Inteligencia Artificial en sistemas embebidos

Autor:
Marcos Brito Devoto

Director:
@todo Nombre del Director (@todo pertenencia)

Codirector:
@todo Nombre del CoDirector (@todo)

Esta planificación fue realizada en el curso de Gestión de proyectos entre el 17 de octubre de 2023 y el 22 de abril de 2024.

Índice

1. Descripción técnica-conceptual del proyecto a realizar **5**

2. Identificación y análisis de los interesados **6**

3. Propósito del proyecto **7**

4. Alcance del proyecto **7**

5. Supuestos del proyecto **8**

6. Requerimientos **8**

7. Historias de usuarios (*Product backlog*) **9**

8. Entregables principales del proyecto **10**

9. Desglose del trabajo en tareas **11**

10. Diagrama de Activity On Node **12**

11. Diagrama de Gantt **12**

12. Presupuesto detallado del proyecto **15**

13. Gestión de riesgos **15**

14. Gestión de la calidad **16**

15. Procesos de cierre **17**

Registros de cambios

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	17 de octubre de 2023
1	Se completa hasta el punto 4 inclusive	31/10/2023
2	Se completa hasta el punto 7 inclusive	07/11/2023

Acta de constitución del proyecto

Buenos Aires, 17 de octubre de 2023

Por medio de la presente se acuerda con el Ing. Marcos Brito Devoto que su Trabajo Final de la Carrera de Especialización en Inteligencia Artificial se titulará "Inteligencia Artificial en sistemas embebidos", consistirá esencialmente en un análisis profundo del estado del arte de sistemas de inteligencia artificial en sistemas embebidos, y tendrá un presupuesto preliminar estimado de 600 h de trabajo y @todo $XXX, con fecha de inicio 17 de octubre de 2023 y fecha de presentación pública @todo 15 de mayo de 2022.

Se adjunta a esta acta la planificación inicial.

Dr. Ing. Ariel Lutenberg
Director posgrado FIUBA

@todo Quién de la empresa lo va aprobar
Very

@todo Nombre del Director
Director del Trabajo Final

1. Descripción técnica-conceptual del proyecto a realizar

El proyecto final surge a partir de una necesidad de la empresa Very de incursionar en el campo de la inteligencia artificial en sistemas embebidos. Cómo consultora, la compañia cuenta con vasta experiencia en desarrollo de modelos de inteligencia artificial. Estos modelos suelen tomar datos de múltiples sensores (o nodos) desplegados en el campo que descargan datos a una computadora central o a la nube, donde se ejecutan los modelos de inteligencia artificial y se proveen métricas de interés al cliente. Una tercera alternativa consiste en ejecutar estos algoritmos directamente en los nodos, también llamado "*on-the-edge*", ahorrando costos de transmisión y uso de batería, entre otras importantes ventajas. Actualmente, la empresa no cuenta con experiencia en este tipo de tecnología, por lo que se encuentra limitada para utilizarla en proyectos de clientes donde éste enfoque aportaría gran valor.

@todo discutir de confidencialidad con Very: ¿Existen o aplican condiciones especiales al proyecto, financiamiento de algún programa público o privado, acuerdos de confidencialidad, acuerdos sobre la propiedad intelectual de los entregables u otros?

En la Figura 1 se muestra la diferencia entre dos arquitecturas muy utilizadas en aplicaciones IoT. En el diagrama de la izquierda, cada uno de los nodos es encargado de tomar mediciones y enviar los datos a la nube para ser procesados. Esto simplifica el diseño de los nodos significativamente, cuya programación es sencilla y actúan unicamente como recolectores de datos. La complejidad se traslada a la nube, donde se debe mantener una base de datos y poder de computo suficiente para almacenar y procesar toda la información proveniente de los sensores. Además, se debe garantizar una conexión robusta entre los nodos y el servidor. Por otro lado, en el diagrama de la derecha se puede ver una arquitectura descentralizada, donde cada nodo realiza las mediciones y procesa los datos localmente. Esto se conoce como "*edge-computing*", ya que se realiza el cómputo de la salida "en el borde". De esta forma, el nodo solo debe transmitir las métricas de salida de los algoritmos, disminuyendo significativamente la cantidad de información que se comunica.

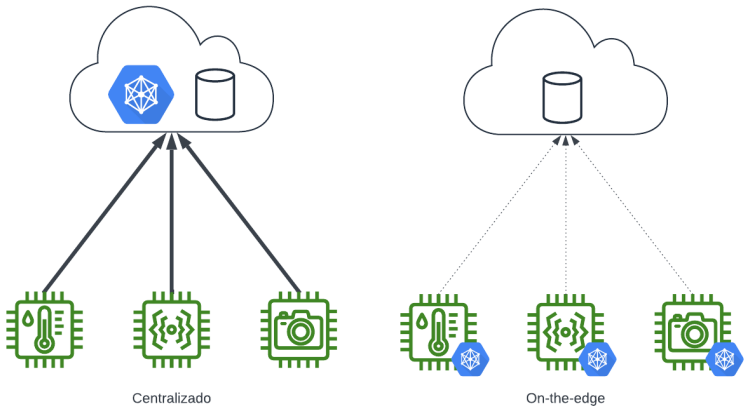

Figura 1. Diagrama de *arquitecturas IoT*.

Entre las ventajas del enfoque descentralizado se encuentran:

- **Reducción de la latencia**: el procesamiento de datos en la nube puede ser lento, especialmente en entornos con una conexión de red de baja velocidad, algo muy utilizado en IoT ya que aumenta la duración de la batería que alimenta al nodo. Esto imposibilita su aplicación para aplicaciones en tiempo real, donde se debe obtener la salida del algoritmos en cuestión de menos de 1 segundo. El *edge-computing* permite procesar los datos localmente, lo que reduce la latencia y mejora la respuesta del sistema.
- **Mejora de la seguridad**: la transferencia de datos a la nube para procesamiento puede exponer información confidencial a riesgos de seguridad. El *edge-computing* reduce el riesgo de acceso no autorizado a los datos, ya que se procesan localmente y solo se transmiten los resultados.
- **Aumento de la eficiencia**: uno de los procesos que más energía consume IoT es la transmisión de datos, ya que generalmente ocurre a través de redes inalámbricas (WiFi, Bluetooth, LoRaWAN, etc). El *edge-computing* reduce la cantidad de datos que es necesario transmitir, lo que genera ahorros de energía considerables. Esto es especialmente importante en dispositivos que deben funcionar durante años con baterías.
- **Reducción de costos**: los servicios en la nube necesarios para ejecutar algoritmos de inteligencia artificial conllevan un costo de mantenimiento mensual que el cliente debe hacer frente cómo un gasto fijo operativo en su negocio. En ciertas aplicaciones, este gasto fijo se puede eliminar casi por completo utilizando un enfoque descentralizado.

Las desventajas de utilizar una arquitectura descentralizada son:

- **Mayor costo de implementación**: la implementación de este tipo de sistemas puede requerir inversiones significativas en *hardware* y *software* especializados.
- **Complejidad de mantenimiento**: el mantenimiento y actualización de estos sistemas distribuidos puede ser más complejo y costoso debido a la necesidad de gestionar múltiples dispositivos en simultaneo.
- **Limitaciones de recursos y escalabilidad**: los dispositivos de *edge-computing*, como los microcontroladores, suelen tener recursos limitados en términos de potencia de procesamiento, capacidad de almacenamiento y memoria. Esto puede dificultar la ejecución de aplicaciones de inteligencia artificial complejas y de alto rendimiento. Por esta razón, es crucial considerar desde las etapas iniciales del proyecto si esto puede llegar a limitar el crecimiento y la evolución futura del negocio.

2. Identificación y análisis de los interesados

Rol	Nombre y Apellido	Organización	Puesto
Cliente	@todo Quién de la empresa lo va aprobar	Very	
Responsable	Marcos Brito Devoto	Very	Alumno
Colaboradores	@todo Preguntar si Jose R?		
Orientador	@todo Nombre del Director	@todo pertenencia	Director Trabajo final
Usuario final	@todo Quién de la empresa lo va aprobar	Very	

3. Propósito del proyecto

El propósito principal de este proyecto es sentar las bases para el desarrollo de aplicaciones de inteligencia artificial en sistemas embebidos, proporcionando a la empresa un repositorio que facilite las primeras etapas de nuevos proyectos de IA "*on-the-edge*". Esto permitirá comprender las capacidades y limitaciones de la IA en sistemas de recursos limitados, como los microprocesadores, y fomentará la eficiencia y viabilidad de futuros desarrollos en este campo.

4. Alcance del proyecto

El alcance del proyecto se centra en el desarrollo de un repositorio *template* para aplicaciones de inteligencia artificial en sistemas embebidos. El objetivo principal es proporcionar una base de herramientas que faciliten el desarrollo de aplicaciones de IA en plataformas de recursos limitados, como microcontroladores. También se incluirá un análisis de las posibilidades de esta tecnología para que Very pueda tomar evaluar rápidamente si es viable su aplicación con un determinado cliente.

El proyecto incluirá los siguientes elementos:

1. Investigación del estado del arte en inteligencia artificial aplicada a sistemas embebidos.
2. Análisis y evaluación de librerías y *frameworks* disponibles para el desarrollo de aplicaciones de IA en sistemas embebidos.
3. Selección de un microcontrolador adecuado para llevar a cabo análisis y pruebas de las aplicaciones de IA.
4. Creación de un repositorio en Github donde se desarrollará el repositorio *template*.
5. Implementación de algoritmos de inteligencia artificial en el repositorio *template*, incluyendo *machine learning* clásico, *deep learning* y, si es posible, visión por computadora.
6. Análisis de métricas para evaluar el rendimiento de los algoritmos implementados en términos de velocidad de ejecución, consumo de memoria y eficiencia energética. También se evaluará la precisión de los algoritmos comparándolo con el mismo modelo ejecutado en una computadora de uso general utilizando Python.
7. Conclusiones basadas en los resultados obtenidos en el análisis de métricas.

El alcance del proyecto no incluye:

1. El desarrollo completo de una aplicación de IA específica, sino que se centrará en proporcionar una base sólida y estructurada para futuros proyectos en este campo.
2. No habrá recolección de datos para generar un *dataset*. Para entrenar y evaluar los modelos, se utilizarán *datasets* provistos por Very o de uso público.
3. La implementación de soluciones de *edge computing* bajo el contexto específico de IoT, ni tampoco un análisis profundo respecto a la escalabilidad y flexibilidad de éstas arquitecturas.
4. Desarrollar nuevas herramientas o tecnologías de inteligencia artificial en sistemas embebidos. El proyecto se centra en conocer y comprender el estado del arte actual en este campo, no en innovación dentro del mismo.
5. Los modelos de *machine learning* no se entrenarán dentro del nodo.

5. Supuestos del proyecto

Para el desarrollo del presente proyecto se supone que:

- Los microcontroladores disponibles comercialmente son capaces de ejecutar los algoritmos de inteligencia artificial implementados en el repositorio *template*.
- Existen librerías y *frameworks* disponibles que sean adecuados para el desarrollo de aplicaciones de IA en sistemas embebidos.
- Se tiene una disponibilidad horaria de al menos 600 horas laborales del responsable para realizar el trabajo.
- Very proporcionará el tiempo y recursos necesarios para poder progresar en la implementación de los algoritmos.
- Los conocimientos adquiridos en el posgrado son suficientes para aplicar las técnicas de IA deseadas.

6. Requerimientos

Aclaración: en este proyecto se implementarán múltiples modelos de *machine learning* para evaluar su performance. Por lo tanto, los requerimientos funcionales son genéricos y aplican a todos los modelos a implementar siempre que sea posible. A partir de los requerimientos planteados, se podrá analizar las limitaciones de cada algoritmo y compararlos.

1. Requerimientos funcionales de los modelos de *machine learning*:
   1.1. El código del modelo debe estar encapsulado dentro de un módulo y debe contar con una API para acceder a las principales funciones del mismo.

1.2. El modelo debe tener la capacidad de actualizar sus pesos sin la necesidad de una actualización completa de *firmware*, siempre y cuando no se modifique la arquitectura del algoritmo.

1.3. El módulo de *machine learning* contará con metadata que permitirá identificar el tipo de algoritmo, versión del modelo y versión de los pesos.

1.4. El módulo de *machine learning* debe permitir la extracción de muestras de validación que incluirán: datos de entrada, datos y métricas de salida y metadata del modelo. Esto facilitará la validación de los algoritmos cuando el sistema se encuentre en producción.

1.5. El módulo debe ser capaz de medir el tiempo de ejecución del algoritmo y generar métricas estadísticas del uso del CPU y RAM.

2. Requerimientos de documentación

2.1. La documentación debe mantenerse dentro del mismo repositorio de Github.

2.2. La documentación debe indicar los pasos para instalar el ambiente de desarrollo, incluyendo la instalación de librerías y módulos necesarios.

2.3. La documentación debe indicar los pasos para ejecutar cada una de las pruebas que se llevaron a cabo durante el proyecto.

2.4. La documentación debe registrar la investigación realizada en el proyecto, identificando claramente las limitaciones de las tecnologías utilizadas.

2.5. La documentación debe indicar los requerimientos mínimos recomendables de *hardware* para ejecutar este tipo de aplicaciones.

3. Requerimientos funcionales de la interfaz de comunicación

3.1. El sistema debe contar con un módulo de comunicación que permitirá enviar y recibir datos con una PC conectada por USB.

3.2. El protocolo de comunicación debe validar la integridad de los datos.

3.3. La PC debe ser capaz de comunicarse con el nodo para:

1) Obtener información del nodo: versión del código, metadata del modelo de *machine learning*
2) Actualizar los pesos del algoritmo de *machine learning*

4. Requerimiento de testing

4.1. Los modelos de *machine learning* ejecutados en el nodo serán comparados contra las métricas obtenido de ejecutar los mismos modelos en una PC, utilizando Python.

4.2. La ejecución de las pruebas debe ser automática mediante un *script*.

7. Historias de usuarios (*Product backlog*)

Roles:

- Evaluador: es quien va a evaluar si utilizar algoritmos de *machine learning* en los nodos es una alternativa viable para el proyecto. Esta persona cuenta con conocimientos técnicos, aunque no necesariamente sea especialista en desarrollos de inteligencia artificial o *firmware*.

- Desarrollador: es la persona que, una vez evaluada la viabilidad de la tecnología para un proyecto, utilizará el repositorio como base de desarrollo para un modelo de inteligencia artificial específico. Esta persona cuenta con conocimientos técnicos en programación, inteligencia artificial y sistemas embebidos.

Story points:

Puntaje #	**Descripción:**
1	Estos son detalles menores o implementaciones simples que se estiman se pueden resolver en horas o en un solo día.
3	Se estima que la ejecución de este tipo de tarea puede tomar hasta un tercio del sprint.
5	Se espera que trabajar en este tipo de historia utilice aproximadamente dos tercios de la capacidad del sprint.
8	Estas son las historias más complejas o difíciles de implementar y se estima que pueden tomar todo el sprint.

Historias de usuario:

- Como evaluador, quiero tener fácil acceso al tiempo de ejecución de los algoritmos. Esto me permitirá evaluar si el consumo es demasiado elevado para, por ejemplo, un sistema que funciona con baterías (Puntos: 3).
- Como evaluador, quiero ver métricas de la ejecución de los algoritmos que sean independientes del microprocesador utilizado, para así poder extrapolar las conclusiones a otras plataformas (Puntos: 8).
- Como evaluador, quiero ver fácilmente qué tipos de algoritmos de inteligencia artificial se pueden ejecutar en sistemas embebidos de recursos limitados (Puntos: 1).
- Como evaluador, quiero conocer las limitaciones de este tipo de tecnologías, para descartar rápidamente en el caso de que no sea compatible con el proyecto que se quiere desarrollar (Puntos: 5).
- Como desarrollador, quiero tener instrucciones claras para configurar el entorno de desarrollo y así comenzar el desarrollo llo más pronto posible (Puntos: 3).
- Como desarrollador, quiero entender las limitaciones de los algoritmos de *machine learning* que se pueden ejecutar en estos sistemas, para desarrollar un modelo acorde (Puntos: 3).
- Como desarrollador, quiero poder actualizar los pesos del modelo fácilmente para poder iterar de forma eficiente (Puntos: 8).
- Como desarrollador, quiero poder tener acceso a las variables internas del modelo siendo ejecutado en el microprocesador, de forma tal de poder validar que el algoritmo se está ejecutando según lo esperado (Puntos: 5).

8. Entregables principales del proyecto

Los entregables del proyecto son:

- Módulos de *machine learning* con distintos modelos implementados.
- Ejemplos de uso de los algoritmos.
- Análisis del estado del arte y performance de los modelos.
- Documentación de instalación del entorno de desarrollo.

9. Desglose del trabajo en tareas

1. Investigación y planificación
   1.1. Investigación del estado del arte (32 hs)
   1.2. Definición de modelos de *machine learning* a implementar (24 hs)
   1.3. Selección de los datasets a utilizar para las pruebas de cada modelo (24 hs)
   1.4. Selección de la plataforma (microcontrolador) donde se llevarán a cabo las pruebas (24 hs)
   1.5. Diseñar arquitectura de firmware del nodo (16 hs)
2. Desarrollo de estructura base
   2.1. Configurar entorno de desarrollo (24 hs)
   2.2. Implementar arquitectura de firmware (40 hs)
   2.3. Implementar protocolo de comunicación (20 hs)
   2.4. Desarrollar script de interacción nodo-PC (24 hs)
   2.5. Implementar flujo del programa: adquisición de datos -¿procesamiento -¿predicción (40 hs)
3. Desarrollo de algoritmos de inteligencia artificial
   3.1. Diseñar módulo de IA y APIs (40 hs)
   3.2. Entrenar modelos en Python (40 hs)
   3.3. Desplegar modelos de IA en nodo (40 hs)
   3.4. Generar muestras de validación y envíar a la PC (24 hs)
   3.5. Testing del funcionamiento de los algoritmos (20 hs)
   3.6. Agregar mecanismos de evaluación de performance dentro del código (20 hs)
   3.7. Adaptar código para permitir la actualización de pesos del modelo (32 hs)
4. Documentación
   4.1. Documentar análisis realizado (20 hs)
   4.2. Documentar conclusiones de los modelos implementados y aspectos a tomar en cuenta respecto a la plataforma utilizada. (30 hs)
   4.3. Redacción del informe de avance (16 hs).
   4.4. Redacción de la memoria del proyecto (40 hs).
   4.5. Preparación de la presentación pública (10 hs).

Cantidad total de horas: (600 h)

10. Diagrama de Activity On Node

Armar el AoN a partir del WBS definido en la etapa anterior.

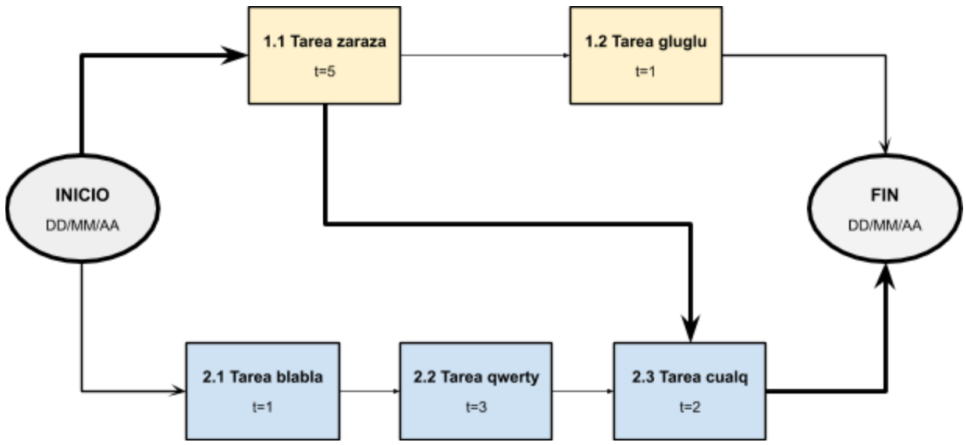

Figura 2. Diagrama de *Activity on Node*.

Indicar claramente en qué unidades están expresados los tiempos. De ser necesario indicar los caminos semicríticos y analizar sus tiempos mediante un cuadro. Es recomendable usar colores y un cuadro indicativo describiendo qué representa cada color, como se muestra en el siguiente ejemplo:

11. Diagrama de Gantt

Existen muchos programas y recursos *online* para hacer diagramas de Gantt, entre los cuales destacamos:

- Planner
- GanttProject
- Trello + *plugins*. En el siguiente link hay un tutorial oficial:
  `https://blog.trello.com/es/diagrama-de-gantt-de-un-proyecto`
- Creately, herramienta online colaborativa.
  `https://creately.com/diagram/example/ieb3p3ml/LaTeX`
- Se puede hacer en latex con el paquete *pgfgantt*
  `http://ctan.dcc.uchile.cl/graphics/pgf/contrib/pgfgantt/pgfgantt.pdf`

Pegar acá una captura de pantalla del diagrama de Gantt, cuidando que la letra sea suficientemente grande como para ser legible. Si el diagrama queda demasiado ancho, se puede pegar primero la "tabla" del Gantt y luego pegar la parte del diagrama de barras del diagrama de Gantt.

Configurar el software para que en la parte de la tabla muestre los códigos del EDT (WBS).
Configurar el software para que al lado de cada barra muestre el nombre de cada tarea.
Revisar que la fecha de finalización coincida con lo indicado en el Acta Constitutiva.

En la figura 3, se muestra un ejemplo de diagrama de Gantt realizado con el paquete de *pgfgantt*.
En la plantilla pueden ver el código que lo genera y usarlo de base para construir el propio.

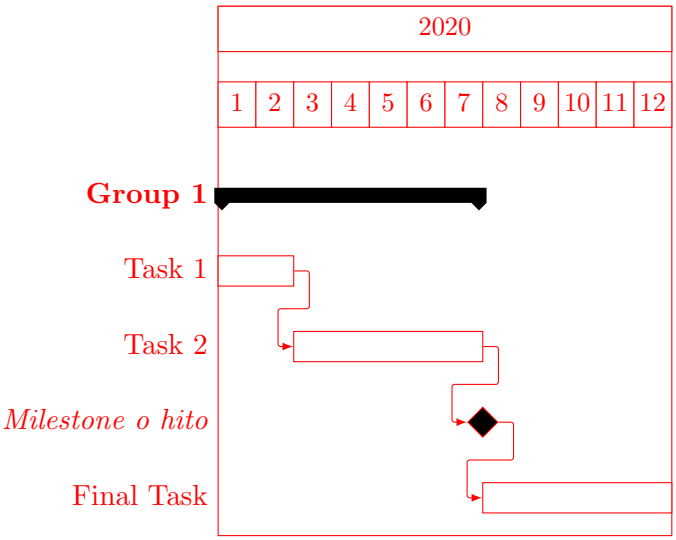

Figura 3. Diagrama de Gantt de ejemplo

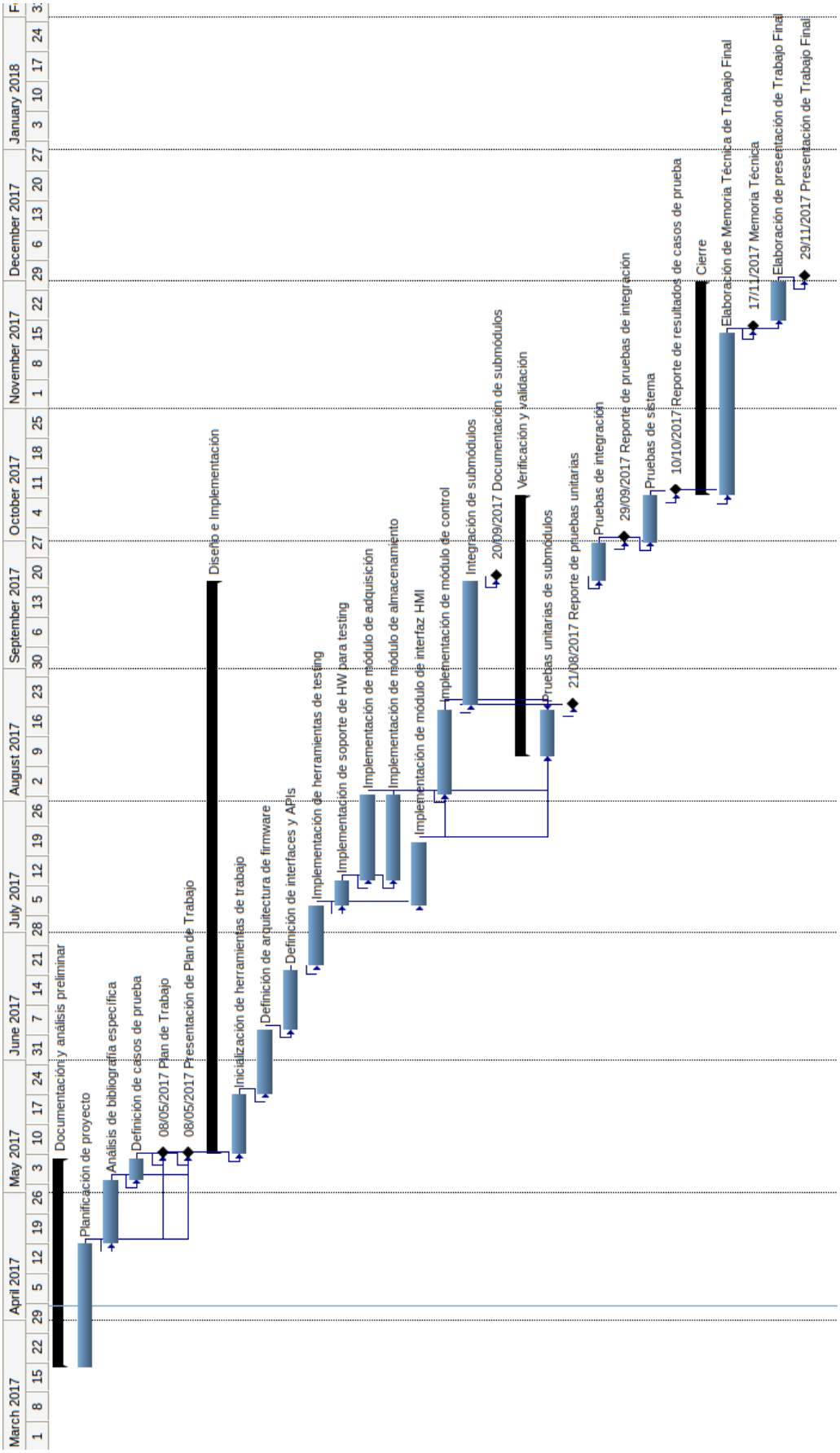

Figura 4. Ejemplo de diagrama de Gantt rotado

12. Presupuesto detallado del proyecto

Si el proyecto es complejo entonces separarlo en partes:

- Un total global, indicando el subtotal acumulado por cada una de las áreas.
- El desglose detallado del subtotal de cada una de las áreas.

IMPORTANTE: No olvidarse de considerar los COSTOS INDIRECTOS.

COSTOS DIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
SUBTOTAL			
COSTOS INDIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
SUBTOTAL			
TOTAL			

13. Gestión de riesgos

a) Identificación de los riesgos (al menos cinco) y estimación de sus consecuencias:

Riesgo 1: detallar el riesgo (riesgo es algo que si ocurre altera los planes previstos de forma negativa)

- Severidad (S): mientras más severo, más alto es el número (usar números del 1 al 10).
  Justificar el motivo por el cual se asigna determinado número de severidad (S).
- Probabilidad de ocurrencia (O): mientras más probable, más alto es el número (usar del 1 al 10).
  Justificar el motivo por el cual se asigna determinado número de (O).

Riesgo 2:

- Severidad (S):
- Ocurrencia (O):

Riesgo 3:

- Severidad (S):

- Ocurrencia (O):

b) Tabla de gestión de riesgos: (El RPN se calcula como RPN=SxO)

Riesgo	S	O	RPN	S*	O*	RPN*

Criterio adoptado: Se tomarán medidas de mitigación en los riesgos cuyos números de RPN sean mayores a...

Nota: los valores marcados con (*) en la tabla corresponden luego de haber aplicado la mitigación.

c) Plan de mitigación de los riesgos que originalmente excedían el RPN máximo establecido:

Riesgo 1: plan de mitigación (si por el RPN fuera necesario elaborar un plan de mitigación). Nueva asignación de S y O, con su respectiva justificación: - Severidad (S): mientras más severo, más alto es el número (usar números del 1 al 10). Justificar el motivo por el cual se asigna determinado número de severidad (S). - Probabilidad de ocurrencia (O): mientras más probable, más alto es el número (usar del 1 al 10). Justificar el motivo por el cual se asigna determinado número de (O).

Riesgo 2: plan de mitigación (si por el RPN fuera necesario elaborar un plan de mitigación).

Riesgo 3: plan de mitigación (si por el RPN fuera necesario elaborar un plan de mitigación).

14. Gestión de la calidad

Elija al menos diez requerientos que a su criterio sean los más importantes/críticos/que aportan más valor y para cada uno de ellos indique las acciones de verificación y validación que permitan asegurar su cumplimiento.

- Req #1: copiar acá el requerimiento.
  - Verificación para confirmar si se cumplió con lo requerido antes de mostrar el sistema al cliente. Detallar
  - Validación con el cliente para confirmar que está de acuerdo en que se cumplió con lo requerido. Detallar

Tener en cuenta que en este contexto se pueden mencionar simulaciones, cálculos, revisión de hojas de datos, consulta con expertos, mediciones, etc. Las acciones de verificación suelen considerar al entregable como “caja blanca”, es decir se conoce en profundidad su funcionamiento interno. En cambio, las acciones de validación suelen considerar al entregable como “caja negra”, es decir, que no se conocen los detalles de su funcionamiento interno.

15. Procesos de cierre

Establecer las pautas de trabajo para realizar una reunión final de evaluación del proyecto, tal que contemple las siguientes actividades:

- Pautas de trabajo que se seguirán para analizar si se respetó el Plan de Proyecto original: - Indicar quién se ocupará de hacer esto y cuál será el procedimiento a aplicar.
- Identificación de las técnicas y procedimientos útiles e inútiles que se emplearon, y los problemas que surgieron y cómo se solucionaron: - Indicar quién se ocupará de hacer esto y cuál será el procedimiento para dejar registro.
- Indicar quién organizará el acto de agradecimiento a todos los interesados, y en especial al equipo de trabajo y colaboradores: - Indicar esto y quién financiará los gastos correspondientes.