PupilAr: desarrollo de un equipo de videonistagmografía portátil.

Proyecto final de Ingeniería en Computación

Emiliano Funes, Gonzalo Avalos Ribas

Director: Dr. Eduardo Blotta
Co-director: Ing. Cristian Ordoñez

Facultad de Ingeniería, Universidad Nacional de Mar del Plata

Agosto 2022

PupilAr: desarrollo de un equipo de videonistagmografía portátil.

Proyecto final de Ingeniería en Computación

Emiliano Funes, Gonzalo Avalos Ribas

Director: Dr. Eduardo Blotta
Co-director: Ing. Cristian Ordoñez

Facultad de Ingeniería, Universidad Nacional de Mar del Plata

Agosto 2022

Índice general

Capítulo 1

Introducción

1.1. Marco contextual

Este proyecto se desarrolla en un marco de cooperación entre el Laboratorio de Procesamiento de Imágenes (LPI) ICYTE-UNMdP y el Hospital Privado de la Comunidad de la ciudad de Mar del Plata.

Se detalla el desarrollo de un equipo de videonistagmografía (VNG) portátil a fin de asistir en la detección de diferentes patologías neurológicas sin dependencia de conexión a la red eléctrica o a algún sistema de cómputo al momento del estudio, a diferencia de los equipos de VNG presentes en el mercado.

El equipo estará acompañado de un software de escritorio que permitirá a un profesional de la salud ver los gráficos de posición de la pupila del paciente en función del tiempo, lo que será necesario para poder ayudar en el diagnóstico de diferentes patologías, tales como:

- Vértigo asociado a migraña
- Hipo-función vestibular uni o bilateral
- Alteraciones del control oculomotor
- Enfermedad de Menière
- Vértigo posicional paroxístico benigno

1.2. Descripción del nistagmo

El término nistagmo [2], o movimiento sacádico, se utiliza para describir movimientos rápidos e involuntarios de los ojos, que pueden ser:

- Horizontales
- Verticales
- Rotatorios

Según la causa, los movimientos sacádicos pueden ser en un ojo o en ambos. El nistagmo puede afectar la visión, el equilibrio y la coordinación.

Los movimientos oculares involuntarios del nistagmo son causados por anomalías de funcionamiento en las áreas del cerebro que controlan los movimientos de los ojos. La parte del oído interno que percibe el movimiento y la posición (el laberinto) ayuda a controlar los movimientos oculares.

La videonistagmografía es el método de diagnóstico de nistagmos utilizando una señal de video.

Capítulo 2

Anteproyecto

2.1. Análisis de soluciones y antecedentes

A partir de un análisis de mercado, se determinó que la mayoría de los equipos VNG comercializados trabajan con dos cámaras [4], cada una apuntando a un ojo distinto, con resoluciones de 640x480 píxeles y tasas de muestreo de 20 a 200 cuadros por segundo, dependiendo de las características que busque reportar el equipo. En general, mayores cuadros por segundo corresponden a equipos que buscan caracterizar a la señal, mientras que los rangos más bajos corresponden a equipos de menores prestaciones que simplemente buscan detectar la existencia o no de un nistagmo.

El Cuadro 2.1 muestra algunos modelos comerciales fabricados por la empresa *VisualEyes* [4].

Característica	**2D-VOGFW**	**BG4.0USB**	**USBM2.1A**
Resolución	640 x 480	320 x 240	640 x 480
Cámaras	2	2	1 (móvil)
FPS máximo	100	100	50
Peso(gr)	305	345	-

Cuadro 2.1: Soluciones VNG del fabricante VisualEyes

Un aspecto compartido por todos los equipos de videonistagmografía del mercado es la necesidad de una computadora (de escritorio o portátil) conectada al equipo para poder realizar el procesamiento de imágenes y visualizar los resultados del estudio. Los requerimientos de hardware de estas computadoras suelen ser de gama media. En el caso de los VNG antes mencionados, los requerimientos mínimos para utilizar el software son mostrados en el Cuadro 2.2.

Característica	Valor
Procesador	Intel i5 2.5GHZ. 4 hilos.
Memoria RAM	8 GB
Disco Rígido	250 GB
Resolución mínima	1366x768

Cuadro 2.2: Requerimientos hardware de escritorio

Los dispositivos en el mercado se encuentran en un intervalo de precios entre 10.000 USD y 40.000 USD [5] [6] [7].

2.2. Perspectiva de los entregables

Es muy importante que el sistema posea la característica de ser portátil, facilitando su uso al no depender de ningún equipamiento adicional o suministro de energía externo. Es por esto que los entregables de este proyecto consisten en un sistema de recolección y muestra de datos adosado a un casco, que luego pueden ser visualizados en una aplicación ejecutándose en una computadora de escritorio.

El sistema desarrollado consistirá en:

- Casco ad-hoc que aloje una sola cámara de vídeo para obtener la secuencia de movimientos oculares del paciente y la electrónica necesaria para procesar dichos datos, extrayendo la información del movimiento de pupilas y realizando su almacenamiento. Además, poseerá un display que permita realizar una primera evaluación de las señales obtenidas por parte del especialista médico. Por otro lado, se incluirá una tarjeta Micro SD para poder guardar los datos de la prueba en una unidad portátil de almacenamiento.

- Software embebido, incluyendo un algoritmo de procesamiento digital de imágenes (PDI) para el seguimiento individual de las pupilas del paciente y el almacenamiento simultáneo de su posición en el plano.

- Software para la gestión clínica de los resultados, donde se podrán visualizar los resultados de la videonistagmografía.

Para el desarrollo se utilizará la placa STM32H747I-DISCO del fabricante STMicroelectronics, que consta de un microcontrolador de capacidad de procesamiento suficiente para la detección de pupilas, y las interfaces necesarias para conectar una tarjeta Micro SD, una pantalla y una cámara de vídeo.

2.3. Requerimientos

En esta sección se detallan los requerimientos establecidos para el equipo de VNG portátil. Para mayor detalle, consultar el documento *Especificación de Requerimientos* (Apéndice A.1.2).

2.3.1. Interfases externas

- El casco deberá contar con una pantalla adosada que permita:
  1. Calibrar la imagen de la pupila del paciente
  2. Seleccionar el tipo de estudio de VNG a realizar.
  3. Dar inicio a la prueba.
  4. Detener el estudio en curso.
  5. Mostrar información que permita saber cuándo comenzó y terminó el estudio.
  6. Ver resultados preliminares del estudio.
  7. Anunciar el estado de carga de la batería.
- El software de escritorio debe:
  1. Contener dos gráficos, uno correspondiente a movimiento vertical y otro a horizontal, donde se podrá visualizar la posición de cada ojo en su respectivo eje.
- Se deberá contar con un arreglo lumínico que el paciente seguirá con su vista para asistir al diagnóstico de nistagmos.

2.3.2. Funciones

1. Poder visualizar mediante el display integrado la posición de los ojos previo al inicio del estudio.
2. Configuración de parámetros:
   a) Tiempo del estudio a realizar.
3. Luego de la etapa de configuración, se hará una adquisición de imágenes en las cuales se buscará identificar la posición de las pupilas.
   a) Se brindará indicaciones de que el estudio se está llevando a cabo en la pantalla adosada al casco.

   b) Se podrá detener el estudio en curso.

4. Una vez finalizada la etapa de adquisición de imágenes:

   a) Se deberá mostrar en la pantalla adosada al casco la siguiente información:

      1) Aviso de que el estudio finalizó, informando al usuario en caso de haberse producido algún error.
      2) Gráfico preliminar, o datos que resuman el estudio realizado.

   b) Los datos del estudio se extraerán mediante un micro SD.

5. El software de escritorio debe:

   a) Procesar la información extraída del micro SD con el fin de graficar la posición de la pupila en función del tiempo.

6. En caso de contar con baja batería se deberá dar aviso de tal situación.

7. Se debe poder recargar la batería del casco.

2.3.3. Requisitos de rendimiento

La adquisición de imágenes se deberá realizar con una frecuencia no menor a los 20 cuadros por segundo.

2.3.4. Restricciones de diseño

1. El equipo de VNG diseñado deberá ser portátil.

2. El equipo no podrá comenzar una prueba si se detecta que la batería es baja.

3. Dada la característica de portabilidad de este diseño de VNG, se debe proveer una batería que asegure una autonomía no menor a una hora.

4. El conjunto de datos recopilados sólo podrá ser extraído una vez finalizado el seguimiento de las pupilas.

5. Los ajustes del casco deben ser regulables.

6. El peso del casco deberá ser el mínimo posible. El peso deberá ser distribuido uniformemente en la medida que el diseño lo permita.

7. Los gráficos generados deben tener unidades de Posición (en píxeles) vs Tiempo.

8. La iluminación del ojo deberá ser infrarroja. Esto genera mejor contraste entre iris y pupila.

2.3.5. Diagrama bloques

La solución propuesta contiene solo una cámara embebida en el casco junto con la placa de desarrollo y alimentación, haciéndola portátil.

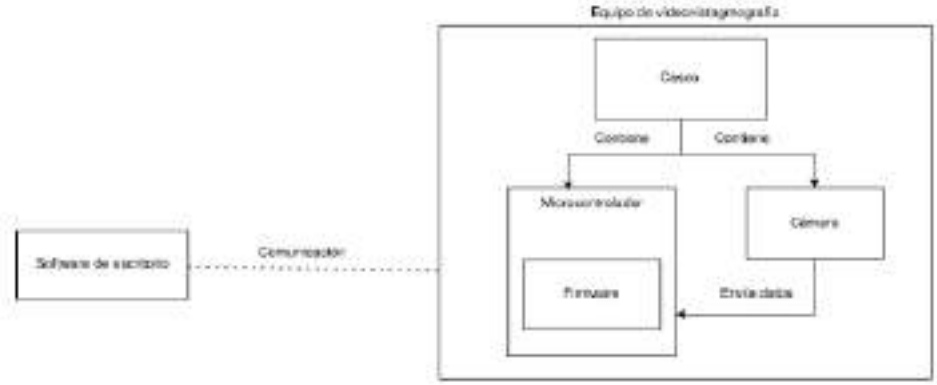


Figura 2.1: Diagrama de bloques básico de la solución.

2.4. Plan de trabajo

Las Figuras 4.1 y 4.2 muestran el diagrama de Gantt que ilustra la planificación del proyecto. Adjunto a este documento, en el *Plan de Proyecto* (Apéndice A.1.1), se encuentra la descripción específica de cada épica visible en el diagrama.

El objetivo principal del plan de proyecto es poder independizar cada una de las tareas principales del desarrollo: el firmware del microcontrolador, el casco y el software de escritorio.

2.4.1. Diagrama de Gantt proyectado

En el documento *Plan de proyecto* (A.1.1) se adjunta el diagrama de Gantt proyectado. En el Capítulo 4 se compara con la planificación ejecutada, y se analizan las divergencias.

Capítulo 3

Desarrollo

3.1. Sistema embebido

3.1.1. Introducción

Esta sección describe la solución desarrollada para el sistema embebido. El diseño se puede ver en los documentos *Especificación funcional del firmware* y *Especificación técnica del firmware*, adjuntos en el Apéndice A.1.3 y A.1.4 respectivamente.

3.1.2. Firmware

3.1.2.1. Introducción

El objetivo de esta sección es describir la solución realizada para el firmware del equipo.

Para la implementación de la solución se utilizó la placa de desarrollo STM32H747I-Disco [8], que contiene dos unidades de cómputo, un Cortex-M7 funcionando a $400MHz$, y un Cortex-M4 con frecuencia de hasta $200MHz$, que no será utilizado en el desarrollo por motivos que se explicarán mas adelante.

Se utiliza un código embebido programado en los lenguajes C y C++: C para las tareas de alto costo computacional y C++ para el desarrollo de la interfaz visual, para la cual se utiliza el conjunto de herramientas provisto por el framework de software gráfico para soluciones embebidas *TouchGFX*.

Además, el software tendrá la responsabilidad de configurar los periféricos conectados, entre ellos el display LCD, la tarjeta Micro SD (para la extracción de datos) y el módulo de la cámara DM-CAM130.

3.1.2.2. Sistema operativo multitarea

Un sistema operativo (SO) es un programa que actúa como intermediario entre el usuario, las tareas o aplicaciones, y el hardware. Se encarga de gestionar los recursos del sistema de cómputo, siendo estos: el procesador, la memoria, los dispositivos de E/S y la información. Es multitarea cuando puede administrar el tiempo del procesador en forma transparente al usuario, dando la sensación de ejecución simultánea de tareas.

Un sistema operativo de tiempo real (RTOS) es un caso particular de sistema operativo multitarea en el cual se prioriza el cumplimiento de plazos temporales de forma estricta.

En general, un RTOS se puede pensar como un conjunto de tareas, la mayoría de las cuales son provistas por el usuario, y un kernel mínimo provisto por el propio RTOS, que suele incluir un pequeño conjunto de funciones. Esto lo hace suficientemente ligero como para ser incluido en el firmware de un microcontrolador. El uso de esta herramienta simplifica la gestión del uso del procesador, garantizando que en ciertos momentos solo hagan uso del mismo las tareas que, por ser críticas, lo requieran.

A continuación, se detallan las tareas implementadas en este diseño:

- Tarea Interfaz: encargada de mostrar la pantalla y actualizarla en tiempo real para procesar las actividades del usuario. La misma está implementada a través de la herramienta TouchGFX.

- Tarea VNG: encargada de procesar la imagen entregada por la interfaz de la cámara y encontrar las coordenadas de la pupila antes de que ingrese una nueva.

Para determinar la posición de la pupila, la tarea de cómputo debe ser capaz de ejecutarse en un tiempo menor al tiempo entre imágenes, dado que el atraso en el procesamiento de una imagen provoca que un nuevo cuadro solape al anterior, no procesado, con su correspondiente pérdida.

Es importante notar que gracias a la optimización realizada sobre el algoritmo de detección de pupila (sección 3.1.3.3), éste es capaz de ejecutarse a una velocidad aproximada de 36 cuadros por segundo para ambas pupilas, superando los 30 cuadros por segundo proporcionados por la cámara utilizada, cumpliéndose así los requerimientos de tiempo explicados anteriormente. Por lo tanto, en los instantes en donde se debe realizar el procesamiento del cuadro de video para la identificación de las pupilas, esta tarea debe adueñarse del procesador hasta concluirlo.

La Figura 3.1 ilustra la pérdida de una imagen.

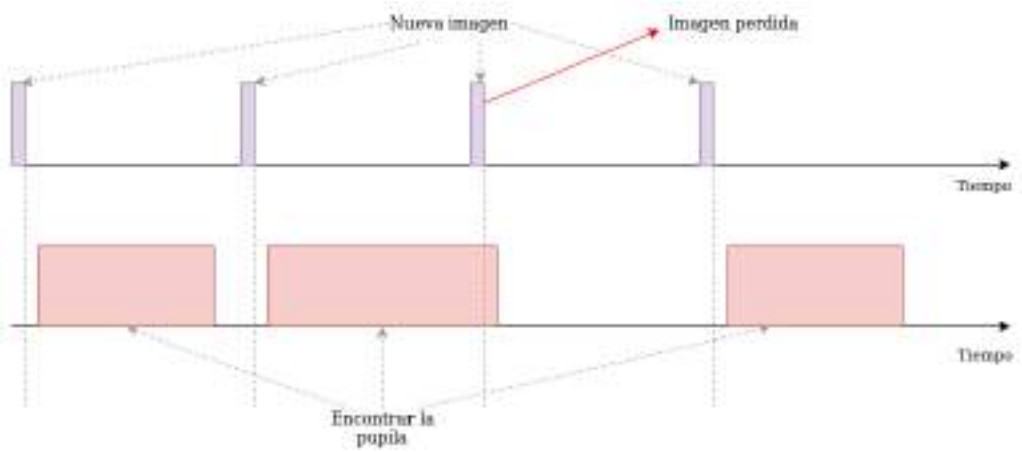


Figura 3.1: Diagrama de tiempos del sistema.

Para el desarrollo del proyecto se utilizó el sistema operativo en tiempo real FreeRTOS [10], que es uno de los sistemas operativos de tiempo real con mayor cuota de mercado.

Algunas de sus características son:

- Usado por empresas reconocidas, confiable.
- Kernel pequeño y de bajo consumo.
- Soporte de alta cantidad de arquitecturas de microprocesador.
- Gratuito.
- Soporte a largo plazo.

Otro aspecto por el cual se eligió el uso de FreeRTOS es la facilidad para implementación provista por la experiencia adquirida en cursos previos.

3.1.2.3. Interfaz Visual

TouchGFX [11] es un conjunto de herramientas que permite el diseño e implementación de una interfaz visual optimizada para los microcontroladores STM32.

El desarrollo de la interfaz se realiza mediante el uso del lenguaje de programación C++, aunque también se tiene acceso a un diseñador gráfico. Se utiliza programación orientada a objetos para describir la interfaz de una manera sencilla y extensible. Además, es configurable para que se ejecute como una tarea de FreeRTOS.

En un principio se contempló utilizar el segundo microcontrolador embebido en la placa de desarrollo para manejar la tarea de TouchGFX, dejando el Cortex-M7 exclusivamente para la tarea de detección de pupilas. La falta de documentación al momento de la realización del proyecto hizo esto imposible.

Para minimizar el impacto de este inconveniente, se decidió suspender la Tarea Interfaz durante la ejecución del algoritmo de detección de pupila, de forma que esta no compita por el procesador, estando todos los recursos disponibles para la Tarea VNG. Una vez terminado el procesamiento, la Tarea VNG es la encargada de autosuspenderse y reactivar a la Tarea Interfaz. Por lo tanto, siempre que una tarea se encuentra en ejecución, la otra se encuentra suspendida, posibilitando que siempre cada una de ellas pueda tener asignados todos los recursos del sistema que se requieran. Si bien este no es el caso de uso típico de un RTOS, soluciona el problema antes planteado.

3.1.2.4. Pantallas de la solución embebida

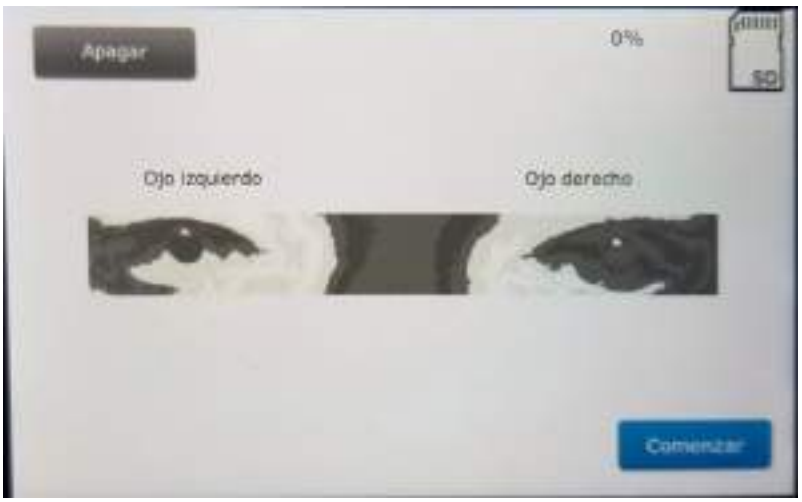


Figura 3.2: Pantalla de calibración

Figura 3.3: Pantalla de introducción de datos del paciente

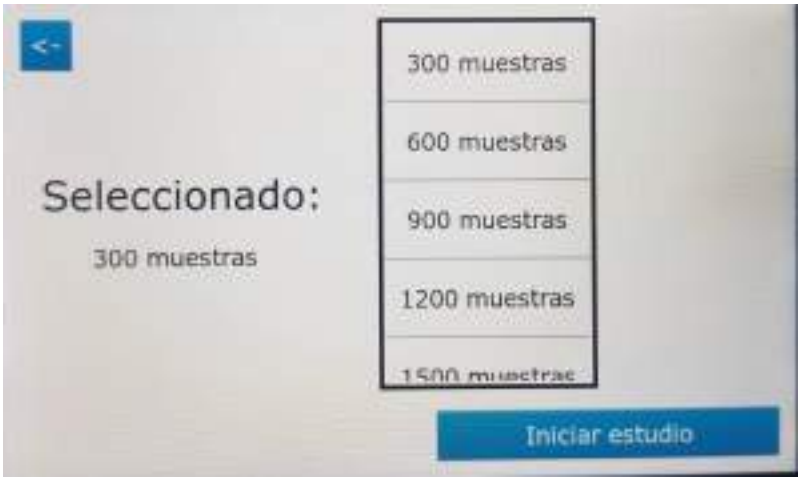


Figura 3.4: Pantalla de selección de estudio

Figura 3.5: Pantalla de estudio en curso

Figura 3.6: Pantalla de muestra de resultados

3.1.3. Detección de centro de pupilas

En esta sección se describe toda la información referida al algoritmo detección de centro de pupilas implementado en este desarrollo.

Inicialmente, se describen los algoritmos de procesamiento de imágenes implementados para alcanzar el resultado buscado. Estos algoritmos se pueden dividir principalmente en dos grupos o etapas: preprocesamiento y detección. La etapa de preprocesamiento se encarga de acondicionar la imagen de entrada para que luego sea procesada por la siguiente etapa. Mientras

que la etapa de detección se encarga de encontrar las coordenadas espaciales del centro de pupila buscado.

Luego, se describen las optimizaciones realizadas al algoritmo de detección de pupilas suministrado por el laboratorio LPI, el cual se utilizó como punto de partida para el desarrollo de este proyecto.

Finalmente, se detallan las pruebas que se realizaron para evaluar el desempeño del sistema implementado.

3.1.3.1. Preprocesamiento

La etapa de preprocesamiento es un paso importante en cualquier algoritmo de procesamiento de imágenes. Como se mencionó anteriormente, esta etapa se encarga preparar la imagen de entrada para que la siguiente etapa, en este caso el algoritmo detección, pueda desempeñar su tarea correctamente.

En la implementación desarrollada, esta etapa se puede dividir en tres pasos:

- Transformación de espacio de color.

  La cámara DM-CAM130 toma imágenes en formato RGB565, el cual describe a cada píxel de la imagen con 16 bits: cinco para el canal rojo, seis para el verde y cinco para el azul. Dado que el algoritmo de detección implementado funciona con imágenes en escala de grises, se deben extraer los canales R, G y B de la imagen, transformarlos a formato de 8 bits y realizar un promedio de los mismos. El procedimiento de transformación de espacio de colores para cada píxel se realiza mediante la siguiente ecuación:

$$P_{Grayscale} = \frac{\frac{P_{RGB}\&0xF800}{2^8} + \frac{P_{RGB}\&0x07E0}{2^3} + P_{RGB}\ \&\ 0x001F \times 2^3}{3} \quad (3.1)$$

  Si llamamos P_{RGB} a un píxel en formato RGB565, y $P_{Grayscale}$ a un píxel en formato de grises, el primer termino del numerador extrae la componente roja, el segundo termino la verde y el tercero la componente azul, y por último se promedian las tres.

- Suavizado.

  La imagen en escala de grises obtenida en el paso anterior alimenta la etapa de suavizado que se encarga de suavizar los datos de la imagen de entrada a través de un filtrado gaussiano, con el objetivo de mejorar los

resultados del procesamiento. Si bien en la búsqueda de acotar costos de procesamiento resulta tentador no implementar esta etapa, esto puede generar una perdida significativa en la precisión del método.

El filtro de suavizado consiste en la convolución de la imagen de entrada por un filtro gaussiano de una sola dimensión, detallado en la Eq. 3.2.

A modo de ejemplo, en la Figura 3.7 se puede ver una imagen de entrada tomada con el dispositivo final, y en la Figura 3.8 la imagen filtrada. Si bien la diferencia es sutil, se puede ver como los bordes de la pupila pasan de estar bien definidos a verse más difusos, es decir que se eliminan componentes de alta frecuencia que pueden generar dificultades en la detección.

$$F = [0{,}160945,\ 0{,}218261,\ 0{,}241587,\ 0{,}218261,\ 0{,}160945] \tag{3.2}$$

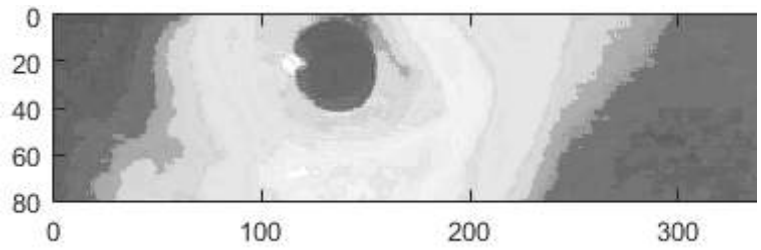


Figura 3.7: Imagen de entrada.

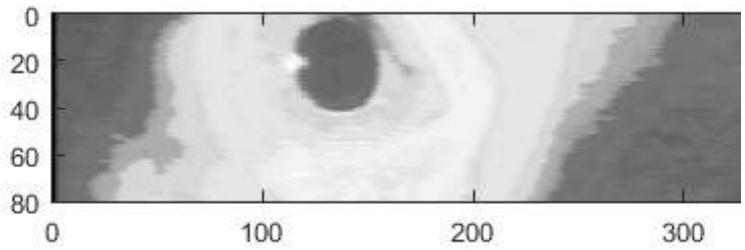


Figura 3.8: Imagen de entrada filtrada.

- Selección de región de interés.

  Dado que la imagen de entrada tiene una dimensión de 640 por 80 píxeles y que la cámara del sistema se encuentra centrada con respecto a la cara del paciente, se puede considerar que el ojo izquierdo está contenido en la primera mitad de la imagen, y el ojo derecho en la segunda. De esta forma, para encontrar la posición de ambas pupilas, primero se aplica el algoritmo de detección, que se detalla en la Sección 3.1.3.2, para la primera mitad de la imagen, y luego para la segunda.

3.1.3.2. Método de detección

El método de detección de pupila que se implementa en este proyecto se basa en el empleo de isocurvas, las cuales permiten obtener invariación a los cambios de iluminación y a las rotaciones, manteniendo un bajo costo computacional [1].

Las isocurvas son una de las curvas características de cualquier superficie. En el campo de procesamiento digital de imágenes se definen como curvas que conectan píxeles de una misma intensidad. En otras palabras, si visualizáramos una imagen en tres dimensiones, donde la vertical fuese la intensidad, las isocurvas serian cortes horizontales de la función. Estas curvas son muy útiles para caracterizar la geometría de una superficie, y utilizadas exitosamente en el campo de visión computacional [23] [24].

Partiendo de la premisa de pupila circular, se calculan los centros de curvatura de las isocurvas para generar un espacio de acumulación de votos que se utiliza para encontrar el centro de la pupila buscada.

Tal como muestra la Figura 3.9 el método se puede descomponer en tres pasos fundamentales: calculo de centros de curvatura, generación de mapa de centros y filtrado de salida, las cuales se detallan a continuación.

Figura 3.9: Diagrama del método de detección implementado

Cálculo de centro de curvatura

Esta etapa consiste en el cálculo de los centros de curvatura asociados a la familia de isocurvas que describen la imagen de entrada, para lo que es necesario calcular las derivadas parciales primeras y segundas.

De esta forma, el centro de curvatura C se puede estimar mediante la Eq. 3.3, a través de la suma vectorial de la posición de un píxel y un vector desplazamiento $D = (D_x, D_y)$. Este vector está compuesto por dos componentes, una horizontal D_x y una vertical D_y, que se pueden obtener mediante la Eq. 3.4 y Eq. 3.5, siendo A la imagen de entrada filtrada, donde $A \in \mathbb{R}^{MxN}$ con M y N la altura y ancho de la imagen, respectivamente.

$$C = (x + D_x, y + D_y) \tag{3.3}$$

$$D_x = -\frac{A_x(A_x^2 + A_y^2)}{A_y^2 A_{xx} - 2A_{xy}A_x A_y + A_x^2 A_{yy}} \tag{3.4}$$

$$D_y = -\frac{A_y(A_x^2 + A_y^2)}{A_y^2 A_{xx} - 2A_{xy}A_x A_y + A_x^2 A_{yy}} \tag{3.5}$$

con:

$$A_x = \frac{\partial A}{\partial x} \tag{3.6}$$

$$A_y = \frac{\partial A}{\partial y} \tag{3.7}$$

$$A_{xy} = \frac{\partial^2 A}{\partial x \partial y} \tag{3.8}$$

$$A_{xx} = \frac{\partial^2 A}{\partial x^2} \tag{3.9}$$

$$A_{yy} = \frac{\partial^2 A}{\partial y^2} \tag{3.10}$$

De esta manera, la forma que se elige para implementar esta serie de cálculos de derivadas se detalla en la Subsección 3.1.3.3.

Mapa de centros

El mapa de centros es una matriz de acumulación de votos que se genera a partir de los centros de circunferencia de las isocurvas, teniendo en cuenta lo siguiente:

- Debido a que la región de la pupila suele ser más oscura que el iris y la esclerótica del ojo, se consideran solamente los votos generados por curvaturas con signo positivo. La curvatura es positiva si la intensidad de la región exterior a la curva es más clara que el interior. La curvatura se puede estimar como:

$$\kappa = -\frac{A_y^2 A_{xx} - 2A_{xy}A_x A_y + A_x^2 A_{yy}}{(A_x^2 + A_y^2)^{\frac{3}{2}}} \tag{3.11}$$

- Se utiliza una matriz de pesos para ponderar los votos generados por los centros de isocurvas. Con esto se busca que las isocurvas con forma circular tengan mayor peso que el resto, ya que debido a reflejos, sombras, ruido, etc. la forma de las isocurvas no siempre coincide con la forma del objeto. Para esto se utiliza un factor que mide la característica de curvatura de la región:

$$\gamma = \sqrt{A_{xx}^2 + 2A_{xy}^2 + A_{yy}^2} \tag{3.12}$$

  Entonces, los votos que provienen de isocurvas con un alto valor de γ tendrán mayor peso que el resto.

- Solo se consideran las isocurvas cuyo centro de curvatura se encuentre dentro de la dimensión de la imagen y además cuya magnitud del vector desplazamiento cumpla:

$$R_{min} < |D| < R_{max} \tag{3.13}$$

  donde R_{min} y R_{max} son el radio mínimo y máximo de pupila, respectivamente, que se utilizan para realizar la búsqueda del centro de pupila.

La Figura 3.10 muestra los resultados del cálculo de centro de curvatura para el ejemplo de la Figura 3.7, donde se puede apreciar como la pupila se corresponde con la zona donde existe mayor densidad de picos de mayor altura.

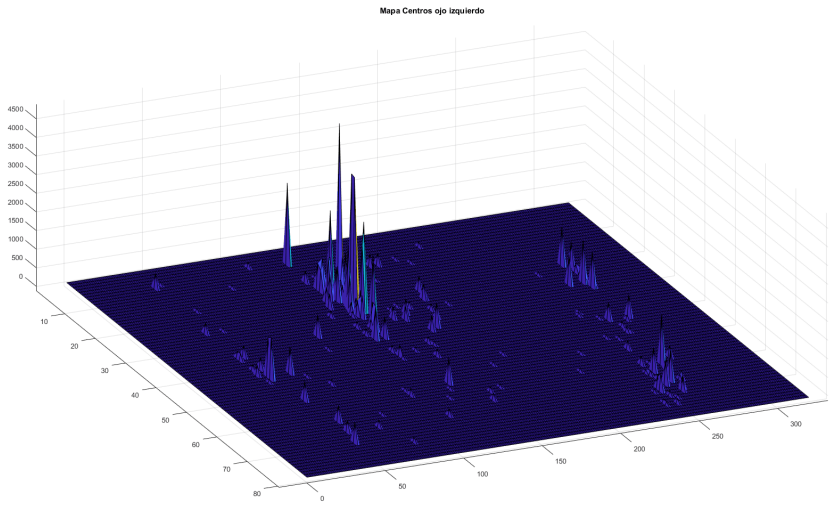


Figura 3.10: Mapa de centros.

Filtrado de salida del mapa de centros

El mapa de centros obtenido en la etapa anterior se caracteriza por su alto nivel de discretización. Debido a que es de interés obtener el área donde existe mayor densidad de picos y no necesariamente con los picos mas altos, se suaviza el mapa de centros realizando un proceso de convolución empleando el kernel gaussiano que se detalla en [1].

A modo de ejemplo, en la Figura 3.11 se muestra el mapa de centros obtenido anteriormente, luego de aplicarle el proceso de convolución con el kernel.

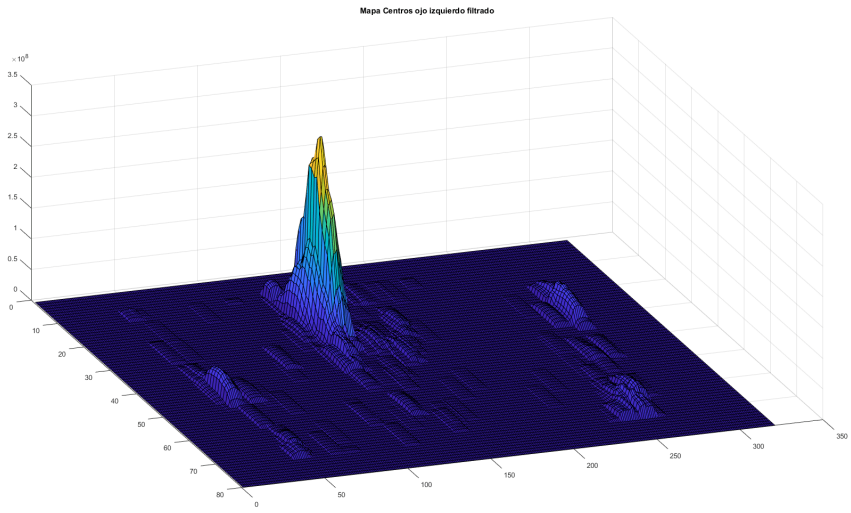


Figura 3.11: Mapa de centros filtrado.

Las coordenadas espaciales del máximo global del mapa obtenido representan el centro de la pupila detectado.

Siguiendo con el ejemplo anterior, en la Figura 3.12 se grafica la imagen original junto con el centro de pupila detectado luego de realizar la búsqueda del máximo global del mapa de centros filtrado.

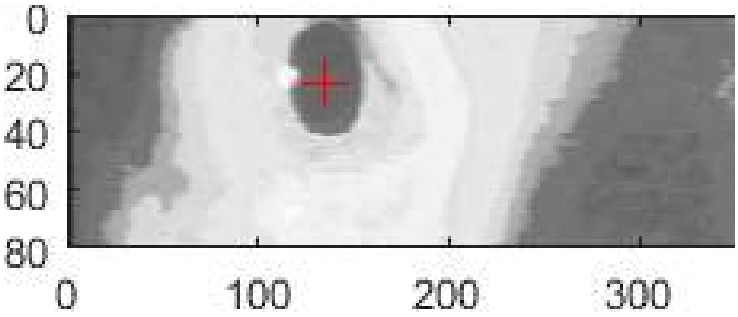


Figura 3.12: Posición de pupila encontrada.

3.1.3.3. Optimizaciones realizadas

Como se mencionó anteriormente, el algoritmo de detección de pupilas implementado en este proyecto fue suministrado por el laboratorio LPI. Sin embargo, este algoritmo era una versión inicial que debía ser depurada con el objetivo de reducir el uso de memoria RAM y, principalmente, el tiempo de procesamiento.

En esta linea, uno de los requerimientos más importantes del desarrollo es el de mantener la tasa de captación de imagen superior a 20 cuadros por segundo, ya que si no se cumple este requisito ciertos movimientos oculares podrían no detectarse. En este sentido, la implementación previa del algoritmo, realizada en el LPI utilizando una placa STM32F4DISCOVERY, no permitía cumplir con estos niveles de rendimiento.

El Cuadro 3.1 muestra la cantidad de cuadros por segundo por tamaño de matriz del algoritmo provisto ejecutándose en la placa mencionada.

Tamaño matriz (píxeles)	Cuadros por segundo a 144Mhz
16000	12
14400	15
7200	32

Cuadro 3.1: Cuadros por segundo según tamaño de matriz en STM32F4DISCOVERY.

Cabe destacar que la implementación en la placa no permitía el uso de matrices de mayor dimensión, debido a la escasez de memoria RAM, específicamente 192 KB. Además, el algoritmo no implementaba filtrado de salida y se submuestraba la imagen de entrada, trabajando realmente con la mitad de los píxeles.

Debido a las deficiencias anteriormente mencionadas, se decidió reimplementar la totalidad del algoritmo incorporando optimizaciones que permitiesen lograr el rendimiento que se buscaba, tanto en tiempo (cantidad de cuadros por segundo) como en espacio (mínima huella en la memoria RAM).

El algoritmo se desarrolló como una librería escrita en el lenguaje C y capaz de ser utilizada en sistemas embebidos, computadoras de escritorio u otros. Acepta tamaños de matriz de entrada de hasta 65535 por 65535 píxeles con valores de 0 a 255. El costo de cómputo crece de manera lineal con el tamaño de la matriz. La memoria RAM máxima ocupada para una matriz de tamaño MxN es de $MxNx7$ *bytes*.

Las optimizaciones incorporadas se pueden resumir en tres grupos:

- Optimizaciones del uso de memoria RAM.
- Optimizaciones de cálculos.
- Optimizaciones del compilador.

Optimizaciones del uso de memoria RAM.

- No guardar matrices que no son necesarias:

  Dado que se planteó el objetivo de ejecutar el algoritmo de detección en un sistema de pocos recursos (STM32H747I-DISCO), es necesario mantener una huella de memoria relativamente baja.

  La implementación original calculaba y guardaba todas las derivadas en memoria simultáneamente de forma innecesaria. Para mejorar este aspecto, la nueva implementación solo guarda las matrices de derivadas primeras, y el resto son computadas en el momento que se necesiten. Si bien es un pequeño cambio, no genera necesidad de recálculo y disminuye las matrices de derivadas necesarias de cinco a dos.

- Separar y serializar el trabajo sobre las matrices:

  Tal como se mencionó en la Sección 3.1.3.1, el algoritmo se ejecuta una vez por cada pupila, es decir, una vez por cada mitad de la imagen captada por la cámara, lo que permite trabajar con matrices que ocupan la mitad del tamaño de la imagen original y que luego son reescritas al trabajar con la otra mitad de la imagen.

  Esta optimización permite reducir considerablemente la cantidad máxima de memoria ocupada, sin generar agregados en la cantidad de operaciones.

Optimizaciones de cálculos.

- Remplazar la forma en la que se realizan las derivadas:

  En la publicación original del algoritmo [1], se proponía el cálculo de las derivadas mediante una convolución doble empleando un kernel de dimensión 1x5. Esto conlleva un total de diez operaciones de producto y ocho operaciones de suma por píxel por derivada.

  Una alternativa para minimizar la cantidad de operaciones necesarias es la utilización de un algoritmo de diferencias finitas de primer orden. Usando este algoritmo se puede aproximar cada derivada realizando solo dos operaciones de producto y una operación de suma por píxel. Si llamamos A a la imagen de entrada filtrada y A(x,y) al píxel en la posición (x,y), las derivadas se calculan como:

$$A_x(x,y) = \frac{A(x+1,y) - A(x-1,y)}{2} \tag{3.14}$$

$$A_y(x,y) = \frac{A(x,y+1) - A(x,y-1)}{2} \tag{3.15}$$

$$A_{xx}(x,y) = \frac{A_x(x+1,y) - A_x(x-1,y)}{2} \tag{3.16}$$

$$A_{xy}(x,y) = \frac{A_x(x,y+1) - A_y(x,y-1)}{2} \tag{3.17}$$

$$A_{yy}(x,y) = \frac{A_y(x,y+1) - A_y(x,y-1)}{2} \tag{3.18}$$

El criterio adoptado para considerar correcta a una detección es que la coordenada hallada como centro de pupila se encuentre dentro de la misma. De esta forma, para el conjunto de 60 imágenes provistas por el LPI, el algoritmo implementado con el nuevo método de calculo de derivadas entrega la misma cantidad de detecciones correctas que la implementación anterior.

- Minimizar la cantidad de operaciones con números flotantes:

  Las operaciones con números flotantes, en general, llevan más tiempo que aquellas con números enteros. Por este motivo, en algunas ocasiones se puede trabajar con números enteros sin perder precisión, lo que permite acelerar algoritmos de cómputo.

  En este sentido, si analizamos el kernel que se utiliza para realizar el filtrado de entrada (Eq. 3.2), la convolución de la imagen de entrada con dicho kernel implica el calculo de cinco productos flotantes y cuatro sumas de flotantes por píxel de imagen. Para mejorar este aspecto, se utiliza el filtro F' (Eq. 3.19) que necesita cinco productos enteros, cuatro sumas enteras y una división entera, que resulta ser más veloz e igual de preciso.

$$F' = [160945,\ 218261,\ 241587,\ 218261,\ 160945]\frac{1}{1000000} \tag{3.19}$$

- Separar kernel gaussiano del filtrado de salida:

  La aplicación del filtrado de salida es particularmente costosa. Conlleva un total de 81 operaciones de producto y 80 operaciones de suma por píxel.

  Debido a que los kernel gaussianos son separables, es posible reemplazar el kernel mencionado en el párrafo anterior por dos kernel de una

dimensión. Si llamamos H_0 al filtro gaussiano de 9x9, entonces se puede encontrar H_1, filtro de 9x1, y H_2 de 1x9 tal que se cumpla la Eq. 3.20.

$$H_0 = H_1 H_2 \tag{3.20}$$

Dado que H_1 y H_2 son filtros fila y columna respectivamente, se cumple que:

$$H_0 = H_1 H_2 = H_1 * H_2 \tag{3.21}$$

Esto permite aplicar la propiedad asociativa de la convolución para modificar la operación a realizar. Si se llama F a un mapa de centros cualquiera, se cumple la Eq. 3.22.

$$F * H_0 = F * (H_1 * H_2) = (F * H_1) * H_2 \tag{3.22}$$

Si bien con este enfoque es necesario realizar dos convoluciones, una por filas y otras por columnas, se reduce la cantidad de operaciones a 18 productos y 16 sumas por píxel.

Optimizaciones del compilador.

Ciertas directivas al compilador permiten optimizar la traducción, minimizando el tiempo de ejecución al usar instrucciones que son más rápidas o intentando reconocer patrones en el código escrito que permitan ser optimizados. En este caso se habilitaron las directivas -O3 ofrecidas por el compilador GCC GNU [17], que buscan minimizar el tiempo de ejecución del código.

3.1.3.4. Resultados

En esta sección se describen los resultados obtenidos luego de realizar distintas pruebas para evaluar el desempeño del sistema implementado. Estas pruebas se realizaron inicialmente sobre una computadora de escritorio y luego sobre la plataforma STM32H747IDISCO.

El desempeño del sistema se evalúa teniendo en cuenta dos aspectos importantes: tiempo de procesamiento y precisión. El primer parámetro se refiere al tiempo que tarda el algoritmo en ejecutar todas las operaciones necesarias desde que recibe la imagen de entrada hasta que encuentra la posición del centro de pupila. El segundo parámetro se refiere a la razón entre cantidad de detecciones correctas y cantidad de detecciones incorrectas.

Para evaluar el tiempo de procesamiento del algoritmo, se calculó la cantidad de cuadros por segundo (FPS) que se obtuvieron en tres escenarios distintos:

1. STM32H747I-DISCO: se capturó una sola imagen, la cual fue procesada mil veces con el algoritmo de detección de pupila.

2. Sistema VNG: se captó una señal de video, en la que cada cuadro se analiza en tiempo real con el algoritmo de detección de pupila. La diferencia con el escenario anterior es que en esta prueba intervienen diferentes retardos temporales generados por el funcionamiento del sistema, tales como los que genera el procesamiento de interrupciones.

3. Computador de escritorio: se ejecutó 100.000 veces el algoritmo de detección sobre una misma imagen.

Cabe destacar que el computador de escritorio en el cual se realizaron las pruebas consta de un procesador AMD Ryzen 5 3600x, sistema operativo Windows 10 y 16GB de RAM.

Plataforma	CPU	Cuadros por segundo
STM32H747IDISCO	CORTEX-M7	36
Sistema VNG	CORTEX-M7	30
Computadora escritorio	AMD RYZEN 5 3600x	1070

Cuadro 3.2: Cuadros por segundo según plataforma.

Si bien no se estableció un requerimiento específico en cuanto a la precisión en la detección de pupilas, se lo considera una característica fundamental del sistema.

Las primeras pruebas se realizaron en la plataforma de escritorio con un set de datos de 70 imágenes con diferentes niveles de iluminación provistas por el LPI. En esta etapa del proyecto, al tratarse de un prototipo, la detección se considera correcta cuando el píxel calculado como centro de la pupila se encuentra dentro de la misma. La detección fue correcta para 67 de las 70 imágenes.

La Figura 3.13 muestra algunas de las imágenes detectadas correctamente, donde se marca con un símbolo '+' la posición del centro de pupila encontrado.

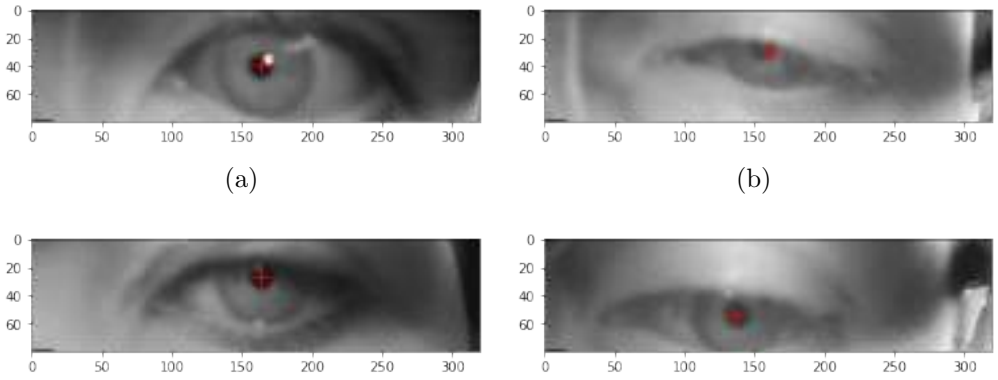
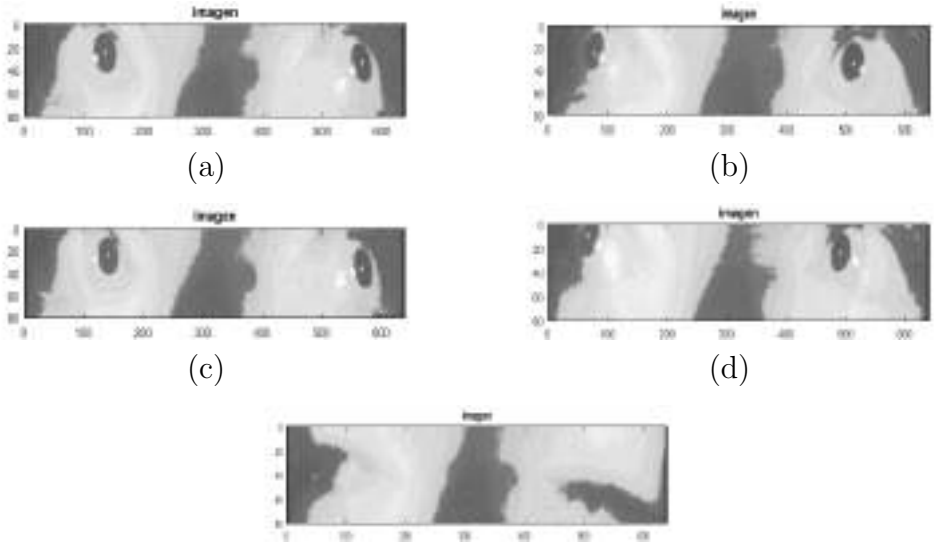


Figura 3.13: Detección de centro de pupila en imágenes provistas por el laboratorio

El siguiente enfoque adoptado para medir la precisión del sistema fue recolectar cien imágenes provistas por la cámara empotrada en el casco para luego ser analizadas en la computadora de escritorio.

Las cien imágenes se tomaron de los ojos de un paciente presuntamente sano, en una habitación con baja iluminación.

Utilizando el mismo criterio que antes, la detección fue correcta para 90 de las 100 imágenes recolectadas. Es importante destacar que algunas de estas imágenes se tomaron durante parpadeos del paciente, generando detecciones incorrectas. La Figura 3.14 muestra algunas de las imágenes detectadas correctamente, donde se marca con un símbolo '+' la posición del centro de pupila encontrado. En la subfigura (e) de la misma se puede ver un parpadeo que genera una detección errónea.



(a) (b) (c) (d) (e) Parpadeo

Figura 3.14: Detección de centro de pupila en imágenes recolectadas por sistema VNG

La ultima etapa de prueba consistió en la visualización de los resultados obtenidos en el sistema integrado, es decir, en el gráfico mostrado en la pantalla empotrada en el casco al finalizar un estudio de videonistagmografía. El objetivo es poder asegurarse que los movimientos oculares son reconocidos correctamente por el sistema con diferentes pacientes. El criterio adoptado para esta etapa de prueba resulta más cualitativo que cuantitativo.

El procedimiento de las pruebas fue el siguiente: se le colocó el casco a diferentes pacientes presuntamente sanos, y primero se les pidió que siguieran las luces que oscilaban de izquierda a derecha cada cinco segundos, y luego que miraran directamente a la cámara. En el primer caso, el resultado esperado era ver una onda símil cuadrada en el gráfico de posición de pupila en función del tiempo que arroja en pantalla el sistema VNG, y en el segundo caso una constante.

La Figura 3.15 muestra el resultado del seguimiento de una pupila quieta en el sistema VNG. En la misma se puede apreciar una tendencia constante con algunos sobrepicos, que pueden deberse a condiciones insatisfactorias de iluminación o a parpadeos del paciente. En general, además se observan ciertas fluctuaciones o *jitter* en la medición.

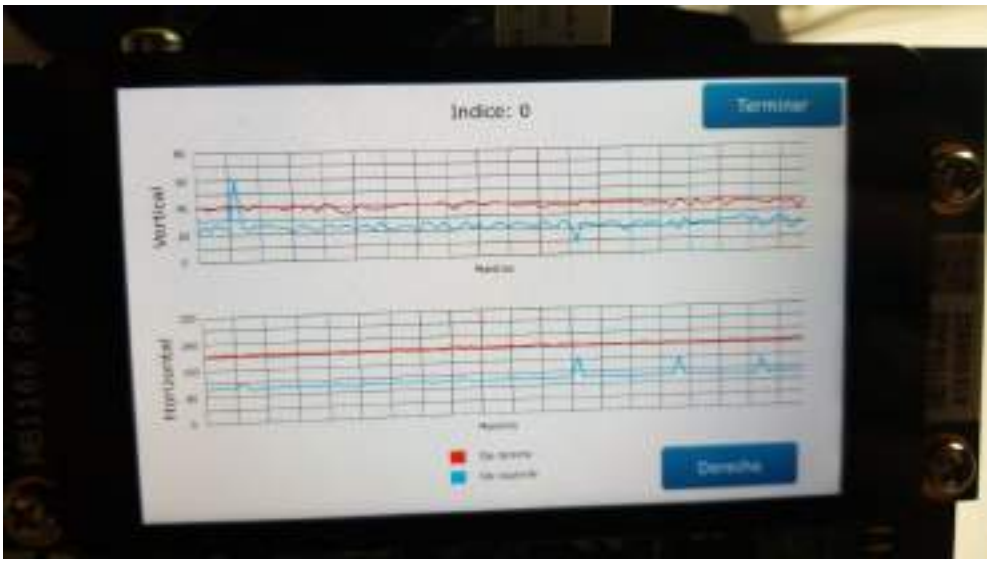


Figura 3.15: Seguimiento de pupila quieta en sistema VNG

Si bien se podría haber profundizado en las pruebas de esta etapa de una forma más cuantitativa, se decidió que no era necesario, pues se trata de un prototipo y una prueba de concepto.

Es de importancia destacar que si bien se tenia intención de probarlo en pacientes enfermos, esto no fue posible debido al daño que sufrió la placa, detallado en la Sección 4.3.3.

3.1.4. Almacenamiento de resultados

Al finalizar el estudio se guarda en un archivo de texto que almacena el resultado del mismo. El nombre del archivo queda definido como:

$$IdPaciente\_NroAleatorio.txt$$

El número aleatorio se utiliza para evitar que dos estudios con el mismo paciente generen archivos con el mismo nombre. El inconveniente de esta implementación es que resulta dificultoso encontrar un criterio coherente para el ordenamiento de archivos dentro de la tarjeta extraíble. Esto es porque un archivo generado en un tiempo T0 podría tener en su nombre un número aleatorio mayor al de un archivo generado en un tiempo posterior T1, provocando que al ordenar los archivos siguiendo un criterio alfabético, el primer archivo parezca ser el último generado.

Como alternativa a esta solución, se consideró concatenar el nombre del paciente a una marca de tiempo obtenida mediante el Real-Time Clock

(RTC) integrado en el sistema. El inconveniente de esta alternativa es que, al no poseer conexión a internet ni una pila que lo actualice con el sistema apagado, el apagado del dispositivo provocaría que al prenderse nuevamente la fecha se encuentre desactualizada, de forma que se generaría un nombre de archivo confuso para el usuario.

El contenido del archivo es la posición de cada una de las pupilas en el tiempo en formato binario. Si bien esto dificulta la lectura del archivo de texto a simple vista, minimiza el tamaño del archivo, lo que haría que en una misma tarjeta de memoria se puedan escribir más archivos.

3.1.5. Interfaz de conexión de la cámara digital

DCMI [12] es un bus de datos paralelo y sincrónico que permite conectarse con diferentes módulos de cámara CMOS y soporta diferentes tipos de formatos de datos. Es soportado por la mayoría de los microcontroladores STM32 [13].

La cámara DM-CAM130 utiliza un sensor CMOS denominado OV9655 que permite captar imágenes hasta una máxima frecuencia de muestreo de 30 cuadros por segundo (FPS).

Originalmente la cámara está configurada para funcionar a 15 FPS, por lo que para utilizar la cámara a 30 FPS es necesario modificar el pre-escalado de la señal de reloj interna del módulo. Para esto se debe modificar el registro en la dirección 0x11 llamado 'CLKRC' asignándole el valor 0x01, que se corresponde con un factor de escala igual a 1.

Además, la cámara cuenta con un controlador automático de ganancia y un controlador automático de exposición. Para garantizar los 30 FPS es necesario mantener niveles de iluminación relativamente altos. En caso de no tener la iluminación correcta, la cámara tiende a pasar más tiempo exponiendo a fin de lograr mejor contraste entre el punto más iluminado y el más oscuro.

3.1.6. Alimentación del sistema

Para la alimentación de la placa se determinó el uso de baterías AAA conectadas en serie por ser un recurso estándar fácilmente disponible en el mercado, en sus alternativas recargable o desechable. Para el desarrollo del prototipo se utiliza, por estar disponible, una batería de Litio 18650 capaz de proveer 2600mAh. Además consta de una tensión nominal de 3.7v.

La placa, en ejecución, tiene un consumo máximo de 600mA a 9V de entrada. Para adaptar la tensión se utilizó un convertidor boost MT3608 [19].

Considerando que la eficiencia mínima reportada en la hoja de datos del convertidor boost es del 80 %, se puede estimar que el dispositivo tendría una autonomía de 1.42 horas, superior a la autonomía mínima estipulada como requerimiento.

3.1.7. Iluminación de los ojos

Para la iluminación de los ojos se utilizó el circuito indicado por la Figura 3.16. La intensidad total aplicada sobre los ojos es menor que 2,8 mW/cm^2 con una longitud de onda de 950 nm.

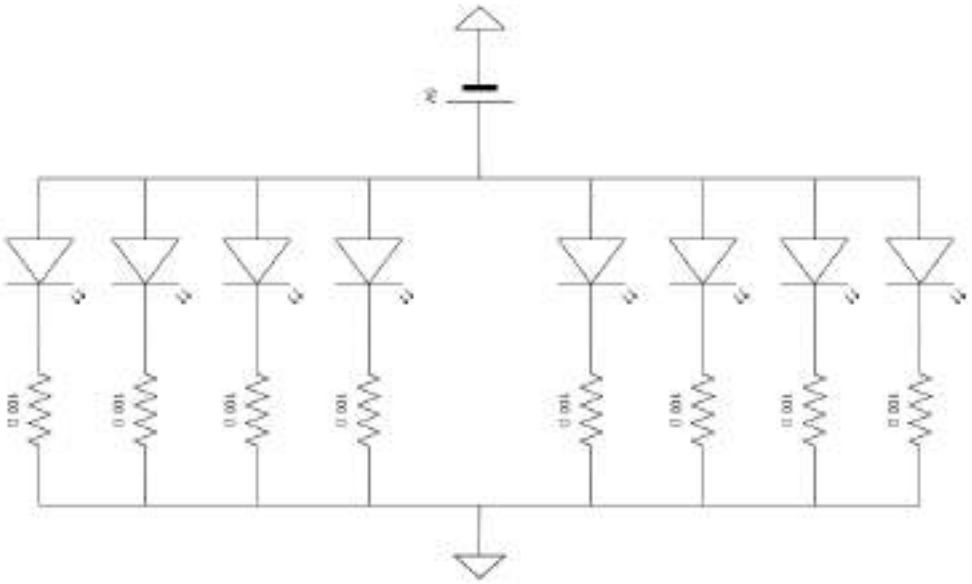


Figura 3.16: Circuito LED.

En cuanto a la cantidad de LED utilizados, se optó por usar la mínima cantidad de unidades que permitan una iluminación suficiente para que la cámara pueda exponer en tiempos cortos, de forma de poder captar imágenes a una velocidad cercana a los 30 FPS.

En cuanto a la disposición de los LED, se eligió una topología en forma de rombo, a partir de una metodología de prueba y error. De esta forma, se generan buenos niveles de la iluminación en la pupila y sus alrededores.

3.1.8. Limites según la norma IEC-62471

La Comisión Electrotécnica Internacional (IEC) estableció estándares seguros de exposición del ojo humano a diferentes fuentes de radiación lumínica.

En particular, los limites de exposición infrarroja máxima (entre longitudes de onda de 770 y 3000 nm) se define en 10 mW/cm^2 [20]. De acuerdo a lo detallado en la sección 3.1.7, el prototipo construido cumple con los estándares de esta norma.

3.1.9. Interconexión del sistema

Hasta ahora, se describieron las partes separadas, es imperante mostrar la interconexión de ellas. La Figura 3.17 muestra un diagrama de bloques de las conexiones de diferentes elementos importantes dispuestos en el casco, con las respectivas salidas que interactúan con la placa.

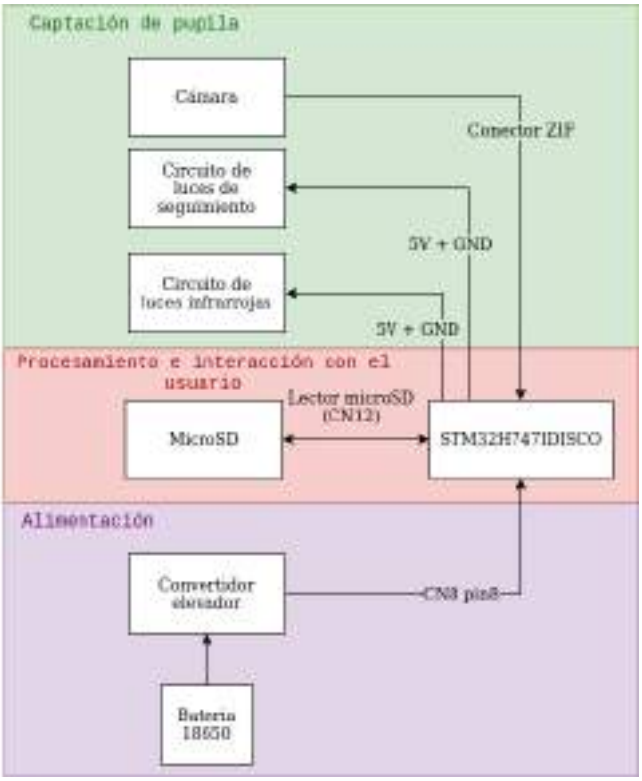


Figura 3.17: Interconexión en diagrama de bloques

Para complementar la figura anterior, la Figura 3.18 agrega una visión de más alto nivel de los componentes del sistema de VNG portátil.

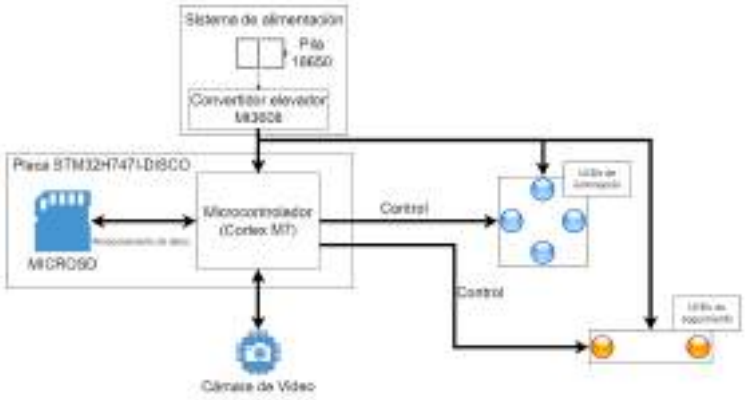


Figura 3.18: Diagrama de bloques del sistema

Para especificar la posición de cada una de las partes anteriormente mencionadas, la Figura 3.19 muestra la parte trasera del casco, señalando las partes relevantes en el entregable final.

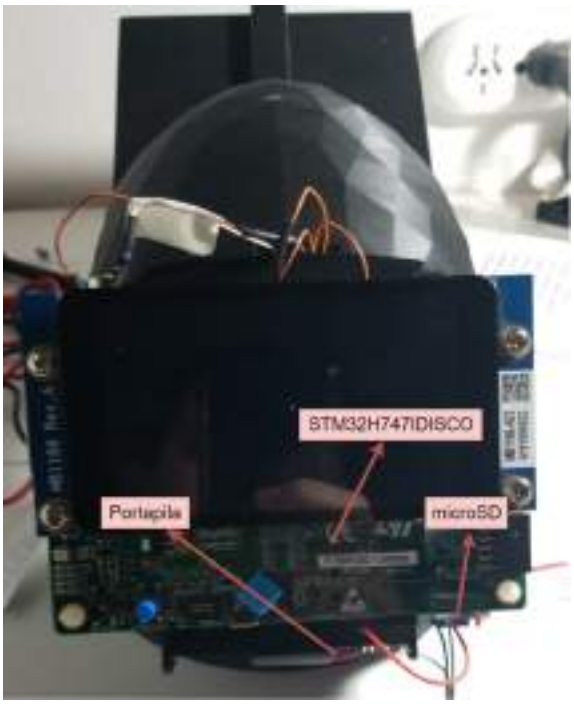


Figura 3.19: Parte trasera del casco

Por ultimo, la Figura 3.20 muestra la parte del casco visible por el usuario durante un estudio.

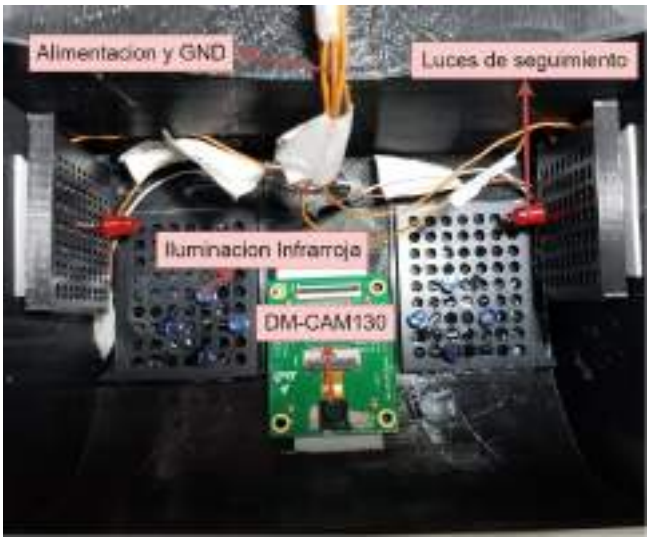


Figura 3.20: Parte delantera del casco

Ambas imágenes omiten el cable que conecta a la placa con el casco, el cual es un cable plano que sigue el mismo curso que el resto.

3.2. Casco

3.2.1. Introducción

El objetivo de este capitulo es describir la solución y metodología utilizada para la construcción del casco. El diseño se puede ver en el documento *Especificación técnica y funcional del casco* (Apéndice A.1.3).

Todas las impresiones se realizaron con la Impresora Creality 3D Ender-3 V2 [21], y diseñadas con el software SketchUp 2017 [22].

3.2.2. Metodología

Existen una gran cantidad de variables involucradas en el desarrollo de un casco, entre ellas:

- Dimensiones
- Forma
- Material
- Peso

- Variables de impresión
- Posición y tamaño del resto de los componentes
- Interconexión de los componentes

Sabiendo que existe una gran incertidumbre a causa de estas variables, se consideró que la mejor metodología para desarrollar esta etapa del proyecto es la de prototipos [3]. Mediante repetición de los ciclos de planificación, diseño y construcción se van mejorando aquellos aspectos que son críticos, haciendo aproximaciones sucesivas y reduciendo la incertidumbre entre iteraciones. La Figura 3.21 muestra las etapas aplicadas al desarrollo del casco.

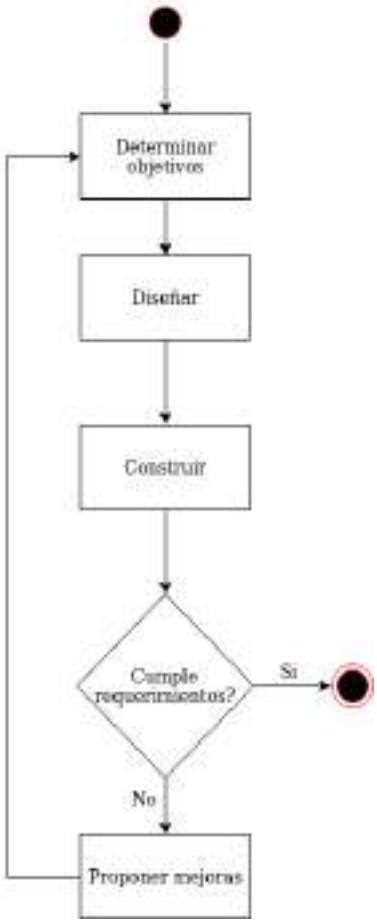


Figura 3.21: Modelo propuesto de la metodología.

Los objetivos y requerimientos se agruparon y se dividieron en etapas, cada una de las cuales se corresponde con un prototipo a realizar.

El Cuadro 3.3 muestra los objetivos por cada etapa.

Etapa	Objetivos básicos
1	Establecer dimensiones, forma general, material y variables de impresión
2	Establecer posición de la placa y peso
3	Establecer posición de los LEDs y cámara

Cuadro 3.3: Tabla de objetivos por etapa.

3.2.3. Desarrollo

3.2.3.1. Primer prototipo

El primer prototipo impreso se diseñó mediante el recorte de un modelo de cabeza 3D (Figura 3.22), con el objetivo de preservar la forma de la misma. Posteriormente, se realizó un escalado para asegurarse de que exista un margen que pueda ser acomodado por algún material compresible o esponjoso, como goma espuma, que permita un ajuste ergonómico de diferentes tamaños de cabezas.

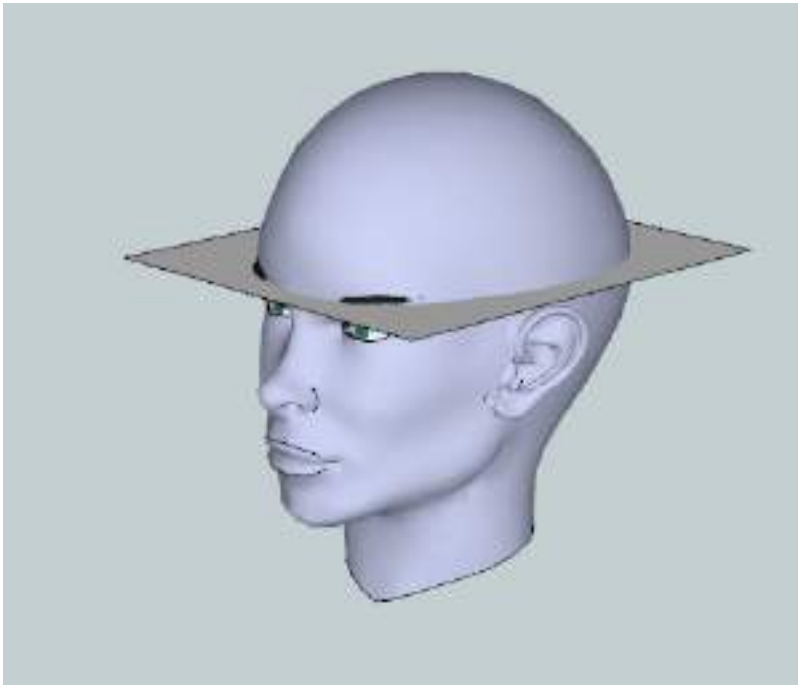


Figura 3.22: Recorte de modelo de cabeza 3D.

Con respecto a la forma, se decidió agregar al primer prototipo:

- Una visera, dado que para la correcta detección es apropiado aislarse de la luz ambiente.
- Dos patillas con agujeros para pasar algún agarre que permita atar el casco a la pera, generando una mayor estabilidad.
- Un soporte en la parte trasera del casco para adherir la placa de desarrollo y otra electrónica necesaria.

El diseño 3D se muestra en la Figura 3.23.

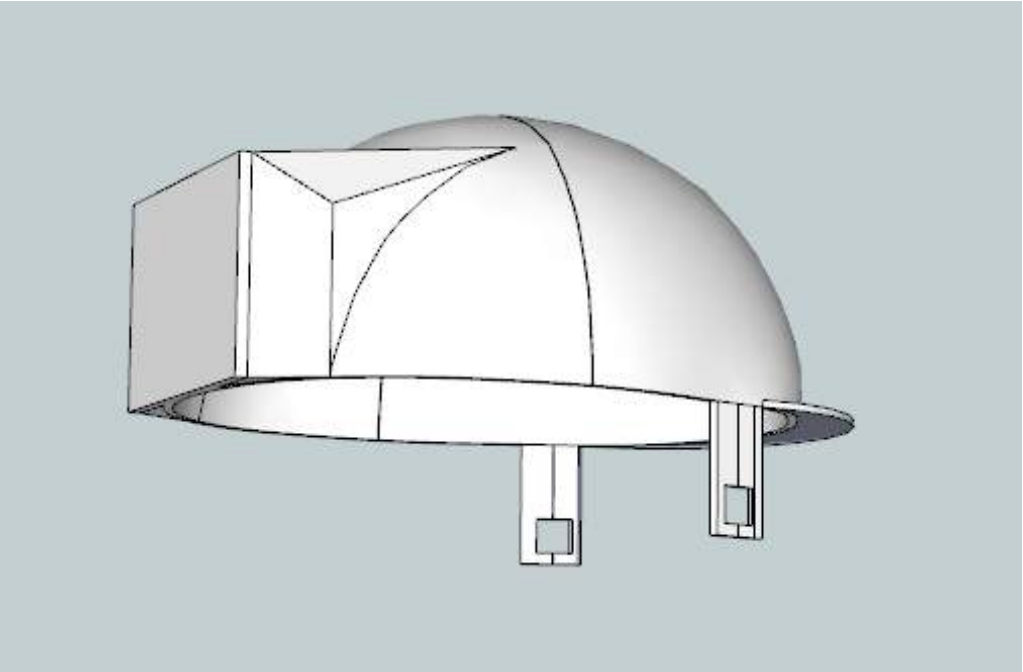


Figura 3.23: Primer prototipo del casco

3.2.3.2. Análisis de aspectos positivos y negativos

Aspectos positivos y negativos del primer prototipo	
Aspectos positivos	Aspectos negativos
La forma del casco es apropiada.	Las dimensiones son incorrectas.
El material es suficientemente robusto.	Las patillas son un punto débil y no cumplen su funcionalidad.
El material es liviano.	La visera no cumple su funcionalidad.
Experiencia de impresión 3D.	La parte trasera desequilibra el casco.
Experiencia de diseño 3D.	

Cuadro 3.4: Aspectos positivos y negativos del primer prototipo.

3.2.3.3. Segundo prototipo

A partir de los aspectos negativos del anterior diseño y teniendo en consideración los objetivos de la segunda etapa, se realizaron las siguientes modificaciones:

- Se corrigieron las dimensiones.
- Se removieron las patillas, dado que no cumplían la funcionalidad para las cuales se habían agregado y además, resultaban una parte débil de la estructura.
- La visera se alargó y se agregó un eje donde pueda encajar una máscara donde se encontraría la cámara y el arreglo lumínico. Además esto permite rediseñar la máscara sin tener que reimprimir todo el casco, mientras el encastre sea el mismo.
- La parte trasera se decidió subir llegando hasta la mitad del casco, de esta manera se posiciona más cerca del centro de masa del casco y genera un momento rotor menor. Además responde a la necesidad de uso del dispositivo, permitiendo al profesional de la salud la realización del estudio mientras el paciente esta sentado y él parado detrás.
- Con el objetivo de reducir el peso, se removieron las partes laterales del casco.

El diseño final de esta etapa se muestra en la Figura 3.24.

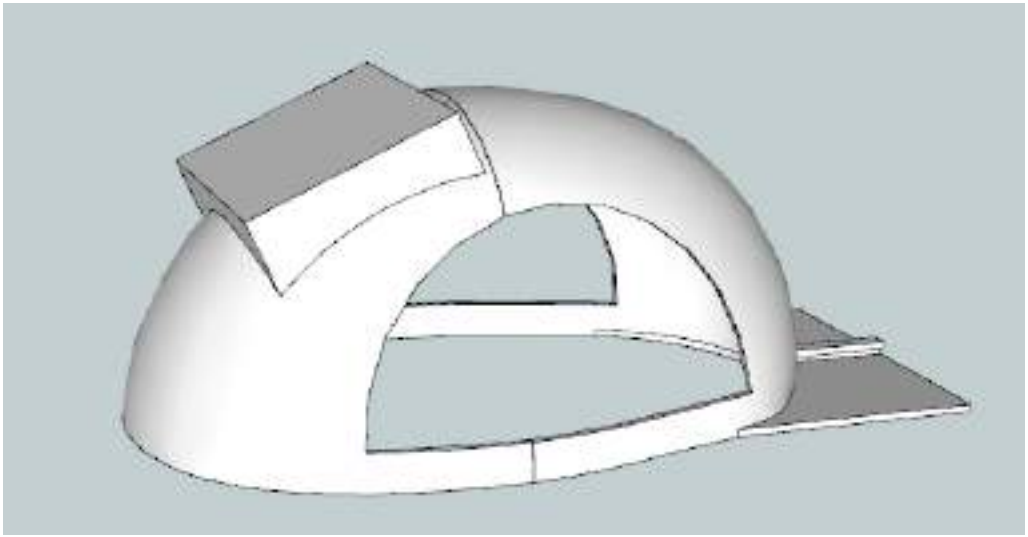


Figura 3.24: Segundo prototipo del casco

3.2.3.4. Análisis de aspectos positivos y negativos

Aspectos positivos y negativos del segundo prototipo	
Aspectos positivos	Aspectos negativos
Las dimensiones son correctas.	Las partes laterales del casco son frágiles
La visera permite intercambiar máscaras de manera simple.	No existe forma sencilla de comunicar la electrónica en la parte trasera con la cámara y arreglo lumínico.
El casco se mantiene equilibrado, sin la electrónica presente.	

Cuadro 3.5: Aspectos positivos y negativos del segundo prototipo.

3.2.3.5. Tercer prototipo

A partir de los aspectos negativos del anterior diseño y teniendo en consideración los objetivos de la tercera etapa:

- Se rellenó nuevamente la parte trasera lateral para mantener una estructura más robusta.

- Se agregó un agujero en la parte superior del casco, a fin de pasar cables que permitan la comunicación con la cámara que se colocaría en la máscara.
- Se diseñó una máscara con el fin de alojar la cámara y el arreglo lumínico encargado de mantener los ojos iluminados durante el estudio. Además, cumple con el objetivo de minimizar la cantidad de ruido lumínico ambiente y sirve como contrapeso a toda la electrónica alojada en la parte trasera del casco.

El primer diseño de esta etapa se muestra en la Figura 4.5.

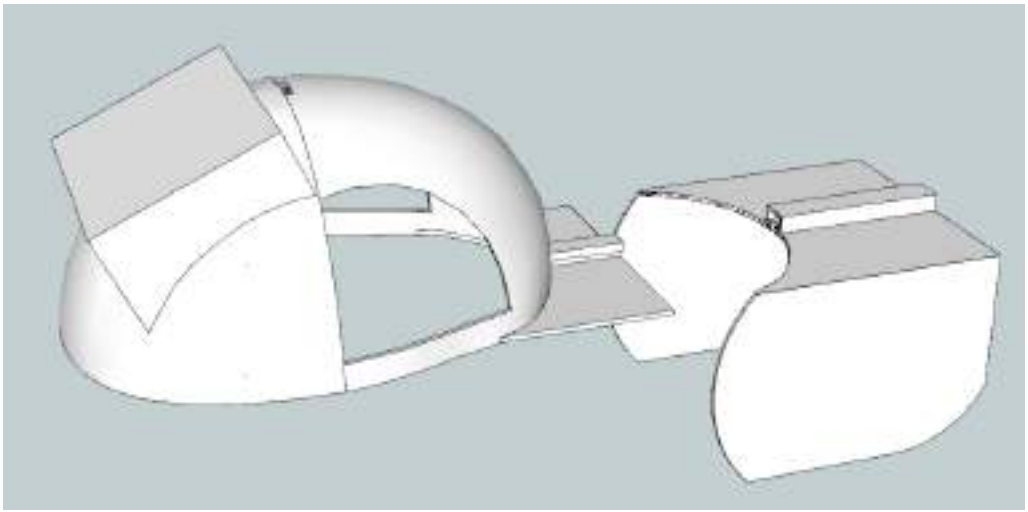


Figura 3.25: Tercer prototipo del casco

3.2.3.6. Análisis de aspectos positivos y negativos

Aspectos positivos y negativos del tercer prototipo	
Aspectos positivos	Aspectos negativos
El casco se mantiene equilibrado con la electrónica presente	Es muy difícil centrar la cámara en la máscara Las partes laterales de la parte frontal del casco son frágiles

Cuadro 3.6: Aspectos positivos y negativos del tercer prototipo.

3.2.3.7. Reimpresión de la máscara

A partir de los aspectos negativos del tercer prototipo se decidió reimprimir la máscara, y dado que el diseño fue pensado de forma modular, se pudo realizar sin tener que reimprimir todo el casco.
Se sumó un eje centrado a la máscara a fin de poder determinar exactamente el centro de la misma. Esto permite posicionar la máscara de manera centrada.
El diseño final de la máscara se muestra en la Figura 3.26.

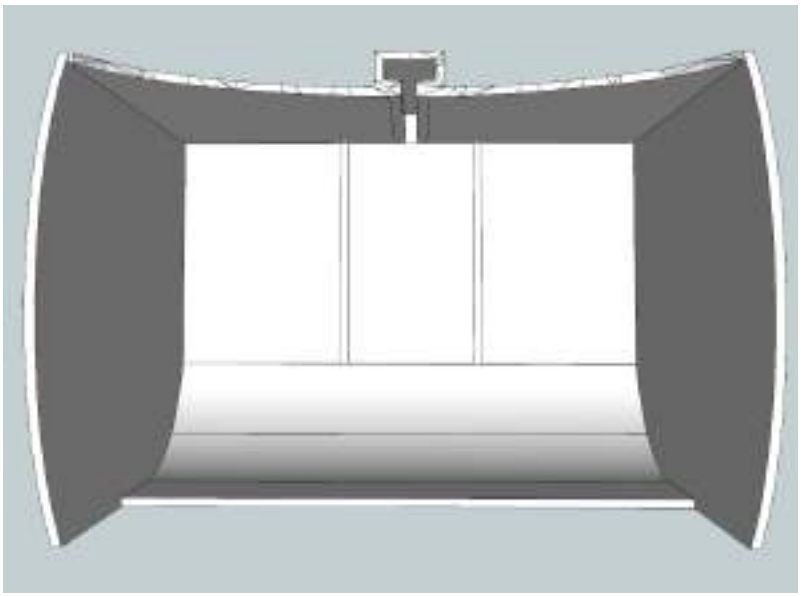


Figura 3.26: Diseño final de la máscara

3.2.3.8. Rejillas de LED

Para estabilizar y homogeneizar la posición de los LED en el casco, se decidió imprimir unas rejillas con el objetivo de poder agregar LED en diferentes posiciones sin tener que reimprimir. La Figura 3.27 muestra el diseño de estas rejillas.

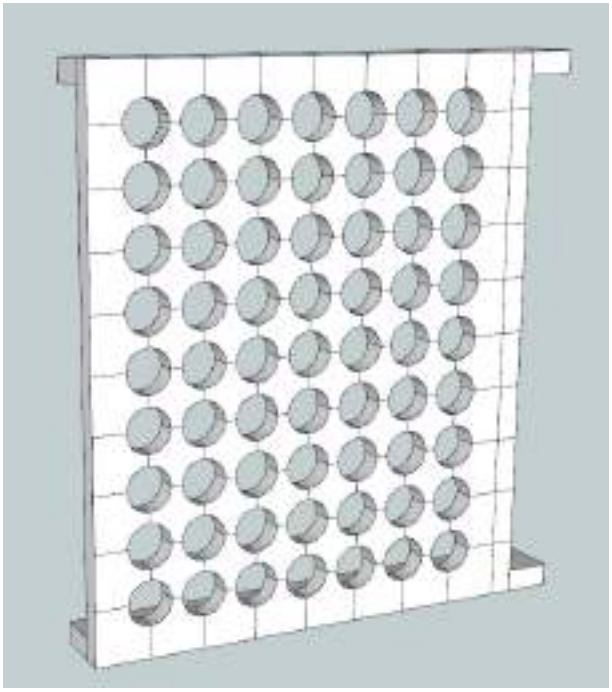


Figura 3.27: Diseño de la rejilla de LED

3.2.4. Características de impresión

El volumen de impresión de la impresora usada es de 220mm x 220mm x 250mm. A partir de esta restricción, el casco se debió imprimir dos partes: una frontal y una trasera, que luego se pegaron utilizando algún pegamento. En el Cuadro 3.7, se mencionan algunas de las características relevantes de configuración de la impresora 3D para la impresión.

Categoría	Valor
Altura de capa	0.25mm
% de rellenado	15
Patrón de rellenado	Cubic
Material	Ácido poliláctico (PLA)

Cuadro 3.7: Configuración impresora 3D.

Respecto de la altura de la capa y el % de rellenado, ambas variables se podrían configurar a fin de obtener un casco más robusto, sacrificando peso.

El patrón de rellenado Cubic se corresponde con rellenar el espacio vacío con cubos 3D apilados e inclinados.

Respecto del material, es destacable que se caracteriza como compostable. Esto quiere decir que es capaz de ser transformado en compost por algunos procesos industriales, minimizando el impacto medioambiental.

3.2.5. Resultado final

En la Figura 3.28 se puede apreciar el resultado obtenido para el casco del sistema VNG luego de las iteraciones y consideraciones detalladas anteriormente.

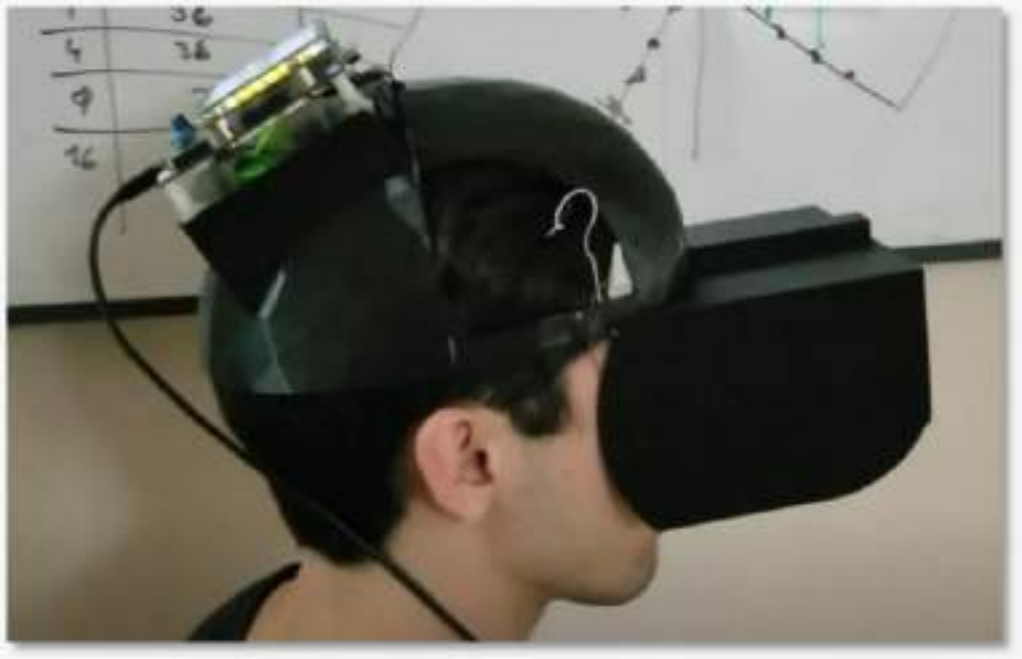


Figura 3.28: Resultado final.

3.3. Software de escritorio

3.3.1. Introducción

El objetivo de esta sección es describir el software de escritorio desarrollado. El diseño se puede ver en la especificación *Especificación de la interfaz de escritorio*, adjunta en el Apéndice A.1.3 del informe.

El objetivo del software de escritorio es permitir la visualización en una plataforma de escritorio de las posiciones de las pupilas en función del tiempo para un estudio en particular, facilitando mayor detalle en la inspección que el que provee la pantalla adosada a la placa de desarrollo.

3.3.2. Dependencias

Uno de los objetivos más importantes de este desarrollo es minimizar la dependencia a otros sistemas. Por este motivo se eligió desarrollar el sistema en el lenguaje de programación Java, permitiendo que el programa se ejecute en una máquina virtual que abstrae al programa del sistema operativo.

Desde la perspectiva del usuario, esto quiere decir que el programa es multi-plataforma, funcionando en un sistema operativo Linux de la misma forma que lo haría en Windows.

Para la realización de los gráficos, se utilizó la librería *JFreeChart* [9]. El resto de la interfaz visual esta basada en componentes de la librería Java Swing.

3.3.3. Resultado

Se muestran las pantallas que deberá atravesar un usuario para visualizar un estudio en particular. Todas las imágenes son tomadas con la aplicación funcionando en un sistema operativo Windows 10. La Figura 3.29 muestra la pantalla que se muestra al abrir el programa.

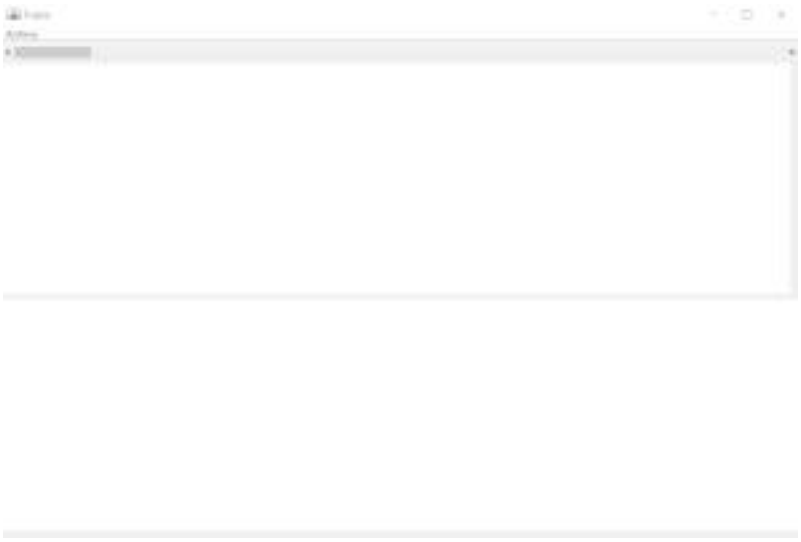


Figura 3.29: Primer pantalla del software de escritorio

Al presionar el botón *Archivo* y seleccionar la opción *Cargar estudio* se abre una pantalla de navegador de archivos que permite seleccionar el archivo almacenado en la tarjeta Micro SD por el sistema embebido. Esta situación se muestra en la Figura 3.30.

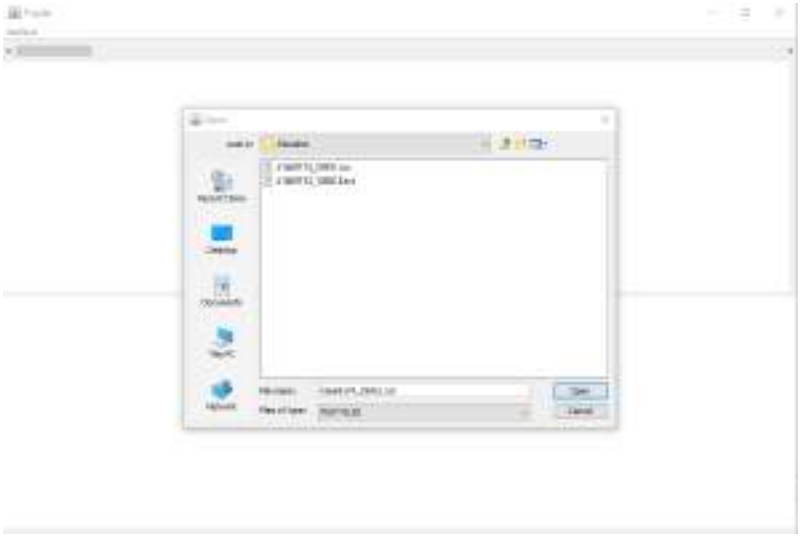


Figura 3.30: Segunda pantalla del software de escritorio

Al seleccionar el archivo de texto que se quiere visualizar y presionar el botón *Abrir*, se muestran los gráficos correspondientes al estudio que ese archivo identifica.

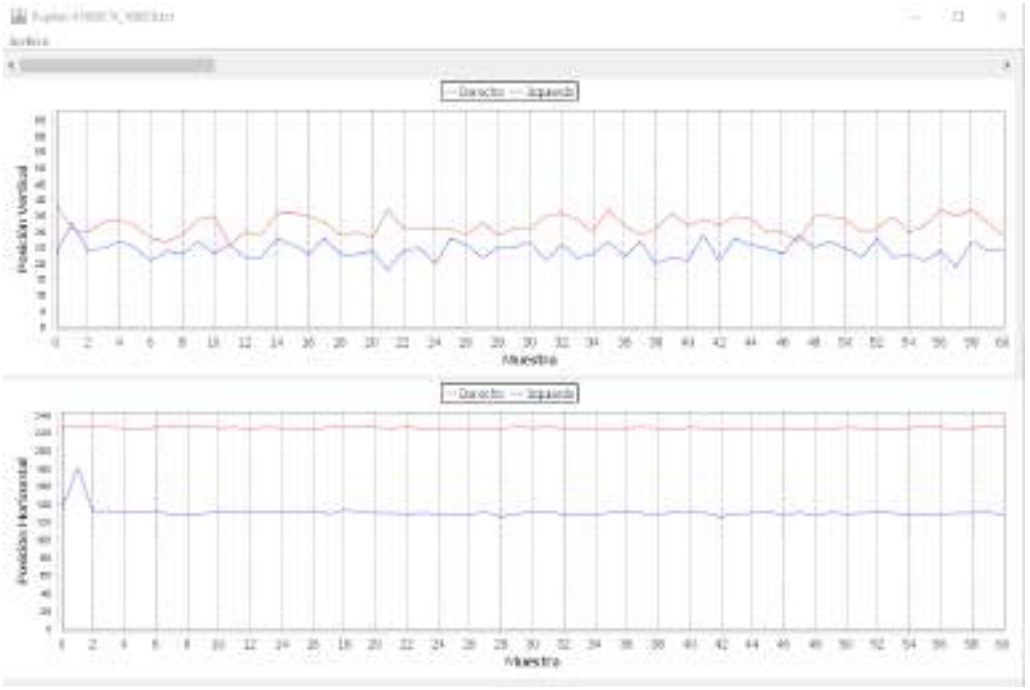


Figura 3.31: Tercera pantalla del software de escritorio

La Figura 3.31 muestra el resultado de un análisis particular. Se puede navegar libremente sobre el estudio, independientemente de la cantidad de pares de puntos en el plano que el mismo posea.

Para la navegación se puede utilizar la barra lateral que se encuentra en la parte superior, o la rueda del ratón en caso de contar con una.

Si el análisis del estudio terminó, se puede cerrar el programa o cargar otro estudio seleccionando la opción de *Archivo*.

Capítulo 4

Conclusiones

4.1. Estudio de mercado

En la subsección 2.1 se mostraron algunas de las características de las soluciones presentes en el mercado. El Cuadro 4.1 incorpora las características de la solución realizada.

Característica	2D-VOGFW	BG4.0USB	USBM2.1A	PupilAr
Resolución	640 x 480	320 x 240	640 x 480	640 x 80
Cámaras	2	2	1 (móvil)	1
FPS máximo	100	100	50	30[1]
Peso(gr)	305	345	-	500

Cuadro 4.1: Tabla de objetivos por etapa.

En comparación a todos estos modelos, la ventaja es la portabilidad y la independencia de sistemas de cómputo y de la red eléctrica.

Respecto del software de análisis de la posición de pupila, el algoritmo desarrollado es capaz de entregar rendimientos superiores en un hardware de menores prestaciones que el especificado para los equipos mencionados en el Cuadro 4.1.

4.2. Aspectos mejorables

Si bien se cumplieron los requerimientos estipulados, se detallan algunos aspectos mejorables para el futuro del producto:

[1] Limitación propia de la cámara y no del algoritmo de cómputo

- Calibración: la calibración de la posición de la pupila con respecto de la cámara se hace solo mediante el movimiento del casco en la cabeza del paciente, lo cual es muy limitado. Para solucionar este límite se debería pensar en un rediseño del casco que permita cambiar la posición de la cámara respecto a los ojos.

- Iluminación: la iluminación es un aspecto importante para la correcta detección. Mejorando la calibración, la simetría del casco y la disposición de los LED se podría homogeneizar aún más.

- Cámaras: uno de los desafíos del diseño es que la cámara esté lo más cerca posible del ojo para maximizar el rango de posiciones posibles de la pupila, pero a la vez esté lo suficientemente lejos como para permitir que ambas sean visibles. Esta relación de compromiso se elimina completamente si se utilizan dos cámaras o si se utiliza un arreglo de espejos que logre un efecto similar.

- Utilizar el segundo microcontrolador: la falta de documentación respecto a la implementación de la interfaz gráfica en el microcontrolador CM4 hizo imposible su utilización en el proyecto. Sin embargo, la potencial utilización de dos procesadores es una de las ventajas más importantes de la placa STM32H747I-DISCO. Se podría investigar y desarrollar una interfaz que esté alojada en el microcontrolador CM4 logrando que la misma sea responsiva incluso cuando se está realizando un estudio.

- Diseño y ergonomía del casco: el alcance de este proyecto no cubría un desarrollo profundo de estos aspectos, ya que estas características no son incumbencias profesionales de los proyectistas. Quedaría en manos de un especialista en dicha área realizar las mejoras necesarias.

4.3. Plan ejecutado y proyectado

Las Figuras 4.1 y 4.2 muestran el diagrama de Gantt ejecutado. La planificación inicial del proyecto, realizada en junio de 2021, se puede ver en el documento *Plan de Proyecto* (Apéndice A.1.1). La misma planeaba la finalización del proyecto para abril de 2022, pero se ha visto alterada por diferentes causas que se detallan a continuación.

Figura 4.1: Diagrama de Gantt ejecutado (parte 1)

Figura 4.2: Diagrama de Gantt ejecutado (parte 2)

4.3.1. Retraso en la obtención de la placa de desarrollo

Se esperaba que la placa de desarrollo estuviera disponible para ser utilizada el día 13 de julio de 2021, pero debido a problemas de movilidad relacionados con la pandemia por COVID-19, terminó siendo recibida el día 22 de agosto.

Debido a esto, fue necesario realizar un reordenamiento de tareas, dado se había planeado comenzar con la implementación del firmware el 13 de julio de 2021. De esta forma, se pospuso el comienzo de esta tarea para el día 13 de septiembre y se comenzó con el diseño del casco el día 15 de julio, por lo que la fecha de finalización del proyecto no se vio modificada.

4.3.2. Escasez de documentación de tecnologías utilizadas

El diseño inicial del equipo de videonistagmografía contemplaba la implementación del manejo de la interfaz de usuario y otras tareas menores

en el procesador CM4, y las tareas de alta demanda de recursos, como la detección de pupilas, en el procesador CM7.

En pos de cumplir con este diseño, se intentó infructuosamente, durante un mes, implementar la interfaz de usuario en el procesador CM4. Para la resolución del problema se efectuaron diversas consultas al fabricante de la placa, ante las cuales se obtuvieron respuestas poco alentadoras, indicando que actualmente no contaban con el conocimiento necesario para realizar este tipo de implementación.

Debido a esto, se desistió de implementar este diseño y toda la funcionalidad del sistema fue construida en el procesador CM7.

Este inconveniente provocó un retraso de aproximadamente un mes.

4.3.3. Daño de la placa de desarrollo

Durante la depuración de la medición de voltaje de la batería se cometió el error de conectar el terminal positivo de la misma al pin A5 del conector CN8, cuando se tenía la intención de hacerlo en el pin A4 del CN8, que estaba configurado como ADC. Esto provocó el apagado repentino de la placa y la terminación abrupta de la sesión de depuración.

El pin al que fue conectada la batería estaba configurado como FMC, que es una interfaz que permite interactuar con la memoria flash NOR de la placa, entre otros periféricos.

El primer paso para identificar el problema fue tratar de determinar qué partes del sistema continuaban funcionales. Se trató de cargar el código nuevamente a la placa, ante lo cual se obtuvieron diversos mensajes de error indicando que ciertas partes de la memoria del sistema se encontraban corruptas.

Luego, con la herramienta STLink Utility, se intentó leer los registros del procesador, lo cual se logró exitosamente, pudiendo comprobar que el CPU CM7 del microcontrolador continuaba funcional.

Posteriormente, se analizaron las temperaturas de la placa con una cámara termográfica, pudiendo determinar que el circuito integrado USB3320C-EZK (que es parte del sistema de USB-OTG de la placa y no estaba siendo utilizado) registraba temperaturas mayores a los 130°C. Ante una inspección más profunda de la placa, se observó que el estaño que unía los pines de este componente al PCB presentaba burbujas debido a las altas temperaturas, por lo que se procedió a desoldar el componente del sistema.

Luego, se analizaron las memorias flash NOR. El sistema cuenta con dos circuitos integrados de este tipo, cada uno de 64MB. El objetivo fue tratar de determinar si ambos de estos integrados se habían averiado, o si alguno continuaba funcional. Se midieron las señales que llegaban a cada

entrada de las memorias, y todas resultaron correctas. Luego se probó la placa con un solo integrado, con los dos y, por último, con ninguno. Esta última prueba fue la única exitosa, pero no presentaba una escenario favorable para este proyecto, ya que se necesitaba al menos una de estas memorias para almacenar el código, dado que la memoria interna del microcontrolador es de 2MB y el binario a ser utilizado ocupaba un espacio superior a los 5MB.

Ante las pruebas realizadas, se llegó a la conclusión de que la entrada de tensión de 5,2V por el pin A5 del conector CN8 actuó como una fuente de tensión conectada en paralelo a la fuente de tensión de 5V que alimentaba la placa, generando intensidades de corrientes superiores a las que son capaces de soportar algunos circuitos integrados de la STM32H747I-DISCO.

Ante la imposibilidad de comprar las memorias flash NOR averiadas por falta de stock, se procedió a tratar de adquirir otra placa igual a la utilizada, lo cual fue imposible por las mismas razones.

Luego, se analizó la posibilidad de comprar una placa alternativa, que debería cumplir los siguientes requisitos:

- Tener conector para cámara
- Tener conector para pantalla
- Contar con una frecuencia de reloj de al menos 400MHz
- Contar con una memoria RAM de al menos 1MB

Además, la placa debía ser del fabricante STMicroelectronics, dado que para este proyecto se utilizó una herramienta de desarrollo de interfaces gráficas propietario de esta marca.

Como se puede ver en el Cuadro 4.2, ninguna placa que se encuentre en stock en los diferentes distribuidores de componentes electrónicos internacionales cumplía con todos estos requisitos.

Placa	Display	Cámara	Clock (MHz)	RAM (KB)
STM32F746GDISCO	Si	Si	216	340
STM32L496GDISCO	Si	Si	80	340
B-U585I-IOT02A	No	Si	160	786
P-L496G-CELL02	No	Si	80	320
STM32F7508-DK	Si	Si	216	340
STM32H750B-DK	Si	No	480	1024
STM32H745I-DISCO	Si	No	480	1024
STM32F769I-DISCO	Si	Si	216	532

Cuadro 4.2: Alternativas de placas.

El último enfoque para la solución del problema fue analizar la viabilidad de compra de una placa que tenga una frecuencia de reloj lo suficientemente rápida para cumplir los requerimientos del proyecto, y luego modificarla para agregarle las características faltantes, ya sea el conector de pantalla, el conector de cámara o una memoria RAM externa.

Para el análisis del mínimo clock necesario, se asumió un comportamiento lineal de la velocidad del algoritmo de detección de pupila con respecto a la frecuencia del reloj del sistema. Se determinó como requerimiento lograr una velocidad superior a los 20FPS, establecido en el documento *Especificación de Requerimientos* de este proyecto. Se conoce que el algoritmo es capaz de funcionar a 36FPS con un clock de 400MHz, por lo que una regla de tres simple indica que la velocidad del reloj del sistema debería ser mayor a los 222,22MHz.

Placa	Encapsulado
STM32F746GDISCO	BGA216
STM32L496GDISCO	UFBGA169
B-U585I-IOT02A	UFBGA169
P-L496G-CELL02	UBGA169
STM32F7508-DK	BGA216
STM32H750B-DK	TFBGA240
STM32H745I-DISCO	TFBGA240
STM32F769I-DISCO	BGA216

Cuadro 4.3: Encapsulados de microcontroladores en las placas alternativas.

Como se puede ver en el Cuadro 4.2, las únicas placas que cumplen este requisito son la STM32H750B-DK y la STM32H745I-DISCO, pero no se cuenta con el conocimiento ni las herramientas necesarias para modificar una placa con un microcontrolador con encapsulado TFBGA240.

Este inconveniente provocó un retraso de aproximadamente 25 días.

4.3.4. Análisis de bitácora

Para un análisis detallado de las horas demandadas por el proyecto se implementó un seguimiento mediante una bitácora, donde se especificaron las tareas realizadas y las horas dedicadas a cada una de ellas.

Las horas invertidas en el proyecto se dividieron en 8 categorías, detalladas a continuación:

- Tareas de seguimiento: consiste en las reuniones llevadas a cabo tanto con los directores del proyecto como con la cátedra de trabajo final.

- Documentación: refiere a las horas invertidas en documentar el desarrollo del proyecto, tanto en las diferentes especificaciones como en la redacción del presente informe.
- Estudio preliminar: estudio de la problemática y del mercado al cual pertenece el prototipo construido.
- Requerimientos: horas invertidas en sentar los requerimientos del prototipo, tal que este sea capaz de solucionar la problemática establecida.
- Diseño: diseño del casco, de la interfaz de escritorio y del firmware implementado.
- Construcción: horas invertidas en la construcción del prototipo construido.
- Pruebas: pruebas realizadas sobre el prototipo construido.
- Gestión: horas invertidas en la confección del plan de proyecto, diagramas de Gantt, etc.

En la Figura 4.3 se pueden ver las horas invertidas en cada una de las categorías detalladas anteriormente. En la misma se puede observar que la categoría que más horas demandó fue la de *construcción del prototipo*, con 364 horas, por sobre el diseño del mismo, que demandó 242 horas. El escenario ideal indica que se deben invertir más horas en el diseño que en la construcción, pero en este caso esta última conllevó más horas que las planeadas debido a la inexperiencia en las tecnologías utilizadas y a la falta de documentación de las mismas.

También es importante notar la baja cantidad de horas dedicadas a las pruebas del prototipo. Esto se debe principalmente a la ruptura de la placa durante la realización de las mismas.

Es interesante notar la cantidad de horas invertidas en documentación, que se deben a que se consideró especialmente relevante generar información precisa y detallada sobre cada una de las etapas del proyecto.

El proyecto demandó, finalmente, 855 horas.

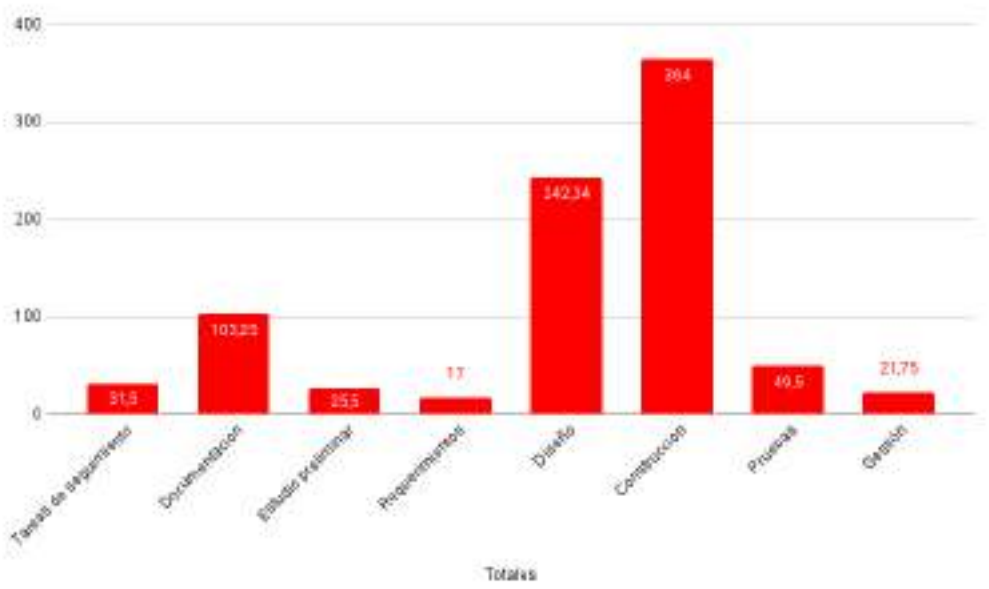


Figura 4.3: Bitácora

4.4. Experiencias y aprendizajes

En general, la experiencia del proyecto fue muy positiva, siendo extremadamente útil enfrentarse a un proyecto de estas magnitudes en las instancias finales de la carrera.

Se entiende que el prototipo implementado es mejorable en diversos aspectos, pero en general se considera que se aportó valor a cada una de las partes sobre las que se trabajó, lo que deja a los proyectistas conformes.

Se logró profundizar mucho los conceptos de trabajo en sistemas embebidos y sistemas operativos, y se identificaron algunas de las falencias que son comunes en el campo, en especial la falta de documentación sobre algunos aspectos.

Se obtuvo una primera aproximación al área de diseño 3D e impresión 3D, lo cual si bien no compete a la carrera de Ingeniería en Computación, se considera que es un valor agregado a la formación personal y profesional.

Por último, respecto al aspecto organizacional, se comprendió el conjunto de retrasos que afectaron al proyecto y se tomó a conciencia la causa de los mismos y qué se podría haber hecho mejor.

Bibliografía

[1] C. Ordoñez, E. Blotta, J. Pastore; "Isophote-Based Low-Computing-Power Eye-Detection Embedded-System". IEEE Latin America Transactions, Vol. 18, No.2, February 2020.

[2] URL: https://medlineplus.gov/spanish/ency/article/003037.htm#:~:text=Es%20un%20t%C3%A9rmino%20para %20describir,Arriba%20y%20abajo%20(nistagmo%20vertical)

[3] Sommerville, Ian. Ingeniería del software. Pearson educación, 2005.

[4] URL: https://www.interacoustics.com/download/visualeyes

[5] URL: https://www.difra.be/shop/00022134-di-140501-nysstar-ii-vng-binocular-system-with-goggles-hardware-only-with-usb-cable-4-5-m-for-connection-to-computer-19023?search=vng#attr=

[6] URL: https://dizziness-and-balance.com/research/equipment.htm

[7] URL: https://store.medtrakvng.com/MedTrak-VNG-Testing-Equipment_p_3

[8] URL: https://www.st.com/resource/en/datasheet/stm32h747ag.pdf

[9] URL: https://www.jfree.org/jfreechart/

[10] URL: https://www.freertos.org/

[11] URL: https://www.st.com/en/development-tools/touchgfxdesigner.html

[12] URL: https://www.st.com/content/ccc/resource/training/technical/product_training/group0/90/5a/91/24/1a/14/4b/40/STM32F7_Peripheral_DCMI/files/STM32F7_Peripheral_DCMI.pdf/jcr:content/translations/en.STM32F7_Peripheral_DCMI.pdf

[13] URL: https://www.st.com/resource/en/application_note/an5020-digital-camera-interface-dcmi-on-stm32-mcus-stmicroelectronics.pdf

[14] URL: https://dl.btc.pl/kamami_wa/dm-cam130_um.pdf

[15] URL: https://www.waveshare.com/w/upload/0/0a/OV9655.pdf

[16] URL: http://www-edlab.cs.umass.edu/~smaji/cmpsci370/slides/hh/lec02_hh_advanced_edges.pdf

[17] URL: https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

[18] URL: https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0029740&type=printable

[19] URL: https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf

[20] URL: https://www.candeltec.es/seguridad-fotobiologica-norma-iec-62471

[21] URL: https://www.creality.com/products/ender-3-v2-3d-printer-csco)

[22] URL: https://www.sketchup.com/es)

[23] Kervrann, C., Hoebeke, M. & Trubuil, Isophotes Selection and Reaction-Diffusion Model for Object Boundaries Estimation, A. International Journal of Computer Vision (2002) 50: 63. https://doi.org/10.1023/A:1020276319925.

[24] J. Lichtenauer, E. Hendriks and M. Reinders, Ïsophote properties as features for object detection,"2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 649-654 vol. 2. doi: 10.1109/CVPR.2005.196

[25] P.W. Verbeek, A class of sampling-error free measures in oversampled band-limited images, Pattern Recogn Lett, 3 (4) (1985), pp. 287-292.

Apéndice A

Apéndices

A.1. Documentos anexos

A.1.1. Plan de proyecto

URL: https://docs.google.com/document/d/e/2PACX-1vQ0mrwowNHd2XD04TsALll_kJuZEWQeFhy4ArUqasyJrQC0LKElYDmud46DK3d0VkdyEnd3KgPPw1DJ/pub

A.1.2. Especificación de requerimientos

URL: https://docs.google.com/document/d/e/2PACX-1vSjvkY5L 4BDmNYVnR05q463I-i54T8w465ncmkkaRJ2QfnuJFcSC_2GiAlx_nqvg
A3zUO9yJqsq3zIc/pub

A.1.3. Especificaciones funcionales

A.1.3.1. Especificación funcional de la interfaz

URL: https://docs.google.com/document/d/e/2PACX-1vQ8Cn3o8 visqGW3nD5zokCnlbvvgl44mJnNx4GpYx/pub

A.1.3.2. Especificación funcional y técnica del casco

URL: https://docs.google.com/document/d/e/2PACX-1vQrM_Tz 6yg-wirXV1bmjnD4v5aYPWYglyfh7K2I3NJmoUUTCzjmvKYb2ngZsE5ou3 rP2bbWGtZeJ_EGg/pu

A.1.3.3. Especificación de la interfaz de escritorio

URL: https://docs.google.com/document/d/e/2PACX-1vTWebvv zh25y_x0SVwXMR8OPN
h8AzS80dtDS7U6QaM/pub

A.1.4. Especificaciones técnicas

A.1.4.1. Especificación tecnica del firmware

URL: https://docs.google.com/document/d/e/2PACX-1vR_al 2JTiY-YWjbt-33TCEEtWbxrIh8tbV8VikQ5Tu7vZCsl5u354zC03waU sidHilual9rAQOXq3Ze/pub