcances del sistema, constituyendo la base del desarrollo posterior.
Diseño de la arquitectura MVC	8	Alta	Establece la estructura del software y la separación de responsabilidades del sistema.
Elaboración del diagrama UML	5	Media	Permite representar las clases del sistema y sus relaciones antes de la implementación.
Definición del Product Backlog y Sprint Backlog	5	Media	Organiza y prioriza las tareas del proyecto bajo la metodología Scrum.
Definición de la estructura base del repositorio GitHub	2	Baja	Facilita la organización del código y la documentación, mejorando el trabajo en equipo.
Definición conceptual del hardware y variables a medir	13	Alta	Identifica los componentes del sistema, su funcionamiento y las variables físicas a medir.
Diseño y especificación técnica del hardware	13	Alta	Requiere conocimiento técnico específico y decisiones de diseño con impacto directo en todo el sistema.
Pruebas de funcionamiento y alimentación del hardware	8	Alta	Involucra validación eléctrica, seguridad y riesgo físico durante la operación del sistema.
Implementación y validación del modelo matemático	8	Alta	Incluye el uso de fórmulas, comparación teórica–experimental y cálculo del error asociado.
Implementación y validación del sistema de registro de datos	5	Media	Verifica el correcto almacenamiento de los datos calculados, dependiendo de la lógica del modelo.
Desarrollo y validación de la interfaz gráfica (Vista)	5	Media	Permite comprobar la interacción del usuario con gráficos, botones y control del sistema.
Implementación del controlador e integración del sistema	13	Alta	Conecta el hardware, el modelo y la vista, coordinando el funcionamiento completo del sistema.
Análisis de tiempos de respuesta y retardo del sistema	3	Baja	Evalúa el tiempo de procesamiento y actualización de datos una vez validada la arquitectura.

Cuadro 1: Product Backlog del proyecto con estimación en Story Points.

6.4. Sprint Backlog (Sprint 1)

El Sprint Backlog del Sprint 1 considera las tareas fundamentales orientadas a la definición, diseño y planificación del proyecto. Estas actividades establecen la base técnica y organizacional del sistema, sin abordar aún la implementación completa ni la integración física del hardware, las cuales se desarrollarán en sprints posteriores.

Ítem / Tarea seleccionada	**Story Points**
Elaboración del documento SRS	8
Diseño de la arquitectura MVC	8
Elaboración del diagrama UML	5
Definición del Product Backlog y Sprint Backlog	5
Definición de la estructura base del repositorio GitHub	2
Definición conceptual del hardware y variables a medir	13

Cuadro 2: Sprint Backlog correspondiente al Sprint 1.

6.5. User Stories

Las siguientes historias de usuario representan las principales necesidades del usuario del sistema, identificado como el operador. Este rol engloba a cualquier persona que utilice el sistema con fines académicos o de investigación, tales como estudiantes, docentes o investigadores.

- Como operador, quiero medir las señales de radiofrecuencia y del fotodiodo para verificar que los sensores funcionan dentro de los rangos definidos en sus especificaciones técnicas.
- Como operador, quiero visualizar valores numéricos en pantalla para comprobar en tiempo real las mediciones entregadas por los sensores.
- Como operador, quiero visualizar gráficos en tiempo real para analizar la variación de las mediciones ante cambios en el sistema.
- Como operador, quiero registrar automáticamente los datos obtenidos durante un experimento para poder analizarlos posteriormente.
- Como operador, quiero descargar los archivos de registro al finalizar un experimento para respaldar y reutilizar los datos obtenidos.
- Como operador, quiero reiniciar el experimento mediante una opción de *reset* para borrar los registros anteriores antes de realizar una nueva medición.

- Como operador, quiero controlar el ángulo del servomotor desde la interfaz gráfica para acercar o alejar el sensor RF respecto de la bobina Tesla.
- Como operador, quiero comparar en un mismo gráfico la curva teórica y la curva experimental para evaluar el comportamiento del sistema.
- Como operador, quiero visualizar el error entre los valores teóricos y experimentales para validar el desempeño del sistema de monitoreo.
- Como operador, quiero asegurar que la alimentación de la bobina Tesla se mantenga dentro de los rangos de funcionamiento definidos para realizar los cálculos correspondientes.

7. Avance funcional alcanzado (20 %)

7.1. Resultados del Sprint 1

Durante el Sprint 1 se logró definir la base conceptual y estructural del proyecto. En esta etapa quedaron completamente desarrollados el documento de Especificación de Requisitos de Software (SRS), la arquitectura del sistema, la metodología de desarrollo utilizada, el diagrama UML de clases y la planificación mediante Product Backlog y Sprint Backlog. Asimismo, se definió el funcionamiento general del sistema, la lógica de control y la descripción detallada de las clases principales.

A nivel de diseño, se cerraron decisiones fundamentales como la adopción de la arquitectura Modelo–Vista–Controlador (MVC), el flujo de datos entre hardware y software, y la separación clara de responsabilidades entre los distintos módulos del sistema. Estas definiciones permiten asegurar un desarrollo modular y escalable.

Las clases del sistema, junto con sus roles y relaciones, quedaron conceptual y estructuralmente definidas, aun cuando su implementación física y de software no se ha realizado en esta etapa. En términos globales, el trabajo desarrollado en este sprint representa aproximadamente entre un 20 % y un 30 % del proyecto total, correspondiente a la etapa de diseño y planificación.

7.2. Evidencias del avance del proyecto

Como respaldo del avance alcanzado durante el Sprint 1, se incluyen diversas evidencias gráficas que complementan la documentación desarrollada. Estas corresponden principalmente a la definición de la estructura del repositorio del proyecto y a prototipos de interfaz gráfica elaborados a nivel conceptual.
Las evidencias presentadas permiten visualizar de forma concreta las decisiones de diseño adoptadas, la organización modular del software y una propuesta inicial de interacción con el sistema, alineadas con la arquitectura MVC definida, sin corresponder aún a una implementación funcional completa.

```
.gitignore
README.md
requirements.txt

docs
└── sprint_1
presentations
└── sprint_1
src
└── tesla_monitor
    ├── main.py
    ├── __init__.py
    ├── config
    │   ├── settings.py
    │   └── __init__.py
    ├── controller
    │   ├── controller.py
    │   └── __init__.py
    ├── model
    │   ├── almacenamiento.py
    │   ├── ecuaciones.py
    │   ├── microcontrolador.py
    │   ├── modelo.py
    │   ├── sensores.py
    │   └── __init__.py
    └── view
        ├── vista_streamlit.py
        └── __init__.py

tests
```

Figura 2: Estructura base del repositorio GitHub del proyecto, organizada de forma modular según la arquitectura MVC, incluyendo carpetas para configuración, controlador, modelo, vista y documentación del Sprint 1.

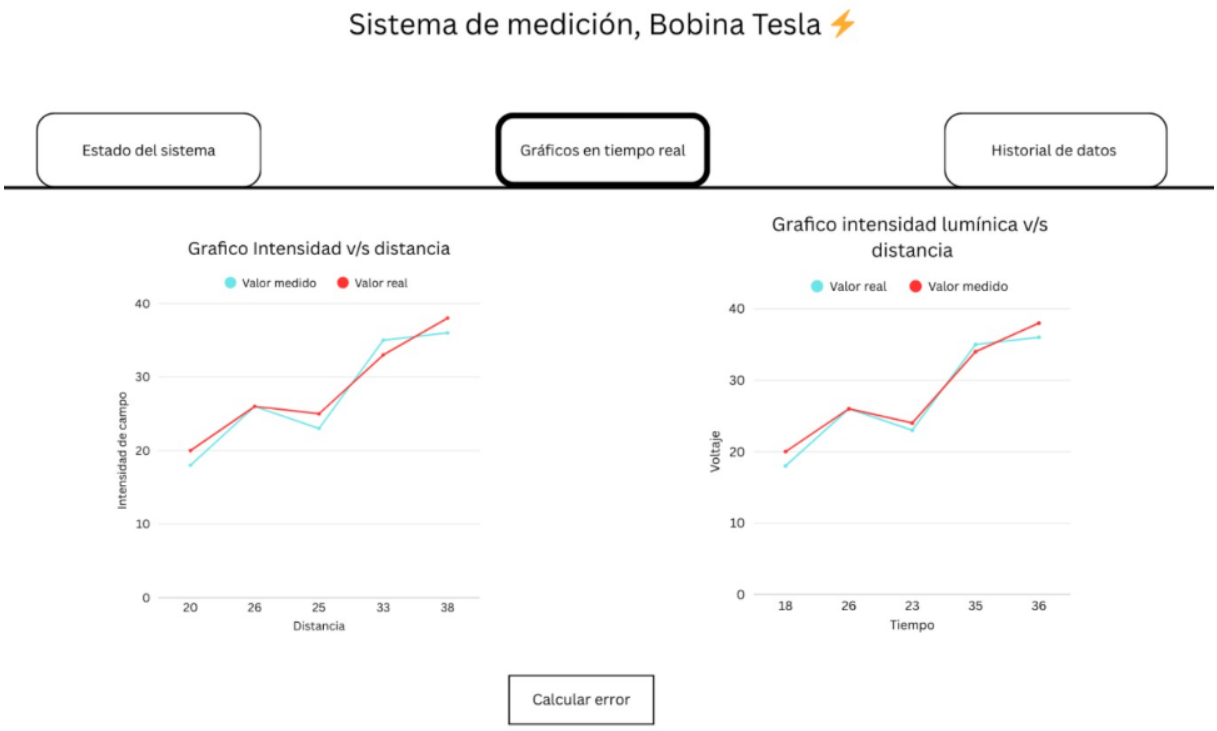

Figura 3: Mockup conceptual de la interfaz gráfica del sistema, correspondiente a la visualización de gráficos en tiempo real de las variables medidas durante el experimento.

Figura 4: Mockup conceptual de la sección de historial de datos, donde se propone la visualización tabular de los registros obtenidos y la opción de exportación para análisis posterior.

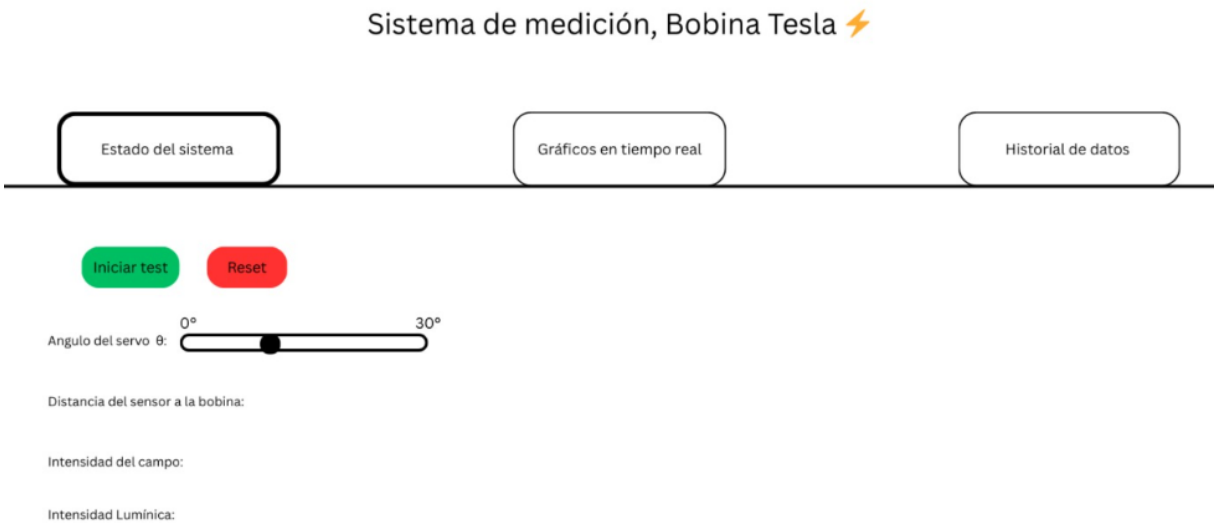

Figura 5: Mockup conceptual de la vista de estado del sistema, que ilustra los controles propuestos para iniciar el experimento, reiniciar el sistema y ajustar el ángulo del servomotor.

7.3. Proyección para el siguiente Sprint

De manera intencional, en este Sprint no se abordó la construcción física del hardware, las pruebas experimentales ni la integración completa entre hardware y software. Estas

actividades forman parte del Sprint 2, donde se priorizará el montaje del sistema, la validación del funcionamiento del hardware y la conexión con el software desarrollado por módulos.

El trabajo realizado en este Sprint permite avanzar hacia las siguientes etapas sin necesidad de rehacer decisiones, ya que las bases metodológicas, estructurales y funcionales del sistema han sido previamente definidas. Esto posibilita que los siguientes sprints se enfoquen principalmente en la implementación, pruebas y validación del sistema completo.

8. Conclusiones y trabajo futuro

El objetivo principal del Sprint 1 se cumplió satisfactoriamente, ya que se completaron todas las tareas definidas en el Sprint Backlog, estableciendo las bases conceptuales, metodológicas y estructurales del proyecto. En esta etapa se logró definir de forma clara los requisitos del sistema, la arquitectura de software, el diseño orientado a objetos y la planificación del desarrollo bajo la metodología Scrum.

El enfoque de partir por requisitos, arquitectura y diseño antes de la construcción del hardware resultó fundamental, ya que permitió modular el sistema, definir responsabilidades claras y evitar problemas de diseño, mantenimiento y escalabilidad. La adopción de la arquitectura MVC y la programación orientada a objetos facilitó la separación entre hardware y software, además de promover un desarrollo flexible y ordenado.

La metodología Scrum aportó una estructura clara al proyecto, permitiendo organizar el trabajo en sprints, dividir tareas y planificar de manera progresiva las distintas etapas del desarrollo. Uno de los principales desafíos de este Sprint fue definir correctamente la lógica del sistema y las relaciones entre las clases, lo que permitió comprender en profundidad el funcionamiento global del proyecto.

Como trabajo futuro, el Sprint 2 se enfocará en la construcción y validación del hardware, su integración con el software y la implementación de los módulos definidos, para finalmente avanzar hacia la etapa de pruebas y validación del sistema completo de monitoreo y análisis del rendimiento de la bobina Tesla.

9. Referencias

Referencias

[1] UNIVERSIDAD DE SANTIAGO DE CHILE. *Ingeniería de Software Interdisciplinaria – Aula Virtual*. Disponible en: https://uvirtual.usach.cl/moodle/course/view.php?id=29656

[2] ENCYCLOPAEDIA BRITANNICA. *Tesla coil*. Disponible en: https://www.britannica.com/technology/Tesla-coil

[3] TEXAS INSTRUMENTS. *RF Measurement Basics*. Disponible en: https://www.ti.com/lit/ml/slap127/slap127.pdf

[4] HAMAMATSU PHOTONICS. *Photodiode Technical Information*. Disponible en: https://www.hamamatsu.com/content/dam/hamamatsu-photonics/sites/documents/99_SALES_LIBRARY/ssd/si_pd_kspd9001e.pdf

[5] SCHWABER, K.; SUTHERLAND, J. *The Scrum Guide*. Disponible en: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf

[6] IEEE COMPUTER SOCIETY. *IEEE Std 830-1998: Recommended Practice for Software Requirements Specifications*. Disponible en: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf