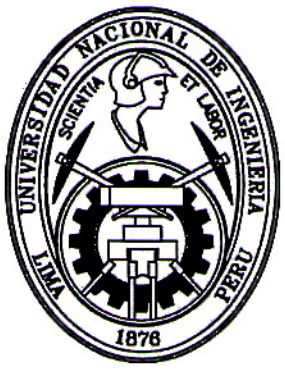
UNIVERSIDAD NACIONAL DE INGENIERÍA

Facultad de Ciencias

Escuela Profesional de Ingeniería Física



TESIS

Para optar el Título Profesional de:

INGENIERO FÍSICO

Titulada:

Desarrollo de un equipo automático de lectura de placas MODS para el diagnóstico rápido de tuberculosis y determinación de susceptibilidad a drogas

Presentada por:

Jorge Daniel Mendoza Ramírez

Asesor:

Mg. Germán Yuri Comina Bellido

LIMA – PERÚ

2011

RESUMEN

En esta tesis se expone el diseño, construcción e implementación de un instrumento óptico automatizado que permite aplicar el método MODS (Microscopic Observation Drug Susceptibility Assay) para el diagnóstico de tuberculosis con alta certeza de resultados en un tiempo óptimo. Este equipo de lectura de placas será denominado en adelante MODS Plate Reader.

En base al diseño del mencionado equipo y al software desarrollado, se redactó un procedimiento experimental detallado para la correcta utilización del mismo a nivel usuario.

Se presentan diagramas y fotografías del lector de placas construido así como los algoritmos, su explicación en diagrama de bloques, y segmentos de códigos fuente de los programas principales que automatizan los procesos de posicionamiento y lectura. Adicionalmente, se desarrolla una breve explicación del funcionamiento del algoritmo de detección de agentes de tuberculosis. En el reporte, se comparan resultados positivos y negativos de muestras analizadas con el algoritmo de detección.

2. Términos relacionados con MODS Plate Reader

MODS Plate Reader. Equipo automatizado de lectura de placas MODS.

Stepper Motors. Motores digitales, base del sistema de posicionamiento [12]. Permiten control en lazo abierto y presentan un comportamiento determinístico.

Tarjeta de adquisición de datos. Interfaz de la computadora con las señales del mundo externo [13]. Contiene elementos hardware tales como: multiplexores, amplificadores, filtros, ADC y DAC necesarios para el proceso de toma de datos. El usuario puede tener acceso a los datos almacenados temporalmente en la tarjeta por medio de un microcontrolador y un bus de comunicaciones. Adicionalmente al hardware, se requiere software tipo driver para realizar las tareas de lectura y escritura a bajo nivel, dejando al usuario una API estándar para aplicaciones de desarrollo en la forma de librerías.

H-Bridge (Puente H). Circuito de amplificación para manejar los motores a pasos [12].

FFT. Siglas para Fast Fourier Transform [11]. Es un algoritmo eficiente para calcular la transformada de Fourier discreta. FFT resulta de gran importancia en una amplia gama de aplicaciones relacionadas con procesamiento digital de señales.

ActiveX. Estructura o marco que define componentes de software reusable que realizan funciones particulares en Microsoft Windows [14], de tal manera que sean independientes del lenguaje de programación con el cual fueron creadas.

MFC. Siglas para Microsoft Foundation Classes [14]. Estas clases en C++ están definidas para muchos objetos manejados por Windows así como ventanas y controles comunes. Se emplean en aplicaciones de tipo interfaz gráfica con programación manejada por eventos.

Máquina virtual (Virtualización). Implementación en software de una máquina que ejecuta programas como si fuera una máquina física [15].

Microscopio invertido. Un microscopio invertido tiene su fuente de luz sobre la muestra, mientras las lentes objetivo se encuentran por debajo. Este tipo de microscopios es útil para observar células vivientes u organismos que se encuentran al fondo de un contenedor [11].

Gradiente de Brenner. Cantidad empleada para cuantificar el contraste de una imagen [10]. Cualitativamente, se espera que cuando una imagen se encuentre mejor enfocada; esto es, con bordes mejor definidos, se obtenga un gradiente de Brenner más alto.

0.6. Definición del mensurando

El método MODS se basa en la detección microscópica de la aparición y crecimiento de organismos en forma de cordones cilíndricos, los cuales son característicos de la bacteria MTB. Por tanto, el procedimiento experimental consta de una rutina de adquisición de imágenes de muestras de esputo de pacientes con la consecuente aplicación de un algoritmo de reconocimiento de TB a las imágenes. La detección se basa en el crecimiento característico de los agentes MTB, así como en su forma y manera de refractar la luz, fácilmente reconocible por medios ópticos. Las muestras de esputo con los reactivos para el análisis se colocan en placas de cultivo de 24 pozos a una temperatura de 37 °C, en laboratorio con nivel de bioseguridad P2.

0.7. Suposiciones y limitaciones

El análisis efectuado a las muestras es estrictamente visual, de donde devienen algunas de las principales limitaciones, entre las cuales podemos citar:

- El reconocimiento sólo puede efectuarse a muestras cultivadas entre 9-11 días. Antes de los 9 días los cordones característicos aún son muy pequeños para que el software los pueda identificar correctamente. Por otro lado, a medida que pasa el tiempo luego de los 11 días, la tendencia de las bacterias es a formar conglomerados que hacen imposible la detección de cordones individuales.

- Las bacterias tienden a crecer en el fondo de los pozos de las placas en una región con una altura de algunas micras de espesor. El enfoque automático en este caso se ve limitado por este hecho, debido a que no se trata de un plano muy bien definido, debiendo limitar el resultado a la mejor posición de enfoque posible. Adicionalmente, el plano en el que se encuentran los cordones varía de un pozo al siguiente, con lo cual la labor de enfoque resulta relativamente compleja (revisar apartado 2.2.5 sobre enfoque automático).

- Las muestras se encuentran en medio líquido, de donde se debe tener especial cuidado al manipular la placa MODS. Se debe buscar minimizar el movimiento de vibración de las placas durante el proceso de lectura.

Índice General

Introducción

0.1. Antecedentes

Cada año en el mundo se produce un estimado de ocho millones de nuevos casos de tuberculosis clínica y tres millones de muertes debido a esta enfermedad. De acuerdo con el Ministerio de Salud, se han registrado más de 33000 casos de tuberculosis a la fecha en nuestro país, principalmente personas en situación de pobreza y pobreza extrema. El Perú es considerado actualmente el país del continente con mayor incidencia de tuberculosis y tuberculosis resistente a drogas. Por otro lado, el incremento en el número de cepas multidrogo resistentes y la pandemia del VIH amenazan seriamente los esfuerzos por controlar esta enfermedad a nivel mundial.

El método de diagnóstico de tuberculosis empleado en la actualidad en nuestro país no es óptimo y puede tomar varias semanas hasta confirmar la presencia de la enfermedad. Actualmente en el Perú, un paciente con tuberculosis multidrogo resistente no cuenta con dicho diagnóstico a tiempo e inicia un tratamiento empírico por lo menos 9-12 meses después de haber sido diagnosticado, con lo cual, no sólo no se consigue un tratamiento eficaz contra la enfermedad, sino que de hecho se reducen las probabilidades de curación al mínimo y se incrementa el riesgo de transmisión de la enfermedad considerablemente.

0.2. Declaración del problema

La incapacidad de poder realizar un diagnóstico y una prueba de susceptibilidad temprana para Mycobacterium tuberculosis, impide el inicio de un tratamiento adecuado temprano. Esta limitación incrementa el tiempo de la enfermedad en su fase infecciosa aumentando la tasa de transmisión de tuberculosis con contagios a la población sana susceptible. Adicionalmente, la prolongación de la enfermedad está asociada a una reducción de la tasa de curación y por lo tanto a un incremento de la mortalidad.

La limitación de no tener un diagnóstico y prueba de susceptibilidad rápida se refleja en un impacto económico negativo, ya que la morbilidad causada por tuberculosis está asociada a una reducción de la capacidad económico-productiva de la población enferma, lo cual a su vez se ve traducido en un deterioro económico familiar progresivo. Adicionalmente, el problema de salud pública de la tuberculosis genera un costo económico importante para el Estado que debe cubrir los tratamientos y cuidados para los pacientes y convalecientes de esta enfermedad. Una dificultad adicional para el control de la tuberculosis radica en que esta enfermedad está fuertemente asociada a la pobreza. Indicadores de pobreza como desnutrición, insalubridad, adicción a drogas, entre otros, constituyen factores de riesgo para contraer tuberculosis.

Dado el impacto económico causado por la enfermedad, la tuberculosis misma causa o agudiza la pobreza, generándose un círculo vicioso y un entrampamiento en el que caen tanto los pacientes como sus familiares más cercanos. La tuberculosis es una enfermedad estigmatizante, así como causa común de marginación, por lo que presenta un impacto social negativo, tanto sobre la población enferma como sobre los familiares en mayor riesgo de enfermarse. La falta de una capacidad de detección temprana y el inicio de un tratamiento personalizado adecuado, incrementa el tiempo con enfermedad y con ello el estigma social causado.

Hace algunos años, se descubrió en los Laboratorios de Investigación y Desarrollo de la Universidad Peruana Cayetano Heredia, un método para diagnosticar tuberculosis y determinar susceptibilidad a drogas en un promedio de 7 días, a un costo muy bajo: el método MODS. MODS ha demostrado ser una técnica rápida y económica con una alta sensibilidad y especificidad. El uso de MODS actualmente alcanza a laboratorios en China, India, Indonesia, Malaysia, Tailandia, Vietnam, Sudáfrica, Uganda, Zimbabwe, Etiopía, Malawi, Ecuador, Bolivia, Venezuela, Brasil, entre otros. En el caso peruano, MODS está terminando de implementarse en el laboratorio de referencia nacional de micobacterias del Instituto Nacional de Salud así como en otros laboratorios de referencia regionales. En la actualidad, este método es reconocido y utilizado como norma técnica en la política de control de tuberculosis y TBC multidrogo resistente en nuestro país. La integridad de países en el mundo que sufren la problemática de la tuberculosis está al pendiente observando al Perú para aprender a utilizar la prueba MODS en sus programas de control de tuberculosis.

La razón por la cual el método MODS aún no puede ser implementado en la mayoría de los laboratorios microbiológicos del Ministerio de Salud y del Programa Nacional de Control de Tuberculosis, se debe al mismo principio básico del funcionamiento del método MODS: El MODS se basa en el reconocimiento visual de patrones morfológicos que son característicos y específicos de Mycobacterium tuberculosis.

El reconocimiento se realiza a través de la observación directa de cultivos de bacterias a 7 días utilizando un microscopio invertido. Debido a la limitación de recursos económicos, estos laboratorios no cuentan con un microscopio invertido, el cual además de ser costoso (aproximadamente US$ 6,000), no es parte de los equipos considerados como convencionales y por lo tanto su uso no se extiende a otras pruebas de interés. Adicionalmente, el reconocimiento de los patrones debe ser realizado por personal técnico calificado con entrenamiento en MODS. De hecho, la alta sensibilidad y especificidad del método MODS dependen críticamente del nivel de experiencia del personal encargado de las lecturas. Lamentablemente, la factibilidad de contar con personal técnico permanente y apropiadamente entrenado para la lectura de placas MODS no es sostenible en el Programa Nacional de Control de Tuberculosis y del Ministerio de Salud, así como ocurre en otros países en vías de desarrollo que están comenzando a utilizar MODS en sus programas de control de TBC. Esto se debe, entre otras razones, a que el personal de laboratorio es escaso y reemplazado con frecuencia.

Frente a este problema, se planteó la construcción de una máquina con el propósito de automatizar la lectura de placas MODS, sin depender del microscopio invertido ni del técnico de laboratorio experto, y por lo tanto volver el método accesible a la mayoría de laboratorios de la red del Programa Nacional de Control de la Tuberculosis. Esta máquina sería capaz de leer y procesar automáticamente las placas MODS, a través de un mecanismo mecánico-óptico-electrónico y un programa de reconocimiento de patrones MODS característicos de tuberculosis. Se espera que este lector de placas automático sea de gran utilidad en hospitales con alta demanda de análisis de placas MODS y que en un futuro pueda convertirse en una potencial actividad productiva de interés, con posibilidades de llegar a ser un producto de exportación que sirva para auto sostener las actividades de desarrollo técnico-científico en el área.

0.3. Objetivos

Los objetivos de esta tesis son:

1. Diseñar y construir un equipo automático que permita la implementación del método MODS, desde la lectura correcta de placas hasta la etapa final de diagnosis de tuberculosis y archivo de información obtenida, tal y como lo realizaría un operador experto.

2. Establecer un procedimiento detallado para la lectura de placas MODS a nivel usuario empleando el equipo automático.

0.4. Alcances del proyecto

Si bien es cierto que en este reporte se expone la implementación del método MODS en base a un instrumento automatizado denominado en adelante como **MODS Plate Reader** (MPR), el proyecto MODS desarrollado desde el año 2007, encabezado por la Universidad Cayetano Heredia – Laboratorio de Bioinformática en conjunción con la Universidad Nacional de Ingeniería – Facultad de Ciencias, planteaba un objetivo general bastante más amplio: desarrollar una estrategia para hacer accesible el método de detección a nivel nacional resolviendo los problemas que complicaban su implementación masiva, a saber:

- El método MODS requiere del uso de un microscopio invertido, el cual es típicamente muy caro y no convencional en laboratorios de microbiología.

- El método MODS requiere de la presencia de personal técnicamente calificado para leer las placas y diagnosticar de manera correcta.

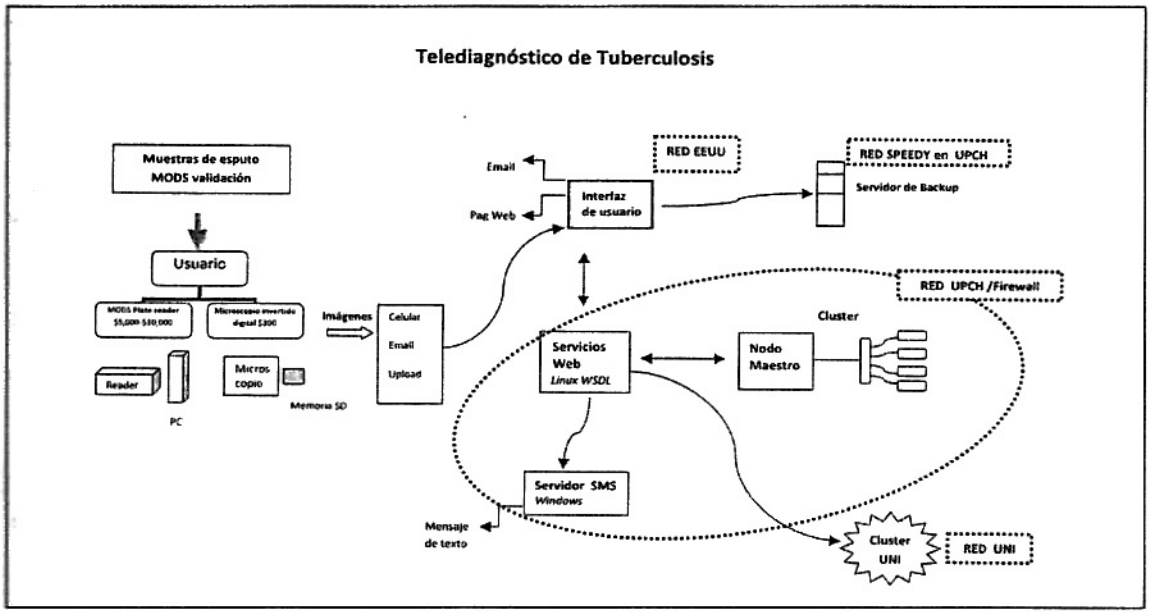


Figura 0.1: Esquema general del proyecto MODS

A fin de sortear dichos inconvenientes, se desarrolló una estrategia integral tal como se muestra esquemáticamente en la figura 0.1, la cual consistió de lo siguiente:

- Distribución de microscopios de bajo costo a laboratorios sobre el territorio nacional, con los cuales sea posible tomar fotografías de placas MODS de manera manual, con las condiciones adecuadas de iluminación y calidad de sensor que permitan un diagnóstico correcto.

- Implementación de un servicio web que permita un diagnóstico rápido y remoto, en base a fotografías tomadas en cualquier laboratorio y enviadas de manera digital a un cluster de procesamiento. Esto contempla la posibilidad de que el operador no tenga un conocimiento profundo sobre el reconocimiento de los cordones de TB.

Por otro lado, el equipo automático MODS Plate Reader, motivo de este reporte, se pensó para su distribución en hospitales con recursos económicos considerables. De esta manera, en un solo equipo se tiene la mejor manera de analizar las placas MODS, debido a lo siguiente:

- Incremento de bioseguridad, debido a la minimización de interacción humana con la placa MODS se consigue reducir el riesgo de contagio por manipulación de muestras.
- Iluminación óptima en términos de intensidad y espectro de frecuencias.
- Calidad adecuada del sensor de cámara digital en términos de resolución, sensibilidad y tasa de transferencia de datos en línea.
- Enfoque automático basado en maximización de contraste.
- Algoritmo de reconocimiento incorporado.
- Almacenamiento de imágenes y resultados en carpetas personalizadas.
- Detección de fallas, sistema de protección en caso de falla.

Durante el desarrollo de esta tesis se analizará en detalle cada uno de los puntos mencionados concernientes a la construcción del equipo MODS Plate Reader.

0.5. Definición de términos

En esta sección se establece la terminología utilizada a lo largo de este documento.

1. Términos relacionados con el método MODS

MODS. Siglas para Microscopic Observation Drug Susceptibility Assay. Podría ser traducido como Ensayo de Observación Microscópica de Susceptibilidad a Drogas. El método MODS fue desarrollado recientemente en los laboratorios de la Universidad Peruana Cayetano Heredia y permite el diagnóstico eficiente y a bajo costo de tuberculosis y susceptibilidad a drogas, a partir de una muestra de esputo del paciente.

MODS se basa en la identificación visual de un patrón microscópico característico de Mycobacterium tuberculosis, durante los primeros 7 días de crecimiento en un medio de fase líquida. MODS actualmente está reconocido y es utilizado como norma técnica en la política de control de tuberculosis y TB multidrogo resistente en el Perú [1, 2, 3, 4, 5, 6, 7].

Placa MODS. Placa de 24 pozos donde se ubican las muestras de pacientes. Estas placas están codificadas de tal manera que se puede identificar fácilmente muestras de pacientes, así como pozos con drogas [1]. Dimensiones aproximadas: Base horizontal = 4½'' x 3¼'', altura = ¾''. Diámetro de pozos = ⅝''.

TB. La tuberculosis es una enfermedad infectocontagiosa, de curso casi siempre crónico, causada por diversas especies del género mycobacterium. La especie más importante y representativa, causante de tuberculosis es el Mycobacterium tuberculosis o bacilo de Koch (descubierto por Koch en 1882). La tuberculosis es una enfermedad predominantemente de los pulmones aunque también puede afectar severamente al sistema nervioso central, el sistema circulatorio, los huesos, articulaciones e incluso la piel. Los signos y síntomas más frecuentes son: tos con flema por más de 15 días, a veces con sangre en el esputo, fiebre, sudoración nocturna, mareos momentáneos, escalofríos y pérdida de peso. El contagio se produce a través del aire, cuando las personas infectadas tosen, estornudan o escupen.

TB MDR. Tuberculosis multidrogoresistente. La tuberculosis MDR es definida como aquella que puede resistir al menos isoniacida y rifampicina, dos de los fármacos más poderosos en el tratamiento de primera línea.

MTB. Mycobacterium tuberculosis. Bacteria responsable de la enfermedad.

Nivel de Bioseguridad 2. En este nivel se manejan agentes de peligro moderado hacia el personal y el ambiente. Se caracteriza por contar con personal de laboratorio altamente calificado en el manejo de los agentes patógenos y el acceso al laboratorio es restringido cuando se está realizando algún trabajo [1].

Isoniacida. Antibiótico activo frente a MTB. Representa un fármaco de primera línea en el tratamiento de tuberculosis. No obstante, puede producir efectos secundarios como toxicidad hepática y neuropatía periférica.

Rifampicina. Antibiótico bactericida del grupo de las rifamicinas. La rifampicina está normalmente indicada en el tratamiento de las infecciones por Mycobacterium, como la tuberculosis y la lepra.

Capítulo 1

Parte teórica

1.1. Conceptos fundamentales

En esta sección de conceptos se presentan los fundamentos del método MODS necesarios para su comprensión a nivel usuario, destacando sus características funcionales y evitando en lo posible entrar en los detalles biológicos, los cuales escapan a los objetivos de esta tesis.

1.1.1. Fundamentos del método MODS

El método MODS consiste de un cultivo en medio líquido que permite el crecimiento de colonias de MTB para la posterior detección de patrones morfológicos característicos por medio de un microscopio invertido. MODS nos permite diagnosticar TB además de evaluar susceptibilidad frente a isoniacida y rifampicina directamente de las muestras de esputo en un promedio de 7 días. De esta manera, se puede determinar casos, no sólo de existencia de enfermedad, sino también de multidrogo resistencia tras el análisis de una sola placa MODS. El método se basa en dos propiedades importantes de MTB:

- El crecimiento es más rápido en medio líquido que en medio sólido.
- Su crecimiento característico en forma de cordón es fácilmente reconocible en el medio líquido usando un microscopio óptico invertido.

De esta forma, mediante el empleo de un microscopio óptico invertido para examinar una placa con muestras descontaminadas de esputo de pacientes, el método MODS nos permite detectar el crecimiento de los organismos responsables de TB y TB MDR de manera rápida, barata y efectiva [1, 2, 3, 4, 5, 6, 7, 8].

1.1.2. Ventajas comparativas del método

Con respecto a las principales ventajas que se encuentran en el método MODS para diagnosticar TB respecto a otros métodos usados en la actualidad [2, 3, 4, 5, 6, 7], podemos citar lo siguiente:

- La simplicidad de la técnica.
- La gran sensibilidad del medio líquido frente al medio sólido para la detección de TB.
- La especificidad del crecimiento característico de MTB en forma de cordones.
- La evaluación de la susceptibilidad frente a drogas (antibióticos) en un corto tiempo y el bajo costo de los reactivos.

1.1.3. Placa MODS

Las muestras de esputo tras su procesamiento son colocadas en placas especiales de 24 pozos, denominadas placas MODS selladas en bolsas plásticas ziplock. En la figura 1.1 se muestra de manera esquemática una típica placa MODS. En la figura se muestra la configuración de las placas indicando el ordenamiento con números (1-6) en columnas y letras (A-D) en filas.

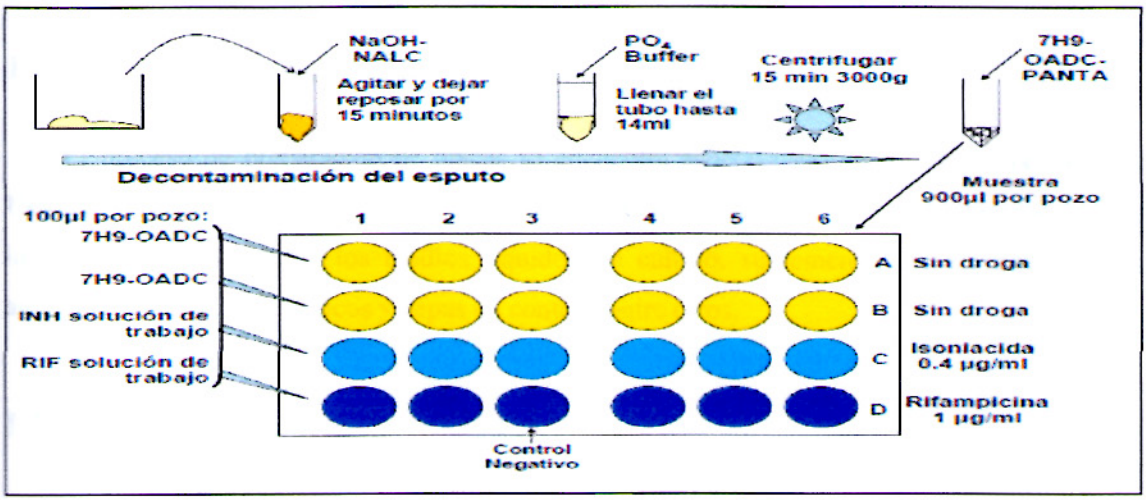


Figura 1.1: Placa MODS

En esta configuración, se indica que tanto la fila A como la B no contienen drogas que combatan la TB, mientras que la fila C contiene isoniacida y la D rifampicina, ambos antibióticos que combaten dicha enfermedad. Para el análisis regular de placas, cada una de las columnas, a excepción de la tercera, representa muestras obtenidas de un paciente distinto. La tercera columna es utilizada como control para detectar un posible caso de contaminación.

De esta manera, por placa se tienen cinco muestras de pacientes y una de control. Se volverá a tocar este punto en secciones posteriores. Finalmente, se muestra un bosquejo del procedimiento de preparación de las muestras, el cual escapa a los objetivos de este trabajo. En el documento denominado "Guía de Usuario MODS" se incluye información referente a este punto [1].

1.1.4. Bioseguridad

Para realizar el análisis MODS se requiere como mínimo llevar a cabo la lectura en un ambiente con Nivel de Bioseguridad 2 (P2). Podemos considerar las medidas de seguridad requeridas en dos grupos: infraestructura adecuada y buenas prácticas de laboratorio. Con respecto a infraestructura podemos mencionar lo siguiente: Cabina de bioseguridad de clase II en buen estado de mantenimiento donde el aire circule con facilidad, así como un ambiente de laboratorio separado del resto de laboratorios con ventanas bien selladas y puertas con cerradura para impedir ingreso de personal y turbulencias mientras se estén manipulando muestras. Con respecto a las prácticas de laboratorio: se debe contar con personal entrenado en bioseguridad y su importancia en laboratorio, protección adecuada (mandiles, guantes y mascarillas) así como la manipulación adecuada de placas MODS.

1.2. Metodología MODS

1.2.1. Preparación de placas

El proceso de preparación de placas se encuentra especificado en detalle en la **Guía de Usuario MODS** [1]. En esta guía se describe el procedimiento para preparar los buffers intermediarios necesarios, los medios líquidos de cultivo, suplementos de enriquecimiento, anticontaminantes, antibióticos y cepas de control, entre otros.

1.2.2. Lectura de placas

Una vez que las placas se encuentran preparadas se debe realizar la lectura de las mismas para determinar la presencia de TB y TB MDR. Para leer las placas, se emplea un microscopio óptico de magnificación total 100X con la característica de que sea invertido. Como su nombre indica un microscopio invertido tiene una disposición inversa de sus componentes respecto a un microscopio convencional: La fuente de luz se encuentra por encima de la placa, mientras que la lente objetivo está por debajo. El enfoque se produce en este caso con la variación de la posición de la componente óptica de lentes, manteniendo el soporte de muestras estático.

El motivo de esta elección tiene que ver con la característica de crecimiento de colonias MODS y la limitación de distancia de enfoque de un microscopio óptico convencional. A una magnificación de 100X, la distancia de enfoque resulta ser de alrededor de 6 mm, con lo cual resultaría imposible enfocar cordones con un microscopio convencional dado que el pozo tiene una profundidad mayor que la distancia de enfoque. Finalmente, este diseño nos permite tener acceso al soporte de muestras y reducir el efecto de vibraciones al estar anclado a la base.

1.2.3. Detección de TB

Un resultado positivo es definido como el crecimiento de dos o más unidades formadoras de colonia (cordones) en cada uno de los pozos sin drogas (filas A y B).

1.2.4. Procedimiento de detección de TB

El procedimiento típico de lectura y diagnóstico llevado a cabo por un investigador en laboratorio se detalla a continuación:

1. Las placas son retiradas de la incubadora para su observación en el microscopio invertido.
2. La lectura de las placas se inicia por los pocillos sin antibióticos en el día 5 de incubación. Como se mencionó, al inicio del crecimiento, las colonias MTB forman pequeñas comas o espirales. Este proceso por lo general progresa formando cordones para finalmente crear una estructura mucho más compleja (figura 1.7).
3. De encontrarse más de dos cordones en los pocillos sin drogas (antibióticos), se determina que el cultivo es positivo.
4. Si los resultados son negativos, se debe continuar con la lectura diaria de los pozos sin antibióticos.
5. De observarse algún resultado positivo, se procede a realizar la lectura de los pozos con isoniacida y rifampicina el mismo día.
6. Si no se observa crecimiento hasta el día 21, el resultado final será determinado como cultivo negativo.
7. No se deben leer los pozos con droga, si la lectura en pozos sin droga es negativa.
8. Si los pozos se contaminaron con bacterias u hongos, se debe reprocesar las muestras ya que en ese estado no es posible diagnosticar.

Resumimos en la tabla 1.1 la lectura e interpretación de los pozos libres de drogas.

Tabla 1.1: Interpretación de resultados en pozos sin drogas

Observación de un pozo sin droga (A o B)	**Interpretación de los resultados**
Más de dos unidades formadoras de colonia	**Positivo**
No crecimiento	**Negativo**
Sobrecrecimiento de bacterias u hongos	**Contaminado**

Antes de considerar a los resultados finales como válidos, se hace notar que los controles positivos y negativos deben ser examinados y validados antes de analizar cualquier muestra de pacientes (revisar apartado 1.2.6 sobre controles internos).

1.2.5. Procedimiento de detección de TB MDR

Los pozos con antibióticos sólo deben examinarse cuando los pozos sin drogas son positivos, esto es cuando en las dos primeras columnas se ha encontrado evidencia de TB. La resistencia es definida como el crecimiento micobacteriano de dos o más unidades formadoras en los pozos con drogas el mismo día en que ambos pozos sin drogas son positivos. La susceptibilidad a las drogas tiene un significado opuesto al de resistencia: si se determina que una cepa es resistente a una droga, entonces se dice que no es susceptible a la misma. En la tabla 1.2 se resume la lectura e interpretación de los pozos con drogas.

Tabla 1.2: Interpretación de resultados en pozos con drogas

Observación en pozos C & D en todas las filas	**Interpretación de resultados**
No hay crecimiento en ninguno de los pozos conteniendo drogas.	**TB Sensible (no es MDR)**
Crecimiento sólo en el pozo de isoniacida (INH).	**Resistente a INH (no es MDR)**
Crecimiento sólo en el pozo de rifampicina (RIF).	**Resistente a RIF (no es MDR)**
Crecimiento en ambos pozos con drogas.	**Multidrogoresistente (MDR)**
Crecimiento de una unidad formadora en cualquier pozo con droga.	**Indeterminado para esa droga.**
Cualquiera de los pozos con drogas contaminado.	**Indeterminado para esa droga.**

1.2.6. Controles internos

1. Control Positivo

Este control se ejecuta cuando se lee la primera placa MODS del día. El procedimiento se explica a continuación: Se debe preparar una placa (aparte de las de lectura cotidiana) con dos cepas denominadas de control positivo: una cepa sensible a todas las drogas y una cepa MDR. En la placa MODS de control positivo, en la columna 5 se coloca la cepa sensible a todas las drogas y en la columna 6, la cepa MDR (el resto de columnas no contienen material biológico). Los controles positivos permiten evaluar la calidad del medio de cultivo y los antibióticos empleados el mismo día del proceso. Si no se observa crecimiento con los patrones esperados en los controles positivos, los resultados de las muestras procesadas en ese día no son válidos, dando como resultado el descarte de la integridad de lotes de placas del día.

2. Control Negativo

Estos son los pozos con medio de cultivo, pero sin muestra, los cuales se ejecutan en cada placa de análisis MODS. Todos los 4 pozos en la columna 3 (3A, 3B, 3C y 3D) no deberían tener crecimiento de cordones. Si se observa alguna colonia en cualquiera de esos pozos, entonces se ha producido contaminación cruzada y la placa debe ser descartada. A manera de resumen se muestra los resultados esperados en la tabla 1.3 para cada tipo de control interno.

Tabla 1.3: Resultados esperados para controles internos

Tipo de control	Cepa	Medio	Resultados esperados
Negativo	Ninguno	4 pozos por columna (2 libres de droga, 1-INH, 1-RIF)	**No hay crecimiento**
Positivo	Completamente sensible	Pozos libres de drogas	**Crecimiento**
		Pozos que contienen INH-RIF	**No hay crecimiento**
	MDR	Pozos libres de drogas	**Crecimiento**
		Pozos que contienen INH-RIF	**Crecimiento**

1.3. Fundamentos de motores a paso

1.3.1. Características principales

Un motor a paso (figura 1.2) es un motor eléctrico síncrono, sin escobillas (brushless) capaz de dividir su rotación en un gran número de pasos. La posición del eje del motor puede ser controlada muy precisamente sin ayuda de ningún mecanismo de lazo de realimentación (feedback loop), siempre que haya sido dimensionado correctamente y no haya problemas de falta de torque. Los motores a paso tienen un fundamento de operación distinto a los típicos motores DC con escobillas, que rotan cuando se aplica un voltaje continuo a sus terminales. Su característica primordial es que el eje de salida rota en una serie de intervalos angulares discretos o pasos. Cada paso se produce cuando una señal de comando de pulsos es recibida de un computador. Cuando el motor recibe un número definido de pulsos, el eje rota un cierto ángulo conocido y esto lo hace ideal para aplicaciones en lazo abierto (figura 1.4).

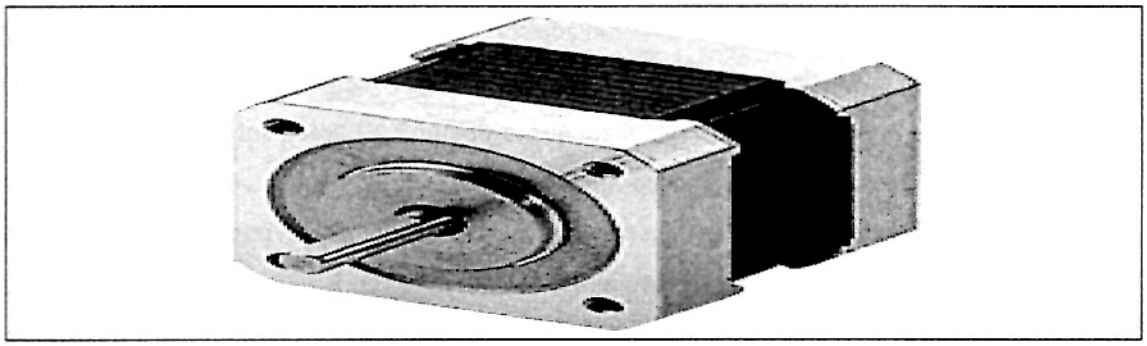


Figura 1.2: Motor a pasos

1.3.2. Principio de operación

El principio en el que se basa el funcionamiento de estos motores es en realidad muy simple: Cuando una barra de hierro o acero está suspendida de tal manera que es libre de rotar en un campo magnético, ésta se alineará de manera natural con el campo externo. Si la dirección del campo cambia, la barra rotará hasta alinearse nuevamente, por acción del torque de reluctancia. El rotor en este tipo de motores está diseñado de tal manera que tiene proyecciones o "polos" que se alinean con el campo magnético producido por los arrollamientos del estator. A continuación, se muestra en la figura 1.3 un típico motor a paso híbrido de 200 pasos por revolución y algunas características de su construcción. Las ocho bobinas o polos principales en la figura 1.3 están conectadas para formar arrollamientos de dos fases. Las bobinas en los polos 1, 3, 5 y 7 forman la fase A, mientras que 2, 4, 6 y 8 forman la fase B.

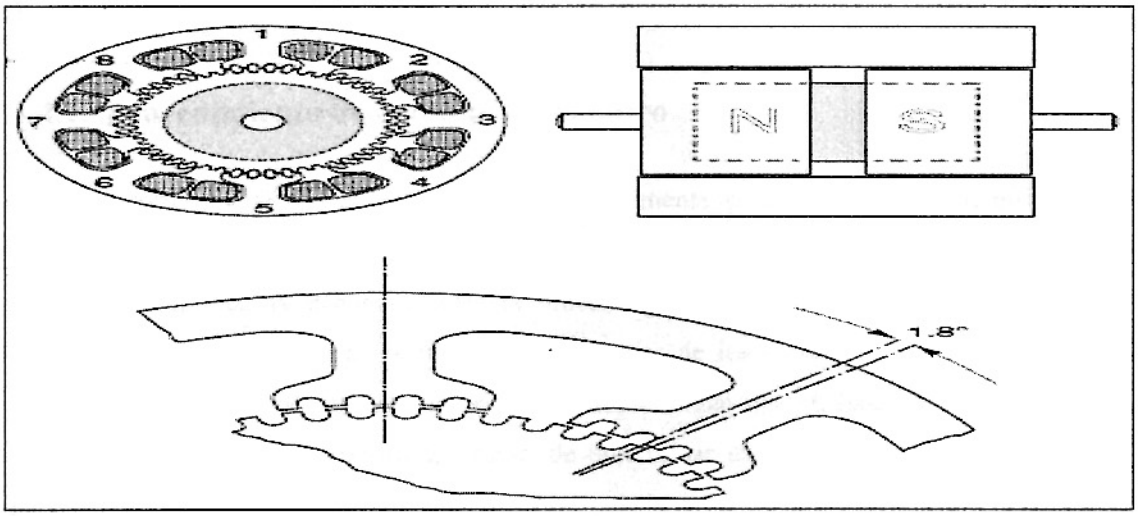


Figura 1.3: Motor a paso híbrido. Se muestra un motor híbrido con 8 polos principales, 50 dientes de rotor y 200 pasos por revolución. El detalle muestra el alineamiento entre el estator y el rotor, así como el ángulo de paso de 1.8°.

Cuando la fase A transporta corriente de estator positiva los polos 1 y 5 se magnetizan como S (sur) y los polos 3 y 7 se convierten en N (norte). Los dientes desfasados en el terminal N del rotor son atraídos a los polos 1 y 5 mientras que los dientes desfasados en el terminal S del rotor son atraídos por los polos 3 y 7. Para conseguir que el motor gire un paso, la fase A se desactiva, y B se activa con corriente positiva o negativa, dependiendo del sentido de giro deseado. Esto causará que el motor se mueva un cuarto del ángulo pitch (ángulo entre dientes), esto es 1.8°, a una nueva posición de equilibrio. El motor continúa barriendo pasos, a medida que las fases son energizadas en la secuencia siguiente: +A, -B, -A, +B, +A (sentido de agujas del reloj) o +A, +B, -A, -B, +A (contrario a agujas del reloj). Finalmente, se resalta la necesidad de contar con un driver bipolar, esto es un dispositivo que tenga la capacidad de invertir corriente por las bobinas del estator. Cuando el motor opera de esta manera, se le conoce como motor de dos fases con fuente bipolar.

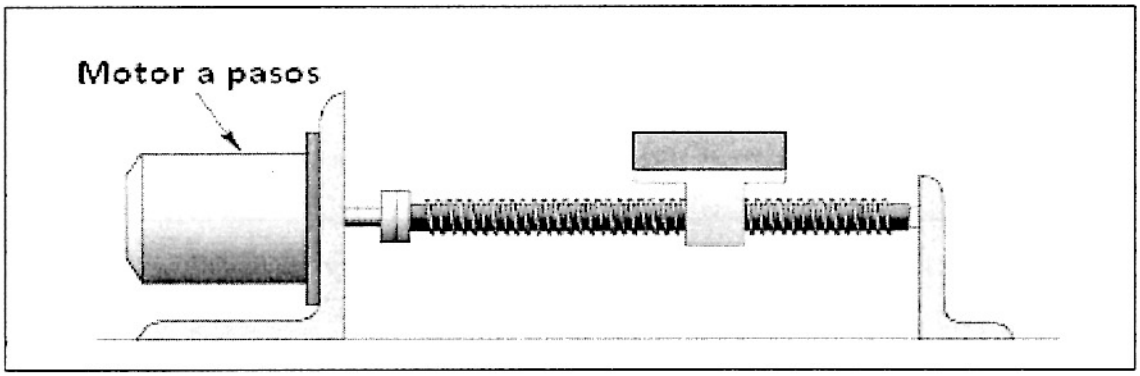


Figura 1.4: Aplicación de motor a pasos para control en lazo abierto

1.4. Enfoque dinámico

1.4.1. Procedimiento de enfoque automático

Los cordones de TB en placas MODS difícilmente se encuentran en un mismo plano horizontal dentro de un pozo. De esta manera, a fin de mantener a los cordones en una posición de enfoque óptima, la distancia entre la muestra y la lente objetivo debe ser modificada dinámicamente de acuerdo a las posiciones aleatorias de los cordones en la muestra. Para el proyecto se ha implementado un algoritmo de autoenfoque que en base al menor número de fotografías tomadas a la muestra sea capaz de determinar una posición de máximo contraste, para luego, mediante un servomecanismo, conseguir la distancia ideal entre muestra y objetivo. La idea general del algoritmo de enfoque se explica como sigue: A varias posiciones bajo la muestra, a lo largo del eje óptico (perpendicular al plano de cordones) se calcula una cantidad en base a una fotografía tomada. Esta cantidad debe contener información con respecto al contraste de la imagen. Luego de tener un juego de datos, se elige una cierta función empírica (gaussiana, polinómica, etc.) para ajustar los datos obtenidos. En base a nuestro modelo encontramos el pico de la curva y lo consideramos como nuestro estimado de la posición de mejor enfoque.

1.4.2. Gradiente de Brenner

El algoritmo de autoenfoque depende fuertemente de nuestra capacidad de definir correctamente un parámetro de enfoque con información relevante sobre el contraste de una imagen, dado que fundamentalmente se espera que el resultado del proceso de enfoque sea la captura de la imagen con mejor contraste posible para el diagnóstico (figura 1.6). En la literatura se encuentra un buen número de algoritmos para definir cantidades de este tipo [10, 11]. Cualitativamente, cuando una imagen está enfocada, se espera que tenga un alto contraste, un gran número de picos de contraste bien definidos y bordes bien marcados. Cuantitativamente, este número debería ser máximo en el foco y decrecer simétrica y monotónicamente hacia cualquier lado del plano focal. La cantidad que hemos usado en este proyecto para estimar el contraste de una imagen se muestra a continuación:

$$B = \sum_{i=1}^{m-2}\sum_{j=1}^{n}\left[s(i,j) - s(i+2,j)\right]^2 + \sum_{j=1}^{n-2}\sum_{i=1}^{m}\left[s(i,j) - s(i,j+2)\right]^2 \qquad (1.1)$$

Esta cantidad será conocida en adelante como Gradiente de Brenner.

En esta fórmula, "s" representa un pixel de una imagen en escala de grises, esto es, un número de tipo entero en el rango 0-255 y el barrido de las sumas se realiza sobre toda la imagen capturada. El gradiente de Brenner es un rápido y rudimentario detector de bordes, midiendo la diferencia entre la intensidad de un pixel y su vecino, separado por dos pixeles en este caso. Notar que el gradiente de Brenner así definido, no muestra ningún tipo de preferencia direccional tal como los cordones en las muestras, los cuales crecen distribuidos aleatoriamente sin mostrar tampoco ninguna preferencia direccional.

Se ha observado de manera empírica que es posible ajustar el gradiente de Brenner por una función lorentziana en un pequeño rango de trabajo (del orden de las micras), tal como se sugiere en [10] y se muestra en la figura 1.5. La ecuación para la curva lorentziana de ajuste de datos se muestra a continuación:

$$y = \frac{\alpha}{\beta + (z - z_0)^2} \tag{1.2}$$

En esta ecuación, la variable independiente es la coordenada z de toma de datos y la variable dependiente se corresponde con nuestro modelo de curva para gradiente de Brenner. Como se puede observar, la ecuación 1.2 requiere de tres parámetros: α, β y z_0 para especificar la curva completamente. Estos parámetros se pueden encontrar como solución de un sistema de tres ecuaciones con tres incógnitas, para lo cual se requiere calcular gradientes de Brenner en tres posiciones distintas z. Entre los parámetros del modelo, el más importante es z_0, el cual nos brinda la información concerniente al máximo de la curva.

Como se mencionó, el enfoque basado en el modelo lorentziano tiene un carácter local y micrométrico de acuerdo al rango de trabajo en el que opera. En la práctica, sin embargo, se observa que los planos de cordones en pozos diferentes pueden estar separados centenas de micras, por lo que un enfoque micrométrico resulta insuficiente. Debido a esto se implementó un procedimiento previo de barrido con un rango de trabajo bastante mayor (del orden de los milímetros), con el objetivo de encontrar un primer estimado del máximo de la curva modelo.

Debido a esta diferencia apreciable en los rangos de trabajo, se decidió utilizar el término enfoque global o macrométrico al proceso "grueso" de barrido en búsqueda del máximo. Una vez obtenido el resultado de enfoque macrométrico, se procede con el enfoque más fino basado en el modelo lorentziano en la vecindad del resultado macro.

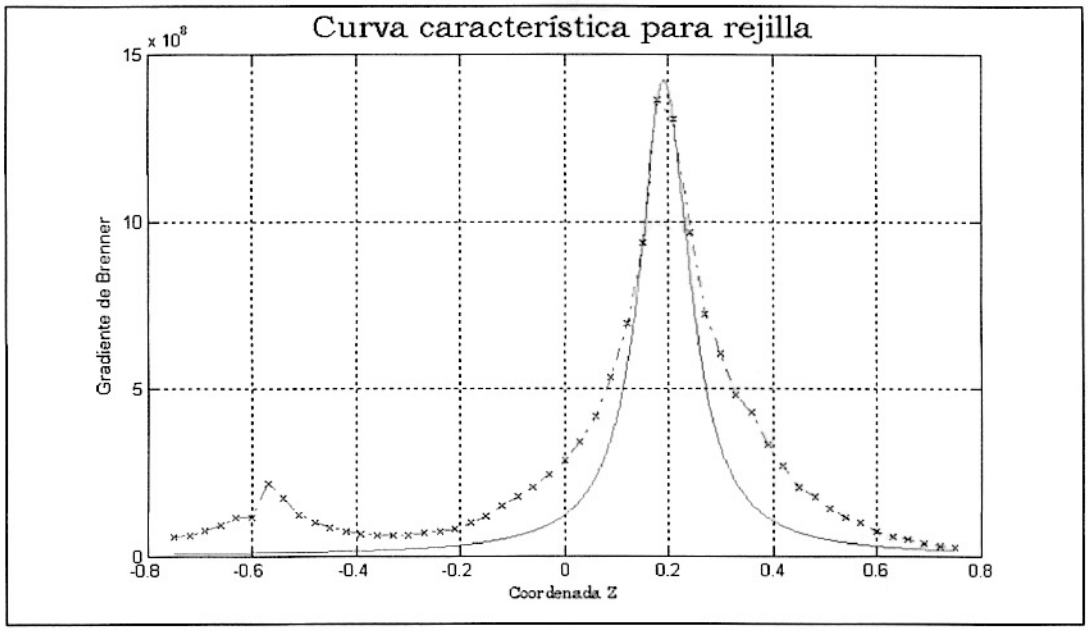


Figura 1.5: Curva de contraste para rejilla de difracción de 300 mesh (líneas por pulgada). En color azul, la curva original obtenida del proceso y en rojo, el ajuste a lorentziana. Se observa un pequeño pico al lado izquierdo del principal, debido a la lectura del fondo del pozo de la placa. Nótese la gran similitud con la curva lorentziana en la vecindad del pico.

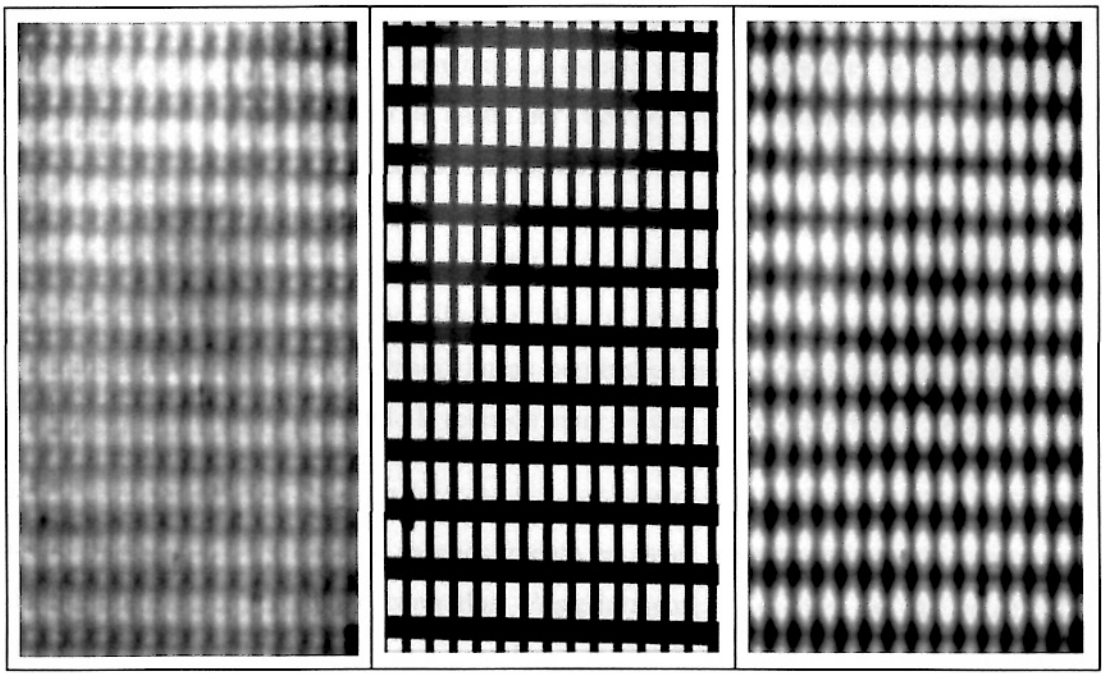


Figura 1.6: Resultado de enfoque automático. Al lado izquierdo se muestra la posición de inicio de enfoque, al lado derecho la posición final de búsqueda de mejor contraste y en la posición central el resultado del algoritmo de enfoque automático para una rejilla de 300 mesh.

1.5. Algoritmo de diagnóstico de TB

A continuación se presenta un breve resumen sobre el extenso trabajo realizado en los laboratorios de Bioinformática en UPCH concerniente al algoritmo de reconocimiento de TB.

En base al estudio y observación de los técnicos de laboratorio que han venido trabajando con cultivos MODS por años, se consiguió identificar cinco características principales que nos permiten reconocer cordones correctamente: forma sinusoidal, longitud y anchura de cordón, terminaciones en forma de punta y refracción característica. De esta manera, a fin de poder replicar los resultados obtenidos por los investigadores acerca de la sensibilidad y especificidad del método de diagnóstico de cultivo MODS, se tuvo que analizar y buscar una interpretación matemática a cada una de las características de los cordones. En este análisis, resulta conveniente trabajar en base al concepto de "backbone" de los cordones. Este backbone resulta de reducir un objeto al límite en el que cualquier punto del objeto procesado tenga como mínimo uno y a lo más tres vecinos; esto es, píxeles blancos. Procesando imágenes de cordones, lo que se encuentra es que los backbones característicos muestran un cuerpo principal y una serie de ramificaciones, las cuales terminan en una punta, al igual que los extremos del cuerpo. En términos del backbone, podemos referir que la longitud del cordón se interpreta como la longitud del cuerpo principal y el ancho como una medida de separación entre puntas opuestas de las ramificaciones del cuerpo principal. La característica de curvatura se analiza comparando el cuerpo principal con su recta de ajuste por regresión lineal, adicionalmente se emplea FFT para encontrar algunas características de formas de onda asociadas con el cuerpo principal. La refracción característica tiene que ver con la observación de que existe una variación de iluminación en dirección desde el centro del cuerpo hacia las puntas de las ramificaciones, esto como consecuencia de la forma cilíndrica de los cordones. En escala de grises es posible cuantificar este efecto construyendo líneas perpendiculares partiendo de las puntas hacia el cuerpo del backbone y analizando la relación entre píxeles de mayor y menor valor.

Hasta aquí se he hecho mención sólo a las características principales, sin embargo se debe mencionar que el algoritmo de reconocimiento es capaz de extraer hasta 48 parámetros con descripción de los objetos encontrados. Posteriormente al cálculo de los parámetros que describen a los objetos, es necesario realizar un análisis estadístico con muestras, a fin de encontrar aquellos parámetros que resultan efectivos para diagnosticar TB. En este punto, se emplea un procedimiento de regresión logística, el cual es considerado el método estándar para modelar una respuesta binaria (presencia o ausencia de TB). De este modelo, se extraen los coeficientes asociados a cada variable.

Después de la selección del modelo más representativo para el diagnóstico se procede con un análisis de coherencia biológica y estadística para cada variable considerada. Finalmente, con el modelo encontrado, se realiza el proceso de validación utilizando muestras positivas y negativas en múltiples oportunidades. Los resultados obtenidos son muy satisfactorios consiguiéndose una sensibilidad de 99.7% y una especificidad de 99.4%. A continuación se muestran algunas fotografías de colonias a diez días obtenidas usando el lector automático.

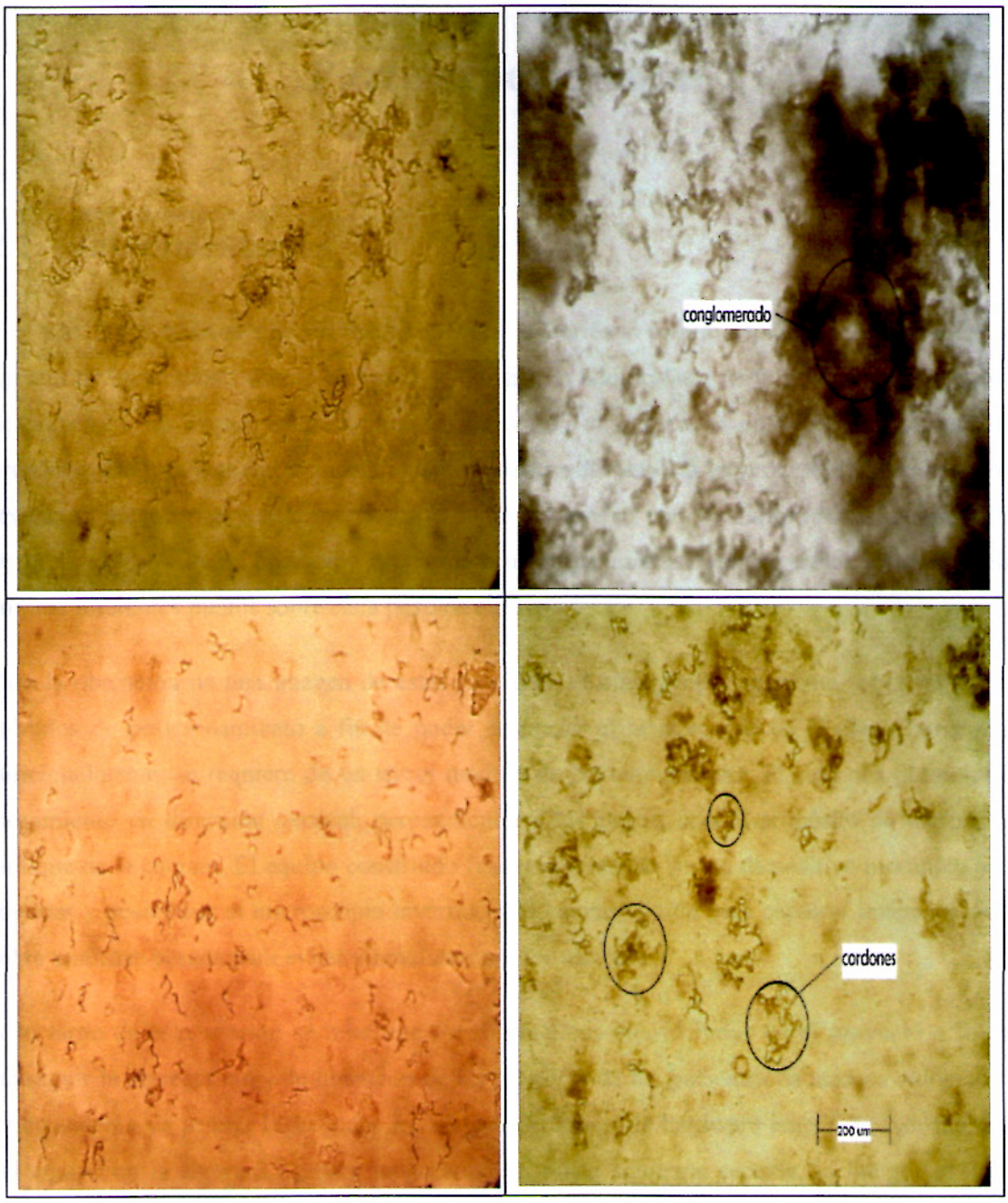


Figura 1.7: Muestras de TB a diez días de cultivo

Capítulo 2

Parte experimental

2.1. Instrumentación

Los datos básicos que se requieren para la correcta determinación de la presencia de agentes de tuberculosis son un conjunto de fotografías adquiridas a partir de muestras de esputo de pacientes, las cuales se encuentran en pozos, en una placa de cultivo adecuada para tal propósito. De esta manera, el procedimiento a desarrollar implica un proceso de adquisición de imágenes, diagnóstico y archivo adecuado de las fotografías para posterior identificación. A fin de llevar a cabo dicha operación se requiere usar un microscopio invertido, el cual pueda enfocar diferentes puntos sobre cada pozo de cultivo.

Para la obtención de una imagen de estas muestras, el microscopio se monta sobre un par de módulos de posicionamiento a fin de poder desplazar el microscopio en el plano horizontal. Adicionalmente, se requiere de un tercer módulo de posicionamiento, el cual nos permitirá el movimiento en dirección vertical, perpendicular a los pozos, con el propósito de mejorar el enfoque de la cámara. El equipo construido finalmente resulta ser un dispositivo mecánico para soportar y posicionar el microscopio invertido y la cámara digital para capturar imágenes, todo sobre una base pesada para evitar vibraciones por los motores.

A lo largo de esta sección se muestran los dispositivos considerados en el proyecto a fin de conseguir la correcta toma de imágenes de las muestras de los pacientes de manera automática. Se consideran en primer lugar todos los elementos físicos de hardware empleado (ver montaje del equipo en la figura 2.1); para luego analizar, en la siguiente sección, la integridad de la componente de software de control y su interacción con el hardware, así como el proceso de captura de imágenes y diagnóstico de la enfermedad. Finalmente, se describen los algoritmos relacionados con el procedimiento de lectura de placas de acuerdo con el método MODS.

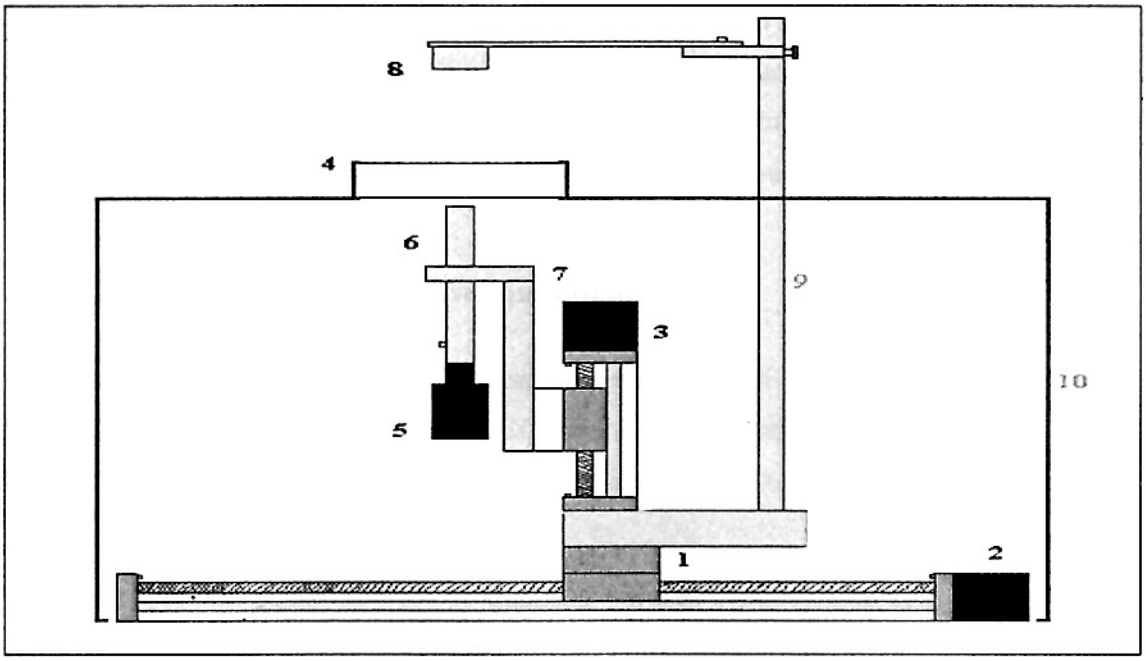


Figura 2.1: Montaje de MODS Plate Reader. Se muestran los posicionadores en dirección X (1) en vista de perfil, dirección Y (2), así como Z (3) en vista frontal, manejados por motores a pasos, placa MODS (4) con muestras de tuberculosis, cámara empleada (5), soporte para lentes del microscopio (6), soporte para la integridad del microscopio (7), lámpara empleada (8), soporte para lámpara (9) y mesa de soporte de placas (10) anclada a la base del equipo.

1.1.1. Sistema de coordenadas

Para la determinación de las posiciones dentro del área de trabajo se empleó un sistema XYZ de coordenadas cartesianas con particularidad de mano izquierda (figura 2.2). Esta elección resulta natural tras considerar el sentido de avance del tornillo sinfín empleado. Se ha considerado como límite superior el fin de carrera más cercano al motor a pasos en cada una de las direcciones y el límite inferior o cero de coordenadas el indicado por los fines de carrera más alejados de los motores a paso.

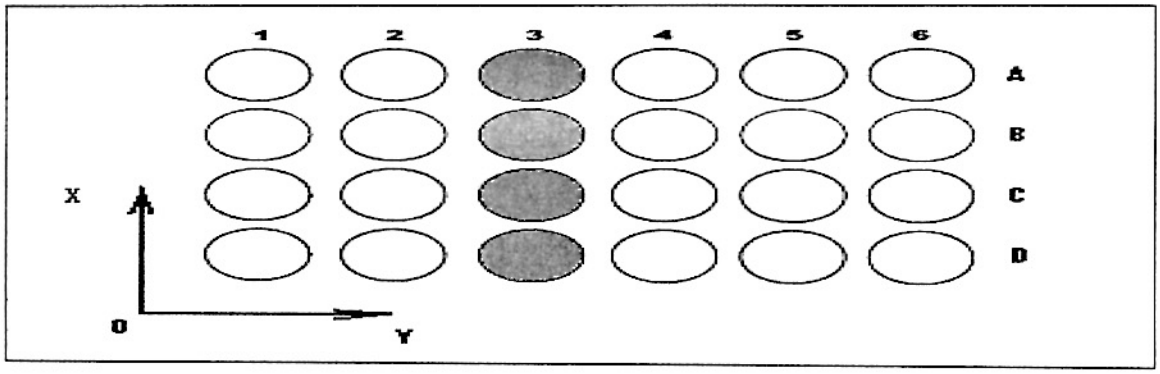


Figura 2.2: Placa MODS y Sistema de Coordenadas

De la figura 2.2 se puede concluir que para un barrido normal de una placa MODS, esto es, desde la columna 1 hasta la columna 6 (sin considerar la columna 3 de control negativo) y desde la fila A hasta la fila D, para un paciente dado (una columna de la placa), la dirección de barrido es en el sentido de -X y entre pacientes distintos es en la dirección de +Y.

1.1.2. Sistema mecánico de posicionamiento

El sistema de posicionamiento se basa en tres módulos de posicionamiento de la marca VELMEX, código: XN10-00XX-E25-71 (figura 2.3). Cada módulo cuenta con un tornillo sin fin, un motor a paso y sensores de fin de carrera en ambos extremos del recorrido.

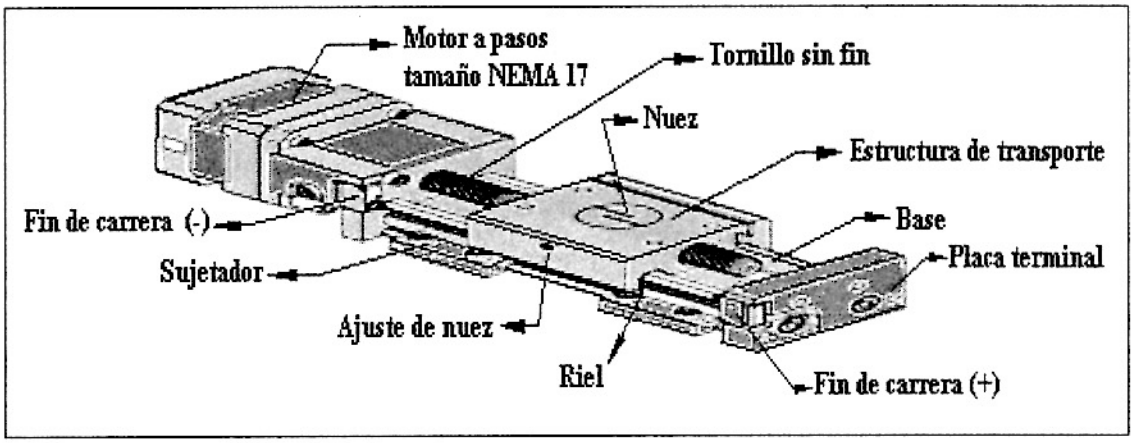


Figura 2.3: Módulo de posicionamiento

Considerando las dimensiones de las placas y la estructura mecánica para sostener los elementos del lector se decidió adquirir tres módulos con las características mencionadas en la tabla 2.1.

Tabla 2.1: Características principales de los posicionadores

Características principales	Valor
Distancia de trabajo	Para dirección X = 6'' Para dirección Y = 8'' Para dirección Z = 2''
Avance por vuelta	En todos los casos 0.025''
Tipo de motor	Motor a pasos PK245-01AA, NEMA 17 de dos fases y 200 pasos por vuelta.
Máx. Torque dinámico perp. al tornillo	3.6 N.m
Máx. torque dinámico paralelo al tornillo	2.5 N.m
Fines de carrera	Incluidos, tipo NC en ambos extremos

En la figura 2.4 se presenta la configuración XYZ utilizada para los módulos. Los módulos XY permiten la ubicación del microscopio en las posiciones de adquisición predeterminadas, mientras que el módulo Z permite mejorar el contraste de la imagen capturada por la cámara. Estos posicionadores son anclados a una base pesada (plancha de fierro 70 cm x 70 cm x ¼'', con patas de amortiguamiento) para evitar el efecto indeseable de las vibraciones sobre las muestras.

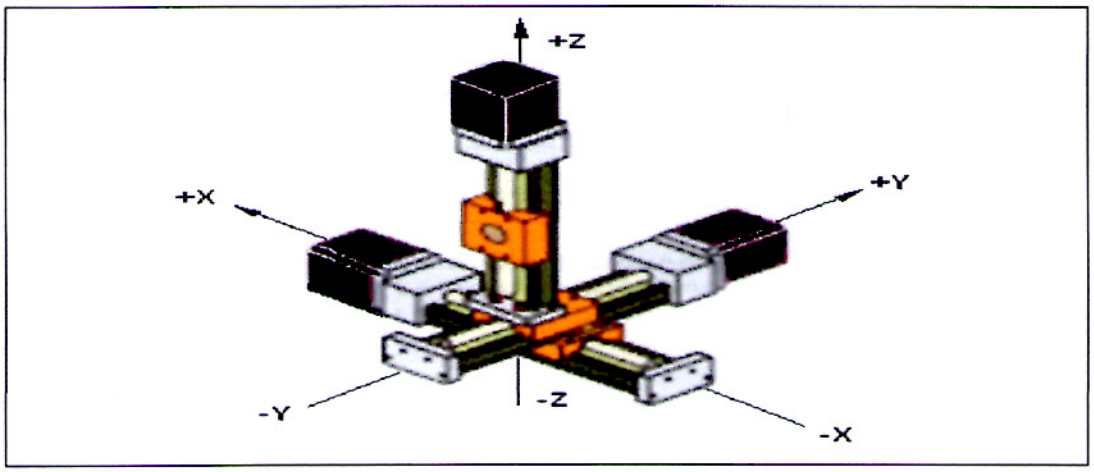


Figura 2.4: Módulos de posicionamiento configurados para movimiento XYZ

1.1.3. Sistema de iluminación

El sistema de iluminación es de importancia crítica para la formación apropiada de imágenes, debido a esto, se dedicó una considerable cantidad de tiempo para analizar distintas fuentes de luz (figura 2.5). Tras observar microscopios comerciales en el laboratorio y en particular, una lámpara comercial de la marca Edmund (con la cual se consiguió ver cordones correctamente), se buscó una lámpara que pudiera reproducir las características de iluminación y espectro de la mencionada lámpara sin encarecer demasiado el equipo. Este análisis se realizó en la Facultad de Ciencias – Universidad Nacional de Ingeniería, utilizando un espectrómetro del laboratorio de Películas Delgadas [9]. El resultado del análisis espectral muestra que la lámpara dicroica de la marca Philips (12 VDC 50 W) es la que más se asemeja espectralmente a la lámpara dicroica de referencia de Edmund (30 W). Cabe señalar que la lámpara dicroica elegida para el sistema de iluminación del MODS Plate Reader es la misma que se empleó originalmente en el prototipo de microscopio económico parte del proyecto integral, debido a su bajo costo y correcta funcionalidad.

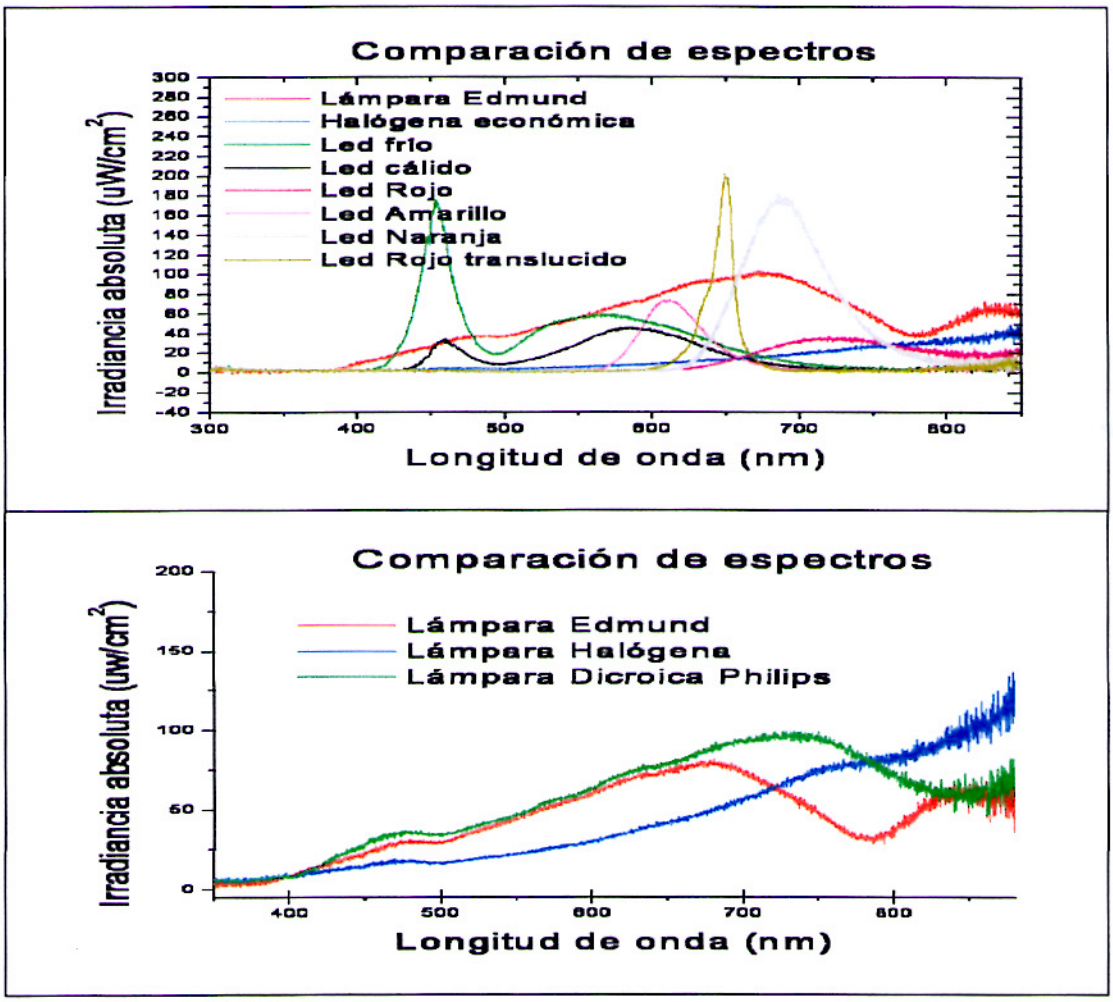


Figura 2.5: Comparación de características espectrales entre diversas lámparas comerciales

En la tabla 2.2 se muestran algunas características de la lámpara seleccionada para el equipo.

Tabla 2.2: Características principales de la lámpara de iluminación

Características principales	Definición
Modelo	Philips's Essential
Código	DIC12V50W36BLIST
Potencia	50 W
Voltaje base	12 VDC
Máxima intensidad luminosa	1200 cd
Apertura de haz	36°
Promedio de vida	2000 horas

El soporte mecánico para la lámpara dentro de la estructura del equipo se muestra en la figura 2.1, indicado con referencia al elemento 9, así como el detalle de sus partes en la figura 2.6. El objetivo del soporte es permitir que la lámpara siga el recorrido XY del microscopio, a fin de mantener un nivel de iluminación constante. Debemos considerar adicionalmente las restricciones de carga mecánica y tipo de esfuerzos que soportan los posicionadores.

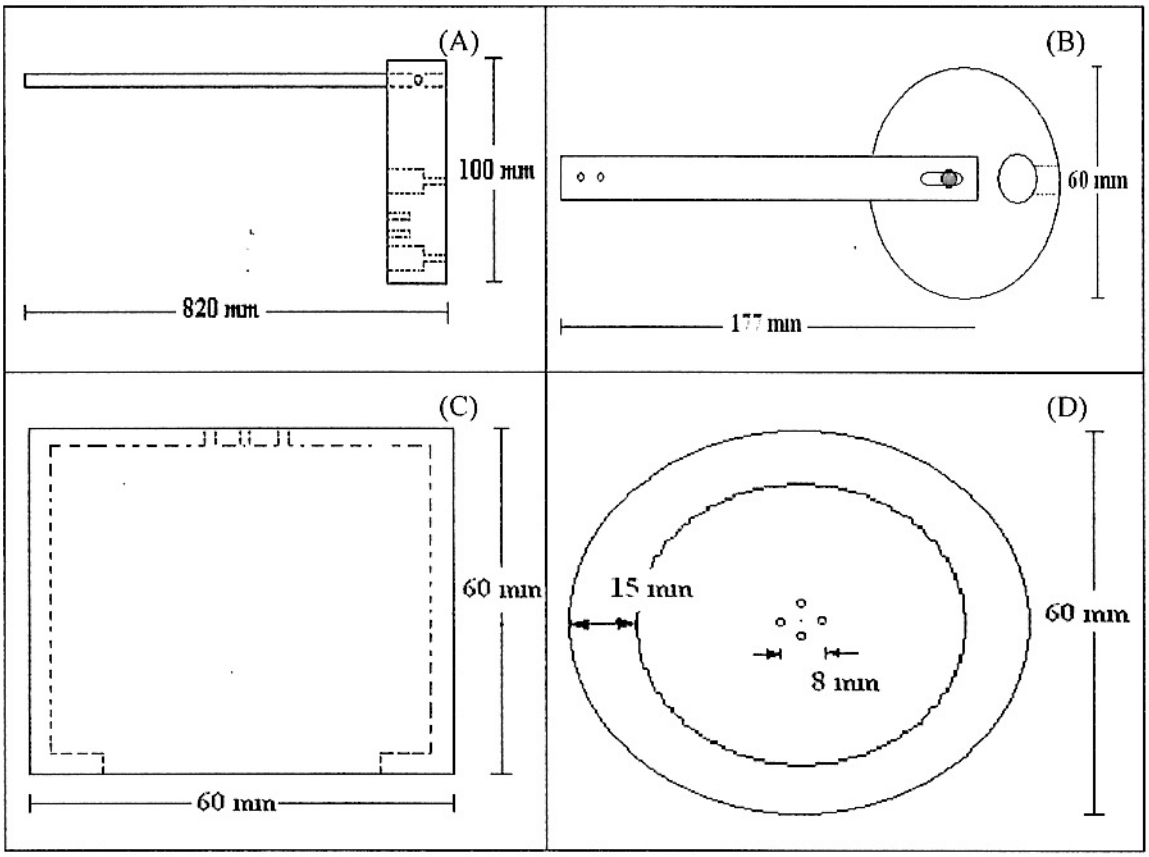


Figura 2.6: Soporte y socket para lámpara

Como se puede observar en la figura 2.1, el soporte se apoya en los posicionadores XY, produciendo un esfuerzo perpendicular al tornillo. Esto deviene del análisis de los datos característicos de los engranajes, según el cual podemos concluir que un esfuerzo para mover una carga en dirección hacia el tornillo sin fin es bastante más eficiente que uno en dirección paralela. De esta manera, se consigue que la lámpara siga la trayectoria de las lentes con un punto de apoyo suficientemente sólido. El soporte mecánico de la lámpara se completa con un disco (figura 2.6B) que permite un giro alrededor de la vertical y un socket especial (figura 2.6C y 2.6D) para la lámpara. Adicionalmente, cabe señalar que, debido al diseño mecánico del equipo, se reduce considerablemente el efecto de falta de alineamiento entre la lámpara y la cámara, pasando por las muestras en los pozos, lo cual contribuye efectivamente a la formación de imágenes correctamente enfocadas.

1.1.4. Sistema óptico y de captura de imágenes

En esta parte del trabajo se decidió aprovechar los resultados obtenidos por miembros del grupo de investigación de la unidad de Bioinformática en la Universidad Cayetano en un estudio previo con el fin de reducir los costos asociados a la componente óptica del microscopio invertido económico, clave en la estrategia para masificar el empleo del método MODS [9].

Recordemos que MODS utiliza un microscopio óptico invertido para visualizar el crecimiento característico de MTB en forma de cordones por la parte baja de las placas de cultivo. El microscopio invertido es usado para superar dos problemas que se encuentran al observar cultivos con microscopios convencionales:

- La distancia de la lente objetivo al fondo de las placas, donde las colonias de MTB tienden a conglomerarse, es mayor que la distancia de trabajo de un objetivo de microscopio de luz estándar. Es más, esta diferencia se incrementa si se requiere mayor magnificación y apertura numérica.

- La condensación que se forma en la tapa de la placa, así como la obstrucción e interferencia del medio líquido del cultivo, usualmente oscurecen la visión desde la parte superior.

Según las pruebas realizadas con el lector manual, para cultivos de alrededor de 7 días en adelante resultaba suficiente contar con una lente objetivo de 10X y un ocular de 10X para que un experto pueda reconocer los cordones.

De esta manera, se comprobó que para enfocar los cordones no era necesario recurrir a costosas lentes objetivo corregidas (típicas de microscopios comerciales), sino que bastaba usar un objetivo acromático con distancia de trabajo bastante pequeña (aproximadamente 6mm). El rendimiento para visualizar patrones MODS del microscopio económico desarrollado con anterioridad al lector automático es comparable al de un microscopio comercial NIKON (NIKON Eclipse TS100-F). El proceso de validación con muestras positivas mostró un elevado acuerdo entre las lecturas llevadas a cabo en ambos microscopios por un mismo técnico de laboratorio, consiguiéndose un 98.3% en el caso del microscopio comercial contra un 96.6% del microscopio económico. Los resultados satisfactorios en términos de calidad de imagen y reducción de costos fueron determinantes para que el diseño de la componente óptica del lector automático se base en el diseño del microscopio manual de bajo costo. En la figura 2.7 se muestra un diagrama simplificado del diseño original del microscopio económico.

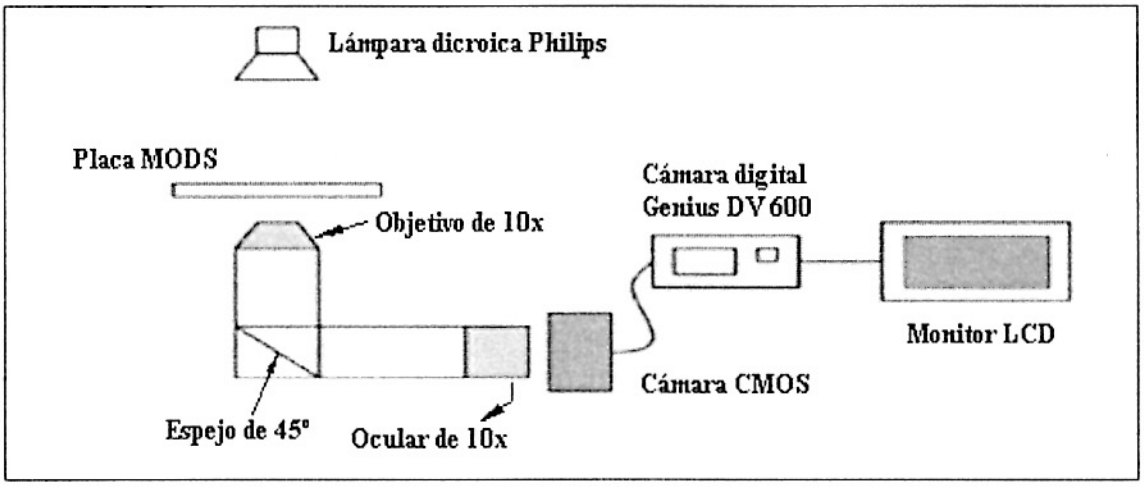


Figura 2.7: Diseño original del componente óptico del lector MODS económico

Volviendo al diseño del lector automático, en la figura 2.1 se muestra la lente objetivo ubicada en la parte superior del soporte estándar de 160 mm, en el lado más cercano a la placa MODS (notar la similitud con la figura 2.7). En la parte inferior del soporte se encuentra empotrada la cámara encargada de captar las imágenes de los cordones.

Cámara digital: MiniVID

La cámara digital que se consideró apropiada para el proyecto fue la denominada MiniVID del fabricante LW Scientific (figura 2.8). Esta cámara nos permite capturar imágenes a 3 MP en línea, con interfaz USB 2.0 directa a la computadora. La capacidad de alta resolución en línea resultaba crítica, debido a que el algoritmo de reconocimiento está diseñado para trabajar correctamente con esa mínima cantidad de información.

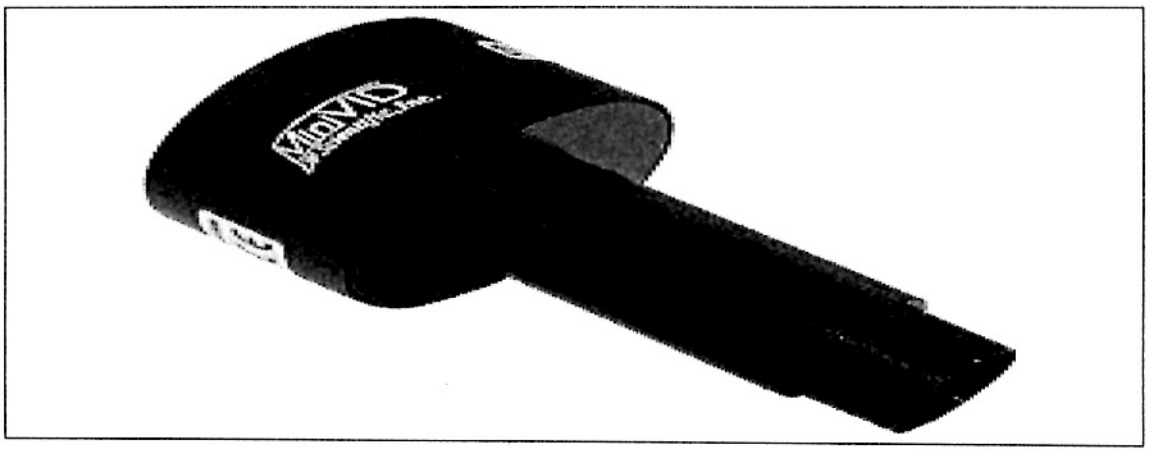


Figura 2.8: Cámara digital MiniVID

En la tabla 2.3, se muestran las características principales del mencionado dispositivo:

Tabla 2.3: Características principales de la cámara MiniVID

Características principales	**Definición**
Modelo	LW Scientific's MiniVID USB
Sensor de imágenes	3.1 MP CMOS chip
Máx. Píxeles	2048x1536 Píxeles
Interface	USB 2.0
Exposición	Automática/ Manual
Formato de imágenes guardadas	BMP, JPG
Requerimientos de sistema	Windows 98, XP, Vista

Adicionalmente, esta cámara incluye una lente ocular de 10X y encaja directamente en el tubo del microscopio, por cuanto resulta relativamente simple ubicarla y capturar imágenes con la computadora, con el soporte del software apropiado (este punto se tocará en la sección de software). Una limitación con relación al software de la cámara es que su uso está restringido a sistemas tipo Windows, por no haberse desarrollado un driver para sistemas tipo Linux.

2.1.5. Circuitos electrónicos de control y potencia

En esta sección se detallan los componentes de hardware necesarios para el control de movimiento de los motores a pasos e iluminación de la lámpara.

1. Tarjeta de Adquisición de Datos: LabJack U3-LV

Para realizar la interfaz entre la computadora y el mundo externo se decidió utilizar una tarjeta de adquisición de datos (figura 2.9) modelo U3-LV de la marca LabJack [16].

Esta tarjeta DAQ de bajo costo (aproximadamente US$ 100) nos proporciona la suficiente cantidad de canales digitales de entrada/ salida para implementar el control secuencial de los motores a paso y recibir las señales de los interruptores de fin de carrera de los posicionadores.

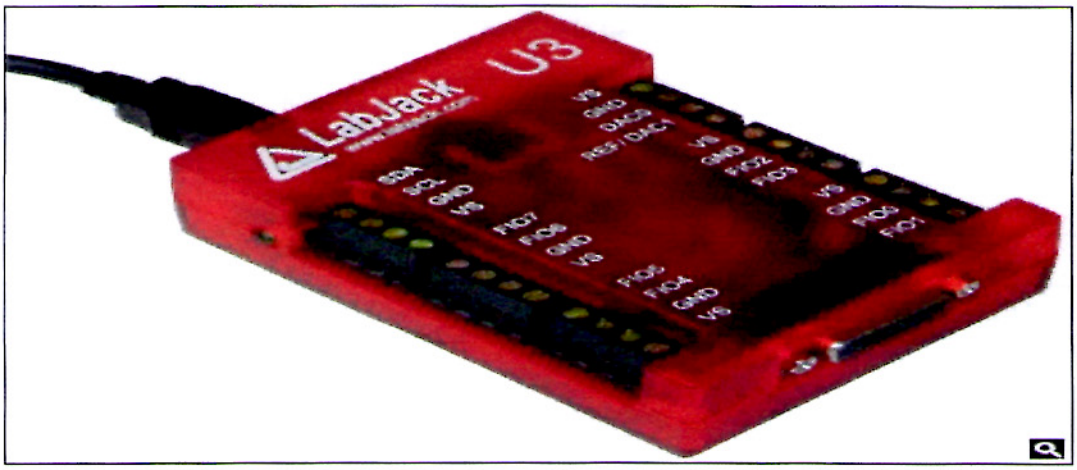


Figura 2.9: Tarjeta de adquisición LabJack U3-LV

En la tabla 2.4, se muestran algunas de sus características principales:

Tabla 2.4: Características principales de la tarjeta LabJack U3-LV

Características principales	Definición
Modelo	LabJack's U3-LV
Interfaz	USB 2.0
Canales digitales I/O	20
Canales Configurables	Entrada, Salida alta, Salida baja
Resistencias pull-up	Entrada: 100 KΩ, Salida: 550 Ω
Librerías	MATLAB, LabVIEW, C
Requerimientos de Sistema	Windows, Linux

Para el control de movimiento en una dirección se requieren cuatro líneas digitales para cada uno de los motores a paso y dos canales adicionales para lectura de fines de carrera para cada uno de los posicionadores. De este modo, con dieciocho líneas digitales se tiene control sobre las tres direcciones XYZ. La facilidad que nos brinda la tarjeta nos permite configurar los canales de entrada/ salida. La configuración empleada se muestra en la tabla 2.5.

Por construcción, las líneas FIO se encuentran sobre los terminales con tornillos en la tarjeta, mientras que los 8 canales EIO y los 4 de CIO aparecen en un conector DB15 en un extremo del dispositivo (ver figura 2.9). Por facilidad de conexión se decidió aprovechar la conexión DB15 para enviar señales a las tarjetas de potencia y dejar los interruptores de fin de carrera para ser conectados directamente con los tornillos.

Tabla 2.5: Configuración empleada para la tarjeta LabJack U3-LV

Canal Digital	Nombre	Configuración	Función
0	FIO0	Entrada	Fin de carrera -Z
1	FIO1	Entrada	Fin de carrera +Z
2	FIO2	Entrada	Fin de carrera –Y
3	FIO3	Entrada	Fin de carrera +Y
4	FIO4	Entrada	Fin de carrera –X
5	FIO5	Entrada	Fin de carrera +X
6	FIO6	Entrada	No usado
7	FIO7	Entrada	No usado
8	EIO0	Salida	Motor Z, Bobina 1
9	EIO1	Salida	Motor Z, Bobina 1
10	EIO2	Salida	Motor Z, Bobina 2
11	EIO3	Salida	Motor Z, Bobina 2
12	EIO4	Salida	Motor Y, Bobina 1
13	EIO5	Salida	Motor Y, Bobina 1
14	EIO6	Salida	Motor Y, Bobina 2
15	EIO7	Salida	Motor Y, Bobina 2
16	CIO0	Salida	Motor X, Bobina 1
17	CIO1	Salida	Motor X, Bobina 1
18	CIO2	Salida	Motor X, Bobina 2
19	CIO3	Salida	Motor X, Bobina 2

2. Tarjeta de Potencia: H-Bridge ET-SMCC V2.0

Con respecto a la parte de potencia, se empleó puentes H para manejar a los motores. Una tarjeta DAQ no suministra suficiente corriente como para mover a los motores, por lo que se precisa contar con una etapa de potencia a la salida. El término puente H se deriva de la típica representación gráfica de este circuito. Los puentes H están compuestos por cuatro switches de estado sólido (MOSFET), los cuales al ser polarizados correctamente, permiten cambiar el sentido de giro de los motores. La secuencia de polarización de switches se realiza a través de los canales digitales de la tarjeta DAQ configurados como salida.

Para el proyecto se compraron tarjetas de la marca ETT, modelo: ET-SMCC v2.0, las cuales están basadas en el circuito integrado H-Bridge L298N e incluyen la electrónica necesaria para proteger a los motores. Cada una de estas tarjetas cuenta con dos puentes H, lo cual las hace ideales para controlar motores a paso bipolares. En la tabla 2.6 se muestran algunas características de la mencionada tarjeta y en la figura 2.10 un esquema simplificado de conexionado de componentes.

Tabla 2.6: Características de la tarjeta de potencia ET-SMCC v2.0

Características principales	**Valor**
Modelo	ET-SMCC v2.0
Canales de Control	2 Canales
Frecuencia de paso	Máximo 40KHz
Tipo de motor	Motor a paso bipolar, motores DC
Corriente de salida	4 A por fase
Lógica de control de entrada	Niveles TTL estándar
Alimentación	Para lógica: 5 VDC / 20 mA Para motor: máx. 50 VDC / 4 A

Con respecto al control de iluminación de la lámpara, se puede mencionar que se empleó un circuito simple basado en una configuración Darlington de transistores para conseguir aproximadamente 4 A para la lámpara de manera regulable con un potenciómetro. Esta configuración de transistores en cascada, permite disparar un transistor de potencia con uno que maneja una corriente mucho menor. El transistor empleado para la parte de baja corriente fue 2N2221A, el cual produce hasta 800 mA de corriente de salida, mientras que para la parte de alta corriente se empleó 2N3055, el cual proporciona hasta 15 A de corriente de colector.

Finalmente, cabe señalar que se preparó un armazón diseñado especialmente para los circuitos de control y potencia de los motores, considerando además una tarjeta impresa de bornera para facilitar el cableado, conectores apropiados y un ventilador para que circule el aire y evitar sobrecalentamiento. De la misma manera, se consideró una fuente diseñada para el equipo (basada en una fuente de computadora), la cual proporciona los niveles de voltaje apropiados para los dispositivos empleados en el equipo. La fuente de potencia incluye un switch de habilitación general, el circuito de control de iluminación y ventilación respectiva.

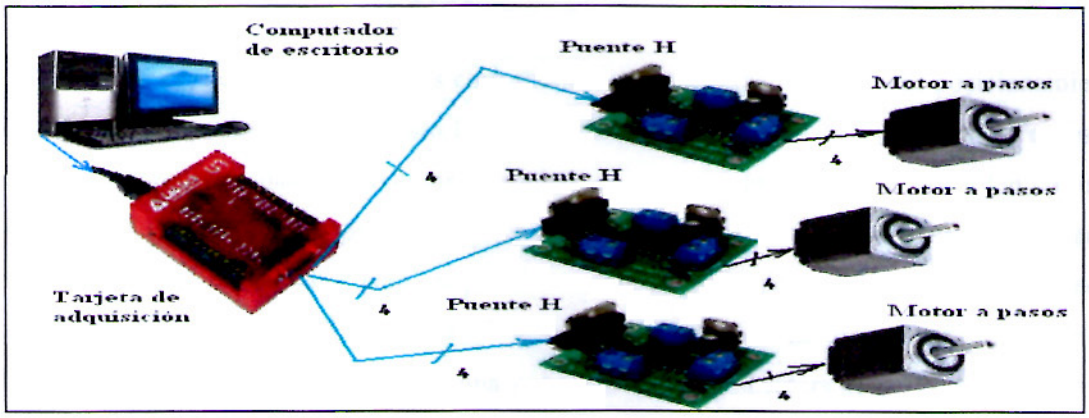


Figura 2.10: Configuración empleada de hardware para control de motores

2.2. Software de Control

En esta sección se consideran todos los programas empleados (figura 2.11) para realizar el control de movimiento con motores a paso, programar trayectorias de barrido de pozos, captura de imágenes, enfoque automático, virtualización para diagnosticar y presentación de resultados. El lenguaje de programación empleado para el diseño del control del equipo fue MATLAB (v7.0), el cual, en su versión básica, nos brinda facilidad para generar código de pruebas. La implementación final del controlador fue realizada en Visual C++ (v6.0) y haciendo gran uso de las denominadas Microsoft Foundation Classes (MFC) para crear la interfaz gráfica de usuario (GUI). El empleo de software en plataforma Windows, tiene que ver con la limitación encontrada en el driver de la cámara MiniVID como se explicó en la sección de captura de imágenes. El programa principal en GUI se denomina "MODS_Plate_Reader_X.dsp". Este archivo de proyecto en formato .dsp en VC++ incorpora todos los controles, funciones y procedimientos desarrollados en este proyecto para lectura de placas MODS.

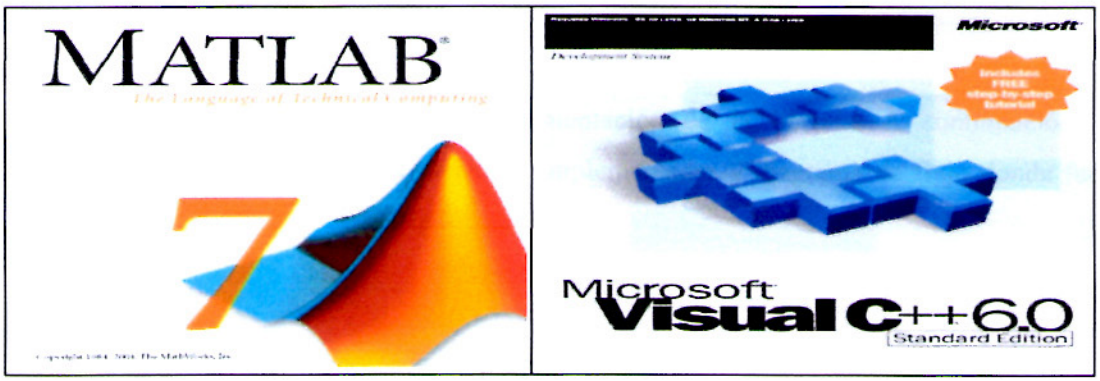


Figura 2.11: Lenguajes de programación empleados en el proyecto

2.2.1. Control de motores a paso

Como se explicó en las secciones correspondientes a sistema mecánico de posicionamiento (sección 2.1.2) y circuitos de control y potencia (sección 2.1.5), los posicionadores están basados en motores a pasos y su control requiere generar pulsos digitales en el orden correcto para conseguir el sentido de giro y el avance deseado. Estos pulsos de control se escriben en la tarjeta de adquisición de datos para ser posteriormente amplificados por el circuito de potencia. La librería que nos permite el control a bajo nivel de los motores es la denominada "Stepper_Motors", la cual consta de una cabecera (Stepper_Motors.h) y un archivo fuente (Stepper_Motors.cpp). Esta librería a su vez, emplea una librería en lenguaje C, proporcionada por el fabricante LabJack para poder utilizar sus tarjetas DAQ [16].

1. **Stepper_Motors.h:** Cabecera de librería de control de motores, prepara al compilador con los tipos de las variables de entrada y salida, así como también declara el prototipo de las funciones que soporta. En la tabla 2.7 se presentan las funciones de control de motores.

Tabla 2.7: Funciones implementadas para control de motores

void ***Read_Sensors*** *(int* **n**, int* **spin**, LJ_HANDLE* **lngHandle**)*
void ***Write_Instructions*** *(int* **x**, int* **n**, LJ_HANDLE* **lngHandle**)*
void ***Search_1D_Array*** *(int* **vector**, int* **x**, int* **r**)*
void ***ErrorHandler*** *(LJ_ERROR* **E1**, long* **E2**, char* **err**)*
void ***Stepper_Motor*** *(int* **steps**, int* **n**, int* **x**, LJ_HANDLE* **lngHandle**)*

El significado de las variables mostradas se explicará en la sección de implementación de la librería de control de motores.

2. **Stepper_Motors.cpp:** Contiene la implementación de la librería. A continuación, se presenta una descripción de las funciones implementadas y diagrama de flujo donde fuera necesario.

a) **Read_Sensors**

Descripción: Esta función realiza la lectura de canales digitales de entrada de la tarjeta DAQ. Recordemos de la tabla 2.5 que existen 6 canales asociados con interruptores de fin de carrera.

La lectura de estos sensores se emplea en dos situaciones: determinación de posición cero de referencia del equipo (explicado posteriormente en "Initial_Position") y detección de falla mecánica (explicado en "Error_Handler").

Argumentos: *n* es un puntero con relación al motor que deseamos mover. A saber, *n=1 indica motor en eje X, *n=2 motor en Y, *n=3 motor en Z. *spin* es un puntero que indica el sentido de giro de los motores: *spin=1 indica dirección positiva, *spin=-1 dirección negativa. Finalmente, *lngHandle* es un puntero relacionado con un identificador para acceder a la tarjeta DAQ. El tipo definido por LJ_HANDLE es equivalente a long.

Funcionamiento: Dada la selección de motor y dirección de avance, se lee el fin de carrera hacia el cual el posicionador se dirige, como se muestra en la tabla 2.8.

Tabla 2.8: Lectura de fines de carrera

Motor seleccionado (*n)	**Sentido de avance (*spin)**	**Acción a llevarse a cabo** Hace uso de librería de LabJack: labjackud.lib
1	1	Petición de Lectura de canal DAQ FIO5
1	-1	Petición de Lectura de canal DAQ FIO4
2	1	Petición de Lectura de canal DAQ FIO3
2	-1	Petición de Lectura de canal DAQ FIO2
3	1	Petición de Lectura de canal DAQ FIO1
3	-1	Petición de Lectura de canal DAQ FIO0

Las peticiones de lectura de canales discretos se realizan por medio de una instrucción de bajo nivel denominada *AddRequest (*lngHandle, LJ_ioGET_DIGITAL_BIT, p, 0, 0, 0)* donde p representa el bit que deseamos leer (revisar tabla 2.8).

b) Write_Instructions

Descripción: Esta función realiza la escritura sobre canales digitales de salida de la tarjeta DAQ. Recordemos de la tabla 2.5 que existen 12 canales asociados con líneas de control de motores.

Argumentos: *x* es un puntero que tiene relación con el valor que deseamos escribir por los puertos de salida. Este valor entero se convierte internamente a binario y se envía por la tarjeta DAQ. El resto de parámetros son los mismos que para lectura de sensores.

Funcionamiento: Dada la selección de motor, se escribe por los canales correspondientes la secuencia generada por "Stepper_Motor" (esto se explica más adelante) como se indica en la tabla 2.5. La petición de escritura de puertos se realiza mediante la instrucción *AddRequest (*lngHandle, LJ_ioPUT_DIGITAL_PORT, q,*x, 4, 0),* donde q se relaciona con el motor que se desea mover. A saber, q=16 indica eje X, q=12 indica eje Y, q=8 indica eje Z.

c) Search_1D_Array

Descripción: Este pequeño código realiza la búsqueda de un elemento en un array 1D.

Argumentos: *vector* indica el array 1D, **x* es el elemento que se busca en el array, **r* es el índice del elemento en el array.

Funcionamiento: Se incrementa un contador en un lazo *while* hasta que ocurre alguna coincidencia de algún elemento del array con el valor que estamos considerando de entrada. Debe aclararse que por diseño de software, siempre existirá una coincidencia.

d) Error_Handler

Descripción: Este programa genera un código de errores. Se consideran fallas en tarjeta DAQ, accionamiento inesperado de fines de carrera y fallo de controles internos.

Argumentos: *E1* es un puntero con relación a fallas detectadas en la tarjeta DAQ. El tipo LJ_ERROR es equivalente a long. La palabra reservada LJE_NOERROR hace mención al número 0, indicador de no existencia de falla. *E2* es un puntero con relación a fallas de tipo mecánicas y de control no superado. Finalmente, *err* es un array de char donde se guarda el mensaje de error.

Funcionamiento: En primera instancia hace uso de la función *ErrorToString (*E1, err)* (proporcionada por el fabricante), la cual nos devuelve en un array la causa de la falla detectada. La lista de posibles errores reportados se encuentra en la guía de referencia de LabJack y suman 102 casos entre los cuales podemos referir errores de configuración, de temporizadores, de comunicación, etc. En segundo lugar, en el programa "XY_Move" y "Z_Move" (explicados más adelante) se asignan codificaciones a errores de procedimiento encontrados durante la ejecución de lectura de placas.

Tabla 2.9: Codificación de fallas encontradas

Indicador de falla	Mensaje de falla
*E1 != LJE_NOERROR	ErrorToString (*E1,err)
*E2==103	Limit Switch -X activated
*E2==104	Limit Switch +X activated
*E2==105	Limit Switch -Y activated
*E2==106	Limit Switch +Y activated
*E2==107	Limit Switch -Z activated
*E2==108	Limit Switch +Z activated
*E2==109	Serious fault: Positive control not completed successfully
*E2==110	Serious fault: Negative control not completed successfully

e) Stepper_Motor

Descripción: Este código genera dinámicamente la señal de control para los motores.

Argumentos: *steps* es un puntero con relación al número de pasos que deseamos realizar, el resto de parámetros ha sido explicado con anterioridad.

Funcionamiento: Para el control de motores a paso se requiere generar por software la siguiente secuencia de pulsos: [0101, 0110, 0110, 0101] con un tiempo de retardo de acuerdo a la velocidad deseada. Las señales se envían por la tarjeta de adquisición, por las cuatro líneas de control asignadas a cada motor, de manera simultánea y cíclica, para conseguir dirigir el motor en dirección positiva. Si se desea invertir el giro, la secuencia debe generarse en sentido inverso. El algoritmo recibe como entrada la señal actual de control, así como el motor que se desea mover y su dirección de giro. Con estos parámetros y mediante el empleo de la función *Search_1D_Array* podemos determinar la siguiente señal de control a aplicarse. Finalmente, dicha señal se envía a la tarjeta DAQ y a la tarjeta de potencia.

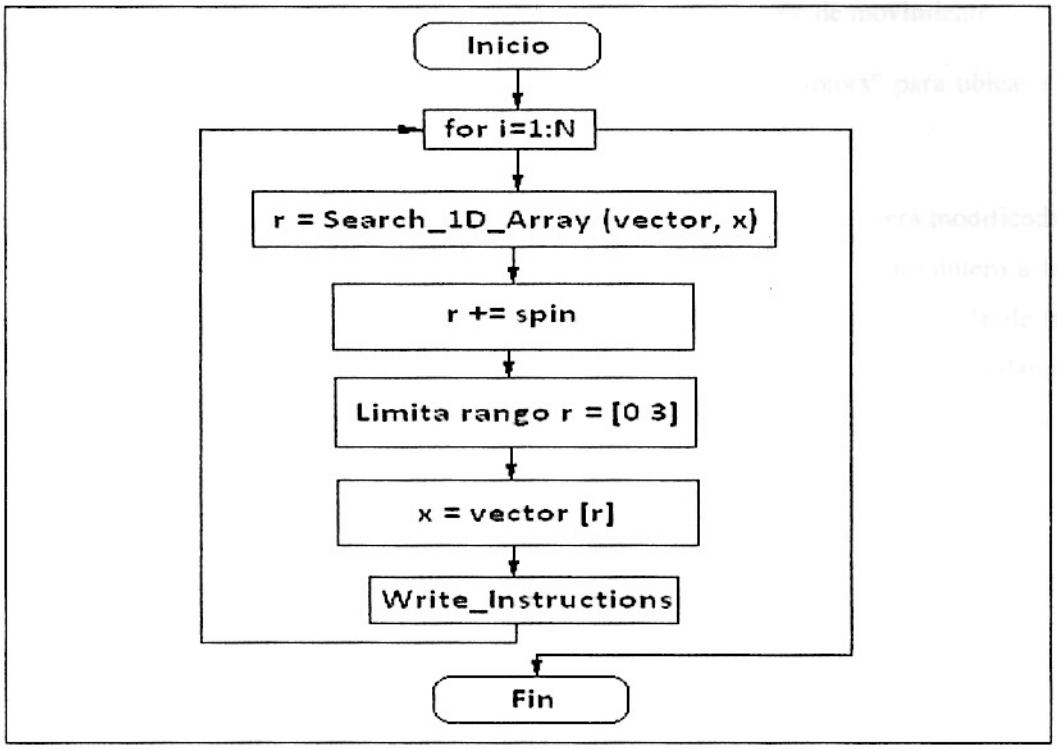


Figura 2.12: Diagrama de flujo de Stepper_Motor. Se puede interpretar que el código ejecuta un lazo *for* con número de cuentas igual al número de pasos que se desea dar. Considerando el sentido de giro en **p_spin* (spin) se puede actualizar el valor de **p_r* (r) que finalmente significa una nueva señal de control. Limita rango se emplea para delimitar los valores de **p_r* a aquellos que tengan sentido en el proceso. Finalmente, se escriben los datos en tarjeta de adquisición.

2.2.2. Control de movimiento

Hasta el momento se ha cubierto los siguientes puntos con relación al control de motores: naturaleza y funcionamiento de motores a paso, señales de control, así como amplificadores para transformar la señal de control en una de alta corriente para el actuador, capaz de producir el giro de los motores. El siguiente punto a tocar corresponde a las rutinas que hacen uso de "Stepper_Motors" para conseguir movimientos coordinados sobre el plano XY, así como en dirección de enfoque Z.

1. XY_Move.h: Cabecera de librería de movimiento de motores en el plano XY.

void ***XY_Move****(int* x, double* X, double* Xi, int* y, double* Y, double* Yi, double* Vxy, double* PV, double* AV, LJ_HANDLE* lngHandle, LJ_ERROR* E1, long* E2)*

2. XY_Move.cpp: A continuación se presenta el desarrollo de esta función de movimiento.

Descripción: Esta función emplea las señales generadas por "Stepper_Motors" para ubicar el microscopio en cualquier posición XY.

Argumentos: *x* es un puntero a la señal de control actual. Esta señal de control será modificada constantemente para producir el movimiento de los motores. La variable *X* es un puntero a la dirección de la coordenada X que se desea alcanzar, mientras que *Xi* apunta a la dirección de la coordenada X actual o de inicio. Las tres variables siguientes tienen significados similares asociados a la dirección Y. El parámetro *Vxy* nos da información concerniente a la velocidad máxima alcanzable en el plano XY. Por otro lado, *PV* es un puntero, cuyo contenido nos indica el parámetro mecánico "Pasos_Por_Vuelta". Este parámetro toma el valor *PV= 200 dadas las características de los motores. El parámetro *AV* nos indica el valor de "Avance_Por_Vuelta" en unidades de milímetros. En nuestro caso, como característica de los posicionadores se tiene que *AV = 0.635 (ó 0.025" según tabla 2.1). Con estas dos cantidades es posible convertir movimiento angular en lineal con gran precisión. Notar que todas las distancias se consideran en milímetros.

Funcionamiento: Para este programa se hace una distinción entre dos posibles casos, movimiento en diagonal en el plano XY y el movimiento a lo largo de una dirección, X o Y. La distinción se justifica en el sentido que para el movimiento en diagonal es necesario un procedimiento de sincronización para que ambos motores lleguen al ángulo deseado (set-point) al mismo tiempo, a diferencia de un movimiento unidimensional X o Y, el cual resulta bastante más sencillo.

a) Movimiento en diagonal. A partir de las posiciones de inicio y destino se calcula la distancia total de movimiento y la dirección de avance. Posteriormente, se calcula la cantidad de pasos discretos requeridos para cubrir esta distancia mediante un proceso de cuantización, dado que este debe ser un número entero. Llamaremos *dXP y *dYP a la cantidad de pasos requeridos en dirección X e Y respectivamente. En este punto, debemos determinar el mínimo entre estas dos cantidades. Sin pérdida de generalidad, asumamos para el presente análisis que *dXP<*dYP. En este caso, por cada paso en dirección X, se producirá una cantidad mayor o igual en dirección Y a fin de que ambos cumplan su destino final al mismo tiempo. Lo que haremos a continuación, será agrupar los pasos en grupos de a diez a fin de ahorrar recursos en llamadas a la tarjeta DAQ para lectura de interruptores de fin de carrera. Obtenemos la cantidad *Nit, número de iteraciones en un lazo *for*, que representa la cantidad de grupos de diez pasos que se deben realizar (para la dirección de menor recorrido), más un posible offset menor a diez pasos.

Dentro del lazo for se ha implementado internamente un control de velocidades, de tal manera se consiga un perfil trapezoidal de arranque y frenado suave, esto se consigue mediante captura de tiempo usando llamadas a sistema y analizando el valor del iterador del lazo. La velocidad va desde *(Vxy)/2 hasta *Vxy en el primer 10% del recorrido total, se mantiene en *Vxy entre el 10% y el 90% para finalmente decaer linealmente hasta *(Vxy)/2 en el último 10% del recorrido. En la figura 2.13 se muestra el tiempo por iteración como función de la distancia discreta recorrida.

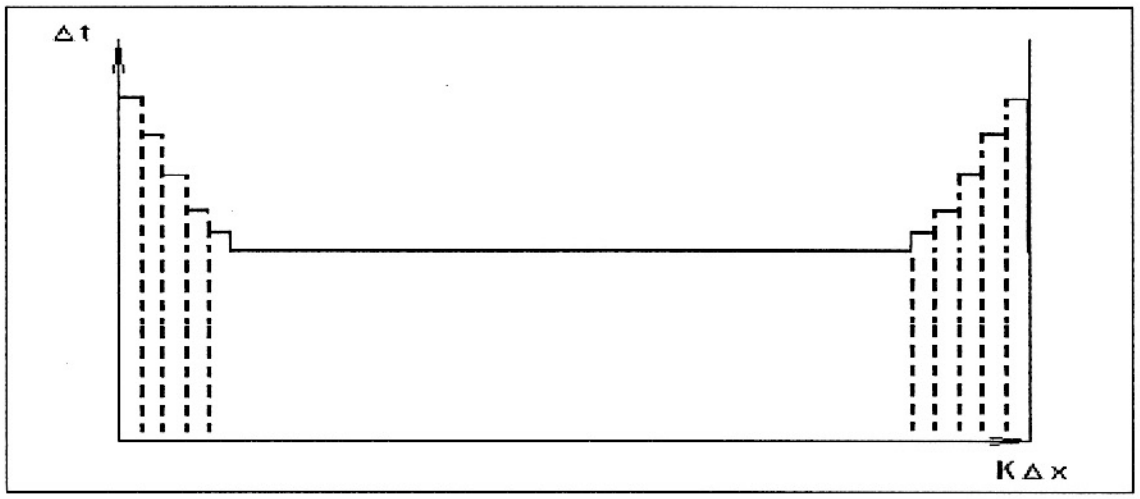


Figura 2.13: Perfil de velocidades trapezoidal para arranque y freno suave

Como se mencionó, para la dirección de menor recorrido se realizan diez pasos por iteración (más un offset, si *dXP no fuera múltiplo de diez). Para la dirección de mayor recorrido se debe determinar el número de pasos correspondiente, este es el proceso de sincronización al que hacíamos referencia al inicio. Básicamente, se debe tomar en consideración la relación entre la distancia que resta por cubrir para la dirección de mayor recorrido y la de menor recorrido, esto es *(temp_dYP)/*(temp_dXP) actualizado en cada iteración, para conseguir el factor de proporcionalidad y a partir de ahí determinar la cantidad de pasos por iteración.

Finalmente, luego de determinar la cantidad de pasos que cada motor debe realizar en la presente iteración, se produce una llamada a la función Stepper_Motor.cpp (considerando el número de pasos variable en cada dirección XY), Read_Sensors.cpp, captura de estado de sensores (luego de ejecutar un avance de diez pasos), análisis de posible fallo mecánico, actualización de variables (distancias por recorrer en ambas direcciones) y retardo de tiempo para el perfil de velocidades. Las variables de posición *X e *Y se actualizan continuamente durante todo el proceso, de tal manera que al culminar el movimiento se tiene la información de las coordenadas actuales reales del microscopio.

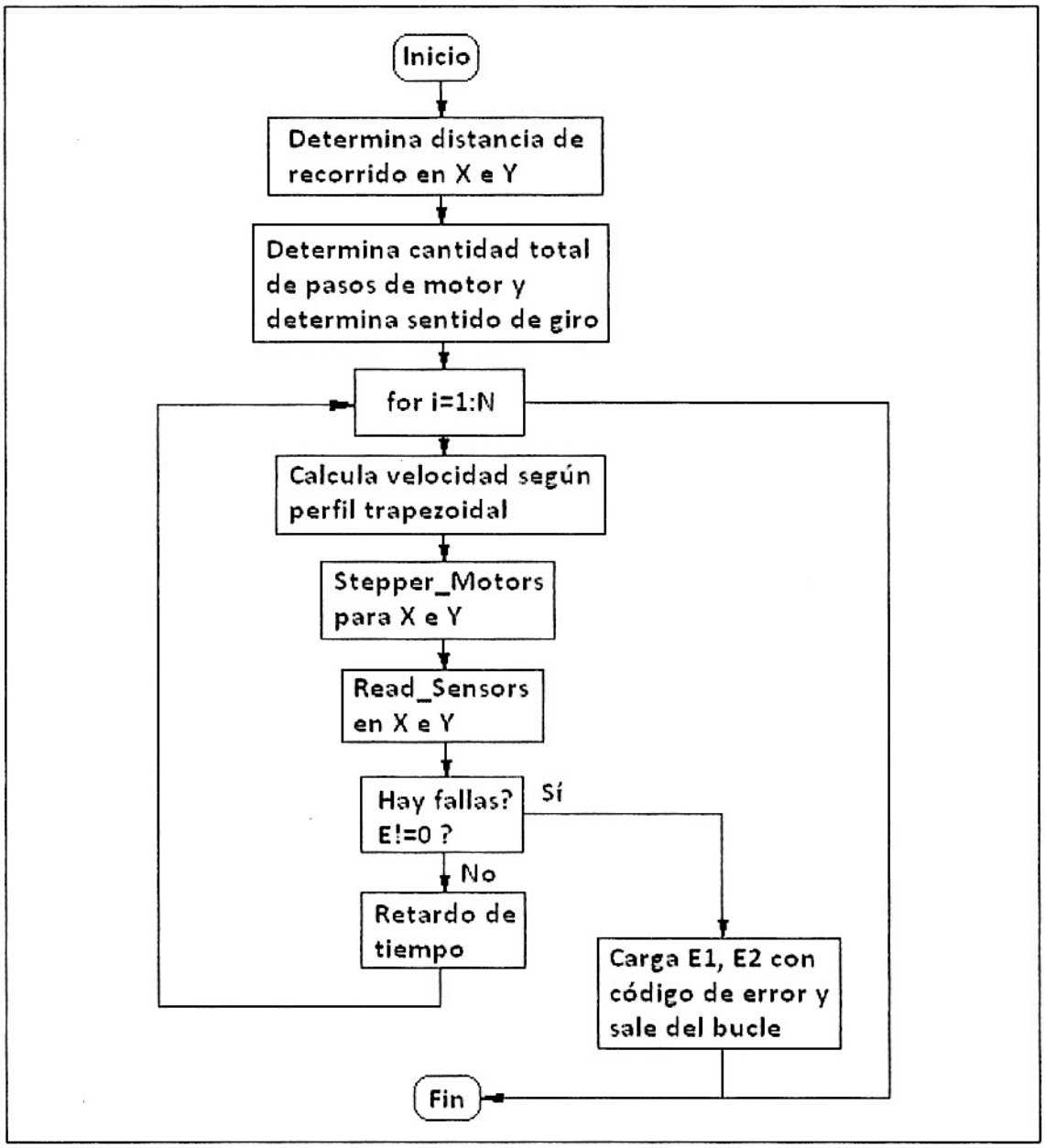


Figura 2.14: Diagrama de flujo de XY_Move en tipo movimiento diagonal en plano XY

b) Movimiento unidireccional. El movimiento en el caso unidireccional X o Y es bastante más sencillo que el de diagonales arbitrarias, dado que no requiere ningún tipo de sincronización. En principio, comprende los mismos pasos fundamentales del algoritmo para movimiento en diagonales, esto es: determinación de distancia total a recorrer en X o Y, cuantización en pasos, encontrar la cantidad de iteraciones y ejecutar Stepper_Motors.cpp, así como el proceso de lectura con Read_Sensors.cpp, retardo de tiempo para conseguir el perfil trapezoidal de velocidades y actualización de coordenadas.

Dado que la secuencia básica es la misma, realmente se puede utilizar el mismo diagrama de la figura 2.14 para explicar el funcionamiento en este caso, teniendo en mente las consideraciones de que dentro del lazo *for* la cantidad de pasos será diez por defecto (salvo la última iteración de offset) y que no se requiere actualizar variables temporales al final del procedimiento.

3. Z_Move.h: Cabecera de librería de movimiento de motor en dirección Z. Sólo se encuentra una función implementada:

void ***Z_Move****(int* z, double* Z, double* Zi, double* Vz, double* PV, double* AV, LJ_HANDLE* lngHandle, LJ_ERROR* E1, long* E2)*

4. Z_Move.cpp: A continuación se presenta el desarrollo de esta función de movimiento.

Descripción: Esta función emplea las señales generadas por "Stepper_Motors" para ubicar el microscopio en cualquier posición Z.

Argumentos: *z* es un puntero a la señal de control actual. Esta señal de control se verá alterada tras el movimiento del motor, como consecuencia de esto debe ser actualizada constantemente. Al emplear direccionamiento por referencia la tarea de mantener **z* actualizada es bastante simple. La variable *Z* es un puntero a la coordenada Z que se desea alcanzar, mientras que *Zi* apunta a la dirección de la coordenada Z actual o de inicio. El resto de parámetros ya han sido explicados con anterioridad.

Funcionamiento: Este código trabaja de la misma manera que XY_Move.cpp en el caso en que el movimiento es unidireccional, sólo que este bloque está dedicado de manera especial para Z. Una particularidad debido al recorrido típico en dirección Z (del orden de fracción de milímetros, excepto en "Initial_Position.cpp"), es que en este caso los pasos se ejecutan uno por uno y no en grupos de diez como en XY_Move.cpp.

2.2.3. Programación de trayectorias

Hasta aquí se ha descrito bloques funcionales de movimiento, los cuales sólo requieren que el operador indique la posición deseada en XYZ para luego ellos encargarse automáticamente de todas las tareas asociadas de bajo nivel. A partir de aquí, las labores de bajo nivel son transparentes para el usuario. Es momento de subir un escalón más y describir los bloques funcionales que se encargan de establecer los set-points para las funciones XY_Move.cpp y Z_Move.cpp y de esta manera determinar dinámicamente la trayectoria de lectura de placas.

1. XY_Global.h: Cabecera de librería auxiliar de programación de trayectorias en el plano XY.

void XY_Global (double **X**, double* **Y**, double* **X0**, double* **Y0**, double* **PV**, double* **AV**, double* **dX**, double* **dY**, int **p0**, int **p1**, int* **Tr**)*

2. XY_Global.cpp: A continuación se presenta el desarrollo de esta función de asignación de referencias entre pozos.

Descripción: Esta función se encarga de apuntar a los centros de los pozos de una placa MODS. Se hace la observación de que esta función no produce un movimiento real entre centros de pozos sino que más bien nos ayuda a mantener un orden de evaluación. Esta función es de tipo auxiliar y es utilizada para separar el problema de recorrido global (entre pozos), del recorrido local (dentro de un pozo). Si el término cabe, podríamos decir que se limita a realizar un recorrido "virtual" del microscopio. El recorrido real (observable) considera puntos estratégicos de lectura, lo cual será explicado con más detalle en la sección de procesos de lectura.

Argumentos: *X* e *Y* son punteros asociados a la posición actual del microscopio. *X0* e *Y0* brindan información de la posición del centro del primer pozo; esto es, aquél con código 1A (figura 2.15). Por otro lado, *dX* es un puntero referente a la distancia entre centros de pozos en dirección X, mientras que *dY* está referido a la distancia entre pozos en dirección Y. Los contadores *p0* y *p1* indican el pozo que se encuentra analizando en un momento previo y en el momento actual, respectivamente. Finalmente, *Tr* es un puntero a array con contenido de los índices de los pozos que deben ser analizados en un cierto proceso de lectura.

Funcionamiento: En primer lugar, mostraremos la codificación empleada para ubicar los centros de los pozos en una placa MODS. En pocas palabras, la codificación empleada y el significado del contenido de *Tr extendido sobre su tamaño (debido a que es un array). Por simplicidad, cada vez que nos refiramos a *Tr en realidad nos estamos refiriendo a todo el array.

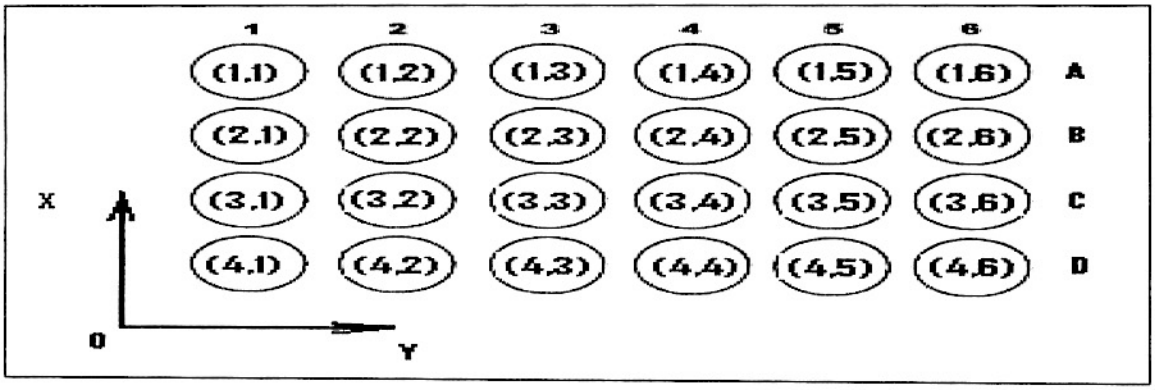


Figura 2.15: Codificación de índices para los pozos en una placa MODS

Como se describió en la sección 1.1.3, el barrido para un paciente se produce sobre una cierta columna (en dirección -X) y entre filas, para distintos pacientes (en dirección +Y). El contenido de *Tr depende del proceso de lectura y nos brinda información de los centros de los pozos, así como el orden secuencial en el cual deben ser recorridos (detalle en sección de procesos de lectura). Recuérdese que los controles internos son las primeras columnas por analizar, por lo que es de esperarse que, por ejemplo, para lectura de placa MODS con muestras de pacientes, los primeros cuatro elementos tengan un factor de columna constante igual a tres (tercera columna de control negativo) y las filas sufran un incremento de uno por vez en cada iteración. Recordemos la explicación en la parte descriptiva: esta función se encarga de resolver el problema de ubicar los centros de los pozos de la manera más natural posible. En el recorrido macro entre pozos, es patente que la unidad en la que se puede medir los recorridos de manera natural es la distancia entre centros, por lo cual se emplean los índices de los pozos (en lugar de distancia en milímetros), para denotar un cierto pozo en particular, como muestra la figura 2.15. Notar adicionalmente que con este sistema, resulta sencillo determinar la posición de cualquier pozo con respecto al primero, de índice (1,1). En XY_Global.cpp se hace una clara distinción entre la primera posición de lectura y el resto de ellas, debido a la forma en que se hace referencia a los pozos y al empleo del array *Tr. El primer posicionamiento de lectura tiene como tarea llevar al microscopio desde su posición actual (*X, *Y) cualquiera que sea, (esto es, totalmente independiente de *Tr), hasta el centro del primer pozo de lectura, contenido de la primera fila de *Tr más un cierto offset que depende de los parámetros de entrada *X0 e *Y0 (coordenadas absolutas en milímetros). Posteriormente, una vez que se ha conseguido llevar el sistema a la primera posición de lectura, el resto de posiciones resulta definido fácilmente de manera relativa al centro del primer pozo de la placa. Las variables de posición *X e *Y se actualizan continuamente durante todo el proceso, independientemente de si se trata de trayectoria virtual o real. En el caso virtual, llevan la información del siguiente punto de tipo virtual y si es real llevan la información de las coordenadas reales del microscopio en el tiempo.

3. XY_Local.h: Cabecera de librería auxiliar de programación de trayectorias en el plano XY.

void ***XY_Local*** *(double* **X**, *double* **Y**, *double* **PV**, *double* **AV** ,*double* **Dm**, *int* **Nx**, *int* **Ny**, *int* **A**, *int* **p2**)*

4. XY_Local.cpp: A continuación se presenta el desarrollo de esta función de asignación de referencias dentro de un pozo.

Descripción: Esta función se encarga de apuntar a posiciones al interior de un pozo de una placa MODS.

Similarmente al caso anterior, esta función tampoco produce un movimiento real dentro de un pozo, sino que se limita a ayudarnos a resolver el problema del recorrido local dentro de un pozo de manera ordenada.

Argumentos: **X e *Y* representan la posición virtual actual del microscopio. **Dm* es un puntero referente al diámetro de un pozo. **Nx* y **Ny* bridan información del número de particiones necesarias para cubrir todo un pozo, dada el área limitada que cubre la cámara digital. **A* es un puntero a array, nos brinda información relativa al cambio de sentido de avance de los motores dentro de un pozo. Finalmente, *p2* nos indica el número de iteración en la que nos encontramos actualmente en el barrido de un pozo en particular.

Funcionamiento: Si bien es cierto el recorrido macro entre pozos depende mucho del tipo de lectura que se realiza, el recorrido dentro de un pozo se diseñó de una manera especial, de tal manera que se optimice la lectura del pozo. En la figura 2.16 se muestra un ejemplo de barrido local considerando *Nx=5 y *Ny=5, para un total de 25 particiones por pozo. Este tipo de recorrido virtual será denominado en adelante como "trayectoria de caracol". Como se puede observar tiene la particularidad de dar prioridad de lectura a los bordes antes que a posiciones céntricas dentro del pozo.

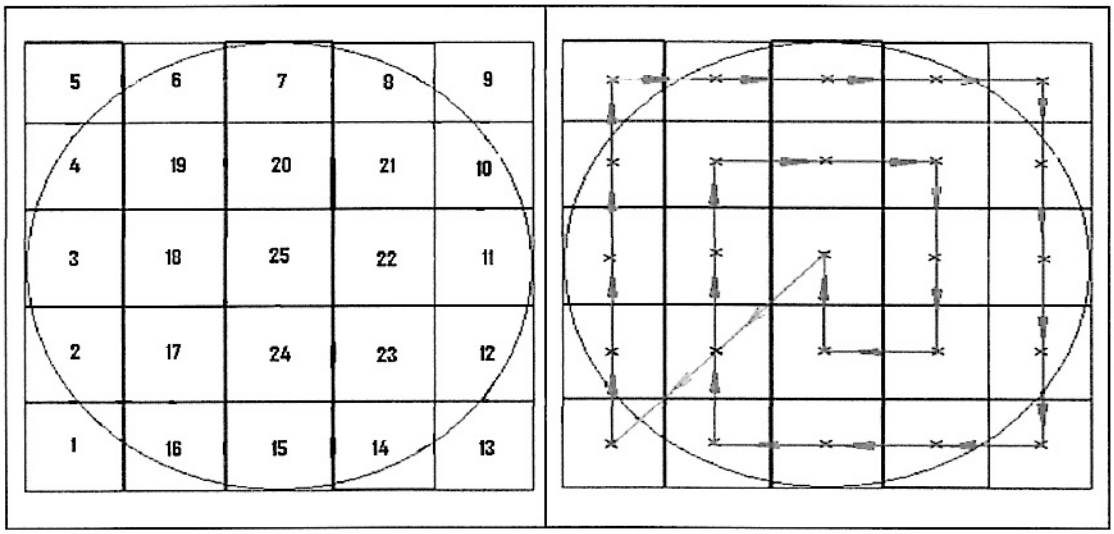


Figura 2.16: Codificación de índices para un pozo particular y recorrido dentro del mismo

Como se mencionó, el recorrido en caracol optimiza el proceso de lectura debido a la siguiente característica de crecimiento de cordones: la manera más rápida de encontrar cordones aislados, es comenzar buscando por los bordes de un pozo. Estos cordones aislados resultan de utilidad para el software de reconocimiento, no así los conglomerados que resultan del crecimiento masivo en la zona céntrica de los pozos.

En la figura 2.17, se muestra el recorrido real del microscopio en un pozo cualquiera, para un caso de resultado negativo, donde se ha considerado 16x12 = 192 particiones, a saber, el número real de particiones implementado por defecto para lectura. En este momento se debe tener un mejor entendimiento del significado de recorrido virtual y real: Tomemos como ejemplo la figura 2.17 en el que se tiene un caso con diagnóstico negativo. Imaginemos que el diagnóstico hubiera sido positivo; esto es, el recorrido resultó interrumpido en algún valor de p2 menor a 192. Para seguir con el argumento, digamos que se interrumpió en p2=10 (similar a la codificación mostrada en las figuras 2.16 y 2.17). Al haberse determinado TB en el pozo, según las reglas del método MODS, ya no es necesario seguir leyendo ese pozo, sino que se debe proseguir con la lectura del siguiente, comenzando con el primer campo que se encuentre dentro del pozo. El proceso completo de posicionamiento para barrido en nuestro ejemplo, se encuentra descrito de la siguiente manera:

1. Se encuentra evidencia de TB en posición p2=10.
2. El recorrido virtual nos regresa al centro del pozo recién evaluado.
3. XY_Global.cpp se encarga de llevarnos virtualmente al centro del siguiente pozo a analizar.
4. XY_Local.cpp determina la primera posición virtual p2=1 dentro del nuevo pozo.
5. Como el campo p2=1 no está dentro del pozo, XY_Local.cpp nos lleva virtualmente al campo p2=2; el cual, para nuestro caso, también se encuentra completamente fuera del pozo. El proceso continúa hasta que se encuentre un campo que contenga en su interior al menos una pequeña porción del pozo. En la figura 2.17, se muestra con una cruz (X) que el primer campo en el interior del pozo válido para proceder con lectura es p2=4.
6. Se realiza el movimiento real desde el pozo anterior con p2=10 hasta el pozo actual con p2=4 mediante una llamada a la función XY_Move.cpp.

En el desarrollo del bloque XY_Local.cpp se distinguen dos casos, similarmente al caso de XY_Global.cpp: el caso inicial en que p2=1 y el resto de casos posibles. Cuando se tiene que p2=1, el programa produce un recorrido virtual en dirección de la diagonal –X, -Y y de magnitud igual al radio del pozo. Para el resto de casos se considera el array *A, el cual contiene "banderas" que indican la dirección de avance (en este caso, la distancia es constante igual a *Dm/Nx) según el valor que va tomando p2. A manera de ilustración para el caso planteado en esta sección: Inicialmente la dirección de barrido es ciertamente +X, sin embargo apenas p2 supere el valor de 16, la dirección de barrido cambia a +Y (con avances de *Dm/Ny).

Esto se repite para los casos en que p2 supera el valor de 27, momento en el cual, la dirección de movimiento cambia a –X y luego cuando p2 supera a 42, cuando la dirección cambia a –Y (ver figura 2.17). Este proceso se repite periódicamente hasta que se encuentre TB o hasta que se alcance la última posición en el interior del pozo. Notar que los movimientos en diagonal son sólo de desplazamiento y no de lectura.

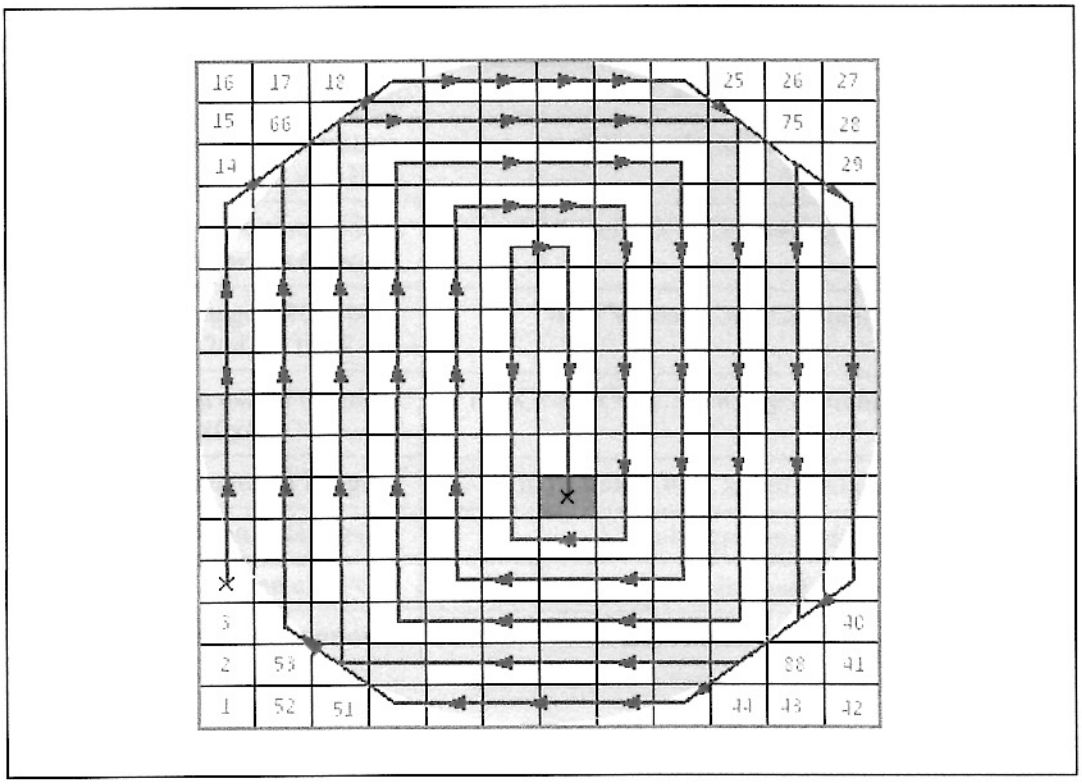


Figura 2.17: Trayectoria de caracol

2.2.4. Captura de imágenes

Una vez que hemos conseguido ubicarnos en posición de lectura en el plano XY, el resto de procedimiento se realiza sobre el eje Z de enfoque. El primer punto a considerar es ciertamente el de captura de imágenes con la cámara digital MiniVID. Con este fin, se decidió utilizar un software ActiveX denominado XVideoOCX [17], el cual nos da la posibilidad de hacer enlace con la cámara. Este software nos brinda una librería en C/C++ sencilla de usar para implementar aplicaciones de captura y procesamiento de imágenes. El procedimiento empleado para realizar la captura de imágenes se muestra a continuación:

Tabla 2.10: Procedimiento de captura de imágenes usando ActiveX XVideoOCX

Paso	Objetivo	Función empleada
1	Crear objeto ActiveX	Se crea directamente en ventana de diálogo
2	Establecer modo video	m_XVideoOCX.XSetInputMode(0)
3	Elegir cámara MiniVID	m_XVideoOCX.XSetVideoDevice(3)
4	Inicia canal de comunicación	m_XVideoOCX.XInit()
5	Habilitar modo Preview	m_XVideoOCX.XSetPreview(TRUE)
6	Mostrar imágenes	m_XVideoOCX.XStart()
7	Obtener buffer de memoria de cámara 2048 x 1536	m_XVideoOCX.XGetImageHandle(24)
8	Crear un buffer para imagen en escala de grises 640x480	m_XVideoOCX.XCreateImageHandle(480,640,8)
9	Captura imagen RGB con escala 2048 x 1536	m_XVideoOCX.XCapture(handle24o)
10	Convertir a escala de grises 640x480	m_XVideoOCX.XToGrayScale(handle24s,handle8)
11	Obtener puntero a imagen	m_XVideoOCX.XGetDataPointer(handle8)
12	Trabajar con la imagen	Calcular gradiente de Brenner
13	Guardar imagen en JPG	m_XVideoOCX.XSaveImage(filename,handle24o,1,100)
14	Liberar buffer de memoria	m_XVideoOCX.XReleaseImageHandle(handle24o)

2.2.5. Enfoque automático

Las bases del procedimiento de autoenfoque fueron establecidas en la sección teórica. Se mencionó, entre otras cosas, el empleo de un modelo lorentziano y un número para caracterizar el contraste, a saber el denominado Gradiente de Brenner. Sin embargo, se precisó que un enfoque microscópico no resulta suficiente dada la naturaleza del crecimiento de cordones en las placas MODS. En este apartado se mostrará algunos detalles adicionales concernientes a la implementación de este algoritmo.

1. Grad_Brenner.h: Cabecera de librería relacionada con el cálculo del gradiente de Brenner. Sólo se encuentra implementada una función:

void ***Grad_Brenner*** *(CXVideoOCX** ***m_XVideoOCX****, BYTE* frame, long* ***handle24o****, long* ***handle24s****, long* ***handle8****, double** ***B)***

2. Grad_Brenner.cpp: A continuación se presenta el desarrollo de esta función:

Descripción: Esta función se encarga del cálculo del gradiente de Brenner.

Argumentos: *m_XVideoOCX* es un puntero al objeto ActiveX creado en ventana de diálogo, así como *frame* representa un puntero al array de imagen (buffer). Por otro lado, los argumentos *handle* tienen que ver con buffers de memoria para distintos tipos de imagen, a saber, *handle24o* se refiere a un buffer para imagen RGB a máxima resolución 2048x1536, *handle 24s* a un RGB de 640x480 y *handle8* a una en escala de grises de 640x480. Finalmente, el resultado de los cálculos se almacena en la variable temporal **B*.

Funcionamiento: Esta función asume que ya se ha obtenido el buffer de memoria para imágenes a máxima resolución y que se han creado los respectivos manejadores para almacenar en memoria los resultados intermedios. El resto es aplicar los pasos 9 al 12, mostrados en la tabla 2.10, desde captura de imagen hasta el cálculo del gradiente. En la sección teórica se mostró la fórmula para calcular el gradiente y a continuación se presenta el pequeño segmento de código que implementa la mencionada sumatoria.

```
/* Implementación de cálculo para Gradiente de Brenner */
*B = 0;                                          //Inicialización de variable de salida
for (i1=10; i1<*m-12;i1++)                       //Suma sobre las filas, una columna fija
    for (j1=10; j1<*n-10; j1++)                  //Suma sobre las columnas
        *B+=pow (*(frame+i1*(*n)+j1)-*(frame+(i1+2)*(*n)+j1),2);  //Fórmula de gradiente
for (j1=10; j1<*n-12;j1++)                       //Barrido en columnas, fila fija
    for (i1=10; i1<*m-10; i1++)                  //Barrido sobre filas
        *B+=pow (*(frame+i1*(*n)+j1)-*(frame+i1*(*n)+j1+2),2); //Sumatoria de diferencias
```

Se decidió limitar la sumatoria a la zona central de la imagen (ignorando 10 píxeles a cado lado) con el objetivo de ganar tiempo de procesamiento y evitar problemas de borde encontrados en las imágenes. Esto no perjudica el diagnóstico en absoluto, debido a que aprovechamos el hecho de que la información sobre los bordes es anulada por el software de reconocimiento.

3. Auto_Focus.h: Cabecera de librería de enfoque automático.

```
void Auto_Focus (char* F, char* NP, int p1, int p2, bool* Opt, char* filename, int* z,
                 double* Z, double* Zw, double* Z0, double* Vz, double* PV, double*
                 AV, CXVideoOCX* m_XVideoOCX, LJ_HANDLE* lngHandle,
                 LJ_ERROR* E1, long* E2)
```

4. Auto_Focus.cpp: A continuación se detalla el desarrollo de la función de enfoque.

Descripción: Esta función implementa el enfoque dinámico de muestras. Consiste de dos partes principales: Enfoque macro y microscópico.

Argumentos: *F* es un puntero a array de char con una dirección base en la cual se guarda la totalidad de fotografías del proceso. *NP* es un puntero con información de código de la placa. Como se mencionó con anterioridad los contadores p1 y p2 brindan información con respecto a los pozos que se vienen analizando (XY_Global.cpp) y la posición dentro de un pozo (XY_Local.cpp), respectivamente. *Opt* tiene que ver con la opción de proceso que uno elige (esto se explica más adelante). Las variables que siguen con relación a Z fueron explicadas en Z_Move.cpp, excepto por *Z0* que indica el centro de la zona de barrido. Finalmente, *filename* es una salida del programa con la dirección donde se almacena una fotografía en particular.

Funcionamiento: En la sección teórica sobre "Procedimiento de Enfoque Automático" se explicó que para puntos cercanos al pico de contraste (según gradiente calculado) la curva podía aproximarse por una lorentziana y entonces era posible encontrar el pico de la curva por interpolación a partir de tres datos en posiciones distintas. Se mencionó adicionalmente que esto sería aproximadamente correcto dentro de un pozo, pero que en pozos distintos este procedimiento no sería suficiente para enfocar una muestra. A continuación, se prosigue con el desarrollo de esas observaciones. Experimentalmente, se ha determinado que los planos de cordones pueden estar separados hasta 1 mm, en el peor de los casos, en pozos distintos dentro de una misma placa MODS. Esto ocurre no sólo debido a imperfecciones mecánicas de los soportes del equipo sino también a la naturaleza misma de los cordones.

El inconveniente con esta observación deriva del hecho de que para que el algoritmo micro funcione bien (esto es, para que estime correctamente el pico) se deben tomar tres puntos apenas ligeramente separados (del orden de 0.05 mm entre puntos extremos). Si consideramos puntos muy separados, entonces nuestra muestra se vuelve borrosa en ambos extremos y no obtendríamos datos con información de valor para encontrar el máximo contraste. De tal manera que si, por algún motivo, consiguiéramos enfocar alguna muestra en un pozo, luego nuestras posibilidades de enfocar una muestra en el pozo siguiente serían muy limitadas sólo con el empleo de un enfoque local. Se consideró, por consiguiente, la necesidad de contar con un algoritmo de enfoque macro con un mayor rango de búsqueda. Este barrido macro cubre una distancia de 1 mm, en la cual se toman datos igualmente espaciados en dirección Z con pasos de 0.03 mm. El objetivo es ubicar el máximo de la curva que se obtiene, para luego entonces recién aplicar el procedimiento de enfoque microscópico local alrededor del pico.

Como se muestra en la figura 2.18, el procedimiento de enfoque macro consiste en barrer una zona de longitud *Zw centrada en *Z0 comenzando desde la parte inferior. Tras minuciosas observaciones se decidió considerar un paso de 0.03 mm y un tiempo de espera de 300 ms entre captura de imágenes en cada posición Z de análisis. En cada posición se calcula el gradiente de Brenner tres veces y se toma el promedio como valor característico del punto. Del vector de gradientes resultante, obtenemos el máximo valor y su índice en el array.

A manera de ejemplo, en la figura 2.18 se muestra un caso en el que el máximo de la curva ocurre en la octava posición (índice 8). Una vez encontrado el índice de la posición de máximo contraste se procede con el ajuste micro. Este bloque de ajuste local constituye un refinamiento del enfoque macro explicado en el párrafo anterior.

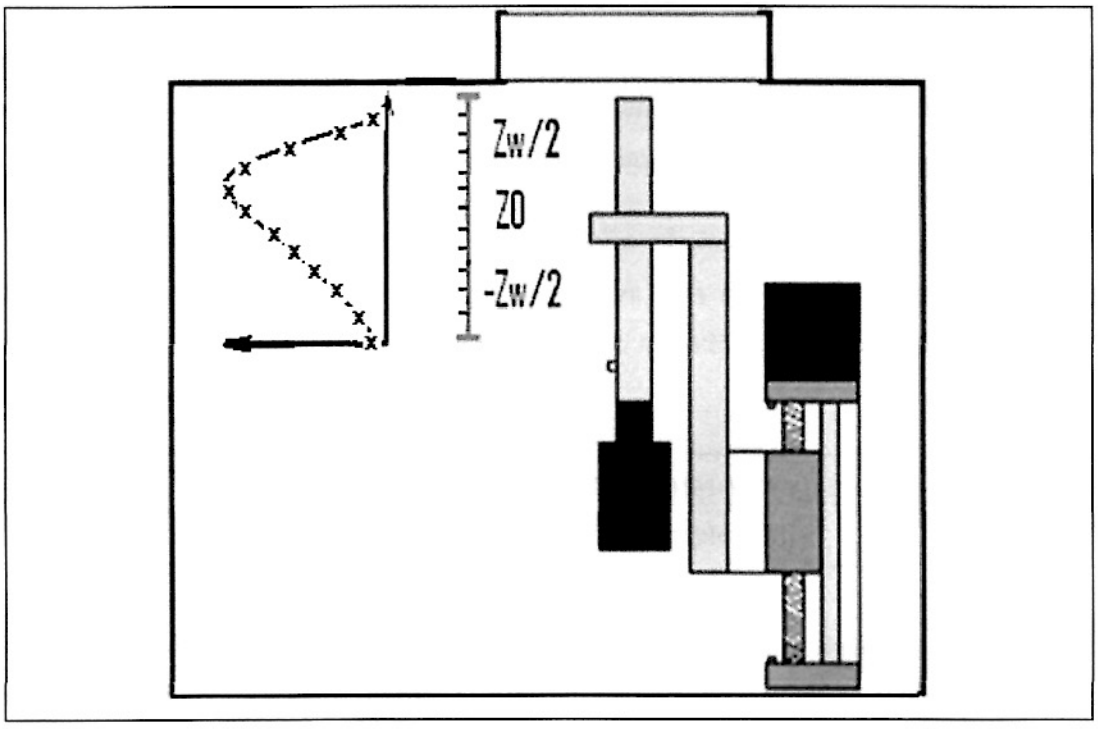


Figura 2.18: Enfoque automático

El enfoque microscópico se basa en un ajuste a curva lorentziana de los datos encontrados a fin de encontrar el máximo por interpolación y mejorar el contraste lo más que se pueda, sin consumir demasiado tiempo ni muchos recursos de procesamiento. En nuestro caso, aprovechamos el barrido realizado por el proceso de enfoque macro para conocer los valores de gradiente de Brenner en la vecindad del pico de la curva. El resultado del algoritmo está restringido a la zona definida por [*Zp- (*Zw)/2 *Zp+ (*Zw)/2], alrededor del máximo *Zp macro.

Experimentalmente se ha encontrado que el valor *Zw para este bloque está alrededor de 0.05 mm. Esta distancia característica podría tener el significado de "espesor" del plano de cordones de TB. En la implementación en C se ha considerado un valor *Zw=0.06 mm, el doble del paso considerado en el enfoque macro, de esta manera aprovechamos los datos obtenidos por el enfoque global. El código empleado en C para encontrar el pico de la curva lorentziana, en base a tres datos en el eje Z, se muestra a continuación:

```
*R = 0.5*((*Y1)*(*Y3-*Y2)*(pow(*Z1,2)-pow(*Z2,2))-(*Y3)*(*Y2-*Y1)*(pow(*Z2,2)-
     pow(*Z3,2)))/((*Y1)*(*Y3-*Y2)*(*Z1-*Z2)-(*Y3)*(*Y2-*Y1)*(*Z2-*Z3));
```

Donde: *Z1 es la posición inferior, *Z2 es la posición actual y *Z3 es la posición superior. Los valores Yi son los gradientes de Brenner promedio en las posiciones Zi. *R es la posición estimada del pico de la curva. Este enfoque micro tiene un efecto correctivo sobre la primera aproximación macro y se espera mejore el contraste obtenido al final del enfoque global.

El bloque de enfoque automático macro tiene la tarea adicional de generar un filename, en el cual se guardará la imagen capturada por la cámara para su posterior procesamiento. Este filename tiene un formato estándar. Para lectura de muestras de un paciente, el formato se muestra a continuación:

```
filename = "[Base address]\MODS\[DD-MMM-YYYY]\[Process]\
           Plate [X]\Patient [Y]\[Well]\Sample [N].jpg"
```

Brevemente podemos señalar que la dirección base señalada es una dirección genérica que el usuario elige, podría ser por ejemplo: C:\Documents and Settings. Luego de la palabra clave MODS, se adjunta la fecha actual de registro. Posteriormente, información sobre el proceso de lectura ejecutado, el código de la placa (NP), el número de paciente (calculado como: (p1-1)/4), el pozo de análisis (codificado) y el número de muestra actual dentro del pozo (p2).

Finalmente, se procede a guardar la imagen capturada en el filename especificado en el párrafo anterior (como se explicó en la tabla 2.10). Esta tarea es sumamente importante, no sólo por fines de archivo de datos, sino porque la tarea de diagnosis depende de la existencia de un archivo de imagen en una dirección conocida, tal como se explicará en el siguiente apartado. A continuación se muestra a manera de resumen un diagrama de bloques con el procedimiento de enfoque automático.

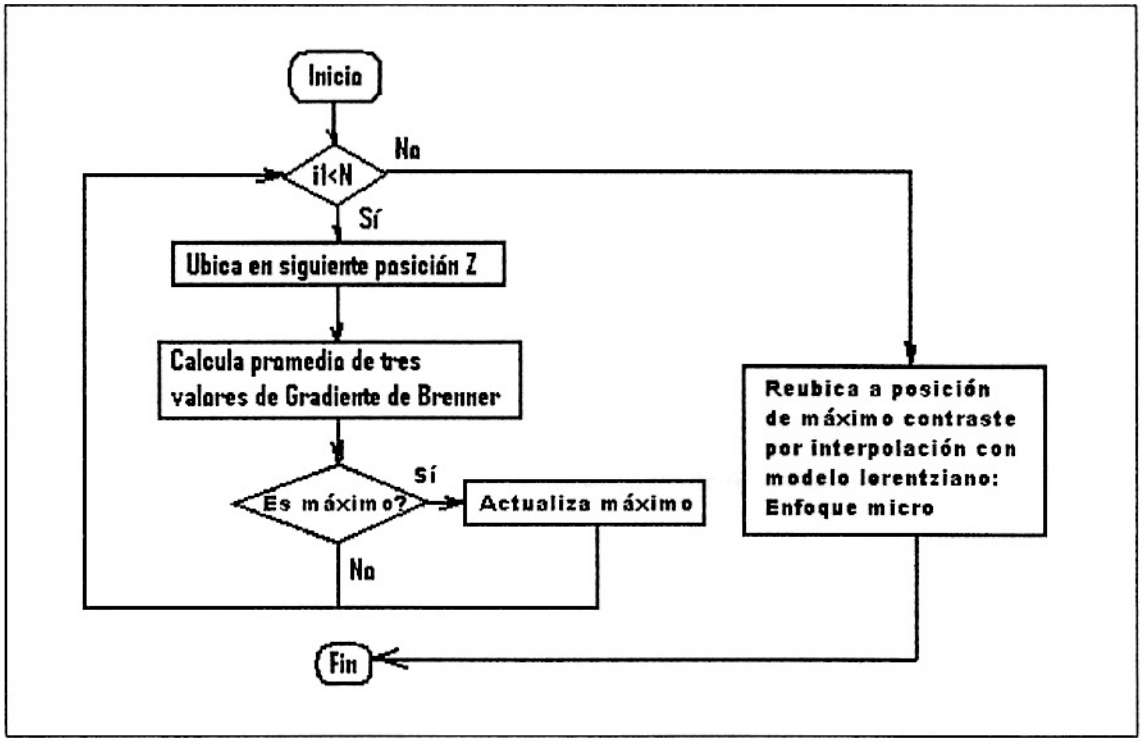


Figura 2.19: Diagrama de flujo de enfoque macro

2.2.6. Diagnosis de TB

El software de control se desarrolló sobre plataforma de Windows con la intención de posterior implementación en Linux. Esto no sería posible durante el presente proyecto debido a que no disponemos de un driver adecuado de la cámara MiniVID para sistemas Linux. Por otro lado, el software de reconocimiento se encuentra implementado ciertamente en C, pero en sistema Linux. Debido a esta incompatibilidad, se debió implementar un procedimiento adicional para poder acceder al código de diagnosis desde el software de control. Para resolver el impase, se decidió instalar una máquina virtual mediante el software VirtualBox de la marca Sun con Windows como host y Linux (Ubuntu 9.10 512 MB RAM) como guest. De esta forma, mediante virtualización es posible tener ambos sistemas trabajando simultáneamente en una misma máquina física [15]. En este caso, no se requiere compartir hardware, sólo implementar un proceso de sincronización para poder intercambiar datos entre las dos aplicaciones.

Antes de comenzar con la explicación del funcionamiento de sincronización, se presenta a continuación un resumen del procedimiento a seguirse en la máquina virtual. El primer paso luego de instalar VirtualBox en Windows y Linux en la máquina virtual, consiste en la creación de una carpeta compartida entre los dos sistemas.

Para poder hacer uso efectivo de la mencionada carpeta es necesario emplear las APIs de VirtualBox, las cuales pueden ser descargadas libremente. Esta carpeta finalmente será el "puente" entre los dos sistemas, dado que cualquier archivo escrito en esa ubicación por una aplicación, podrá ser leído por la otra. El siguiente paso consiste en la instalación del software de reconocimiento, el cual es denominado RMODS. Este proceso implica la instalación de las dependencias asociadas al programa. En resumen, se debe instalar el paquete build-essential, gfortran, g77, libjpeg-progs, fftw-dev y g2c. Si todas las dependencias fueron instaladas con éxito podremos compilar y ejecutar el programa desde su ubicación sin problemas. En la implementación realizada, sólo se requiere iniciar la máquina virtual, abrir un terminal y escribir ./conectorRMODS.sh + ENTER para ejecutar el software de reconocimiento.

1. **Diagnosis.h:** Cabecera de librería de intercambio de datos con RMODS. Sólo se encuentra implementada una función:

*void **Diagnosis**(char* **filename**, bool* **K_bool**, double* **Score**)*

2. **Diagnosis.cpp:** A continuación se detalla el desarrollo de esta función.

Descripción: Esta función se encarga de realizar el intercambio de archivos entre RMODS y MODS_Plate_Reader_X, para así poder obtener el resultado del diagnóstico de TB.

Argumentos: *filename* es un puntero a array con el nombre del archivo con formato, donde se guardó la imagen capturada en mejor contraste posible. *K_bool* es un puntero al resultado booleano del diagnóstico y *Score* apunta al valor de probabilidad de enfermedad, resultado del modelo logístico de reconocimiento de patrones MODS. Internamente, el algoritmo de reconocimiento utiliza un valor umbral o "Cut-off" (encontrado por métodos estadísticos) para determinar si un resultado es positivo (*Score>Cut-off) o negativo (*Score<Cut-off).

Funcionamiento: El procedimiento de intercambio de datos está basado en la generación de archivos en la carpeta compartida. Para nuestro caso, la dirección de la carpeta fue: *C:\Documents and Settings\Daniel\ubuntu_windows*. En un primer momento, el software RMODS se encuentra preguntando si existe un archivo de imagen JPG en un lazo infinito. Este archivo JPG será creado en su momento por el programa Auto_Focus_Micro.cpp tras culminar el proceso de autoenfoque. Internamente, "Diagnosis.cpp" genera una copia de filename en la carpeta compartida con el nombre clave "image.jpg" para que sea accesible a RMODS.

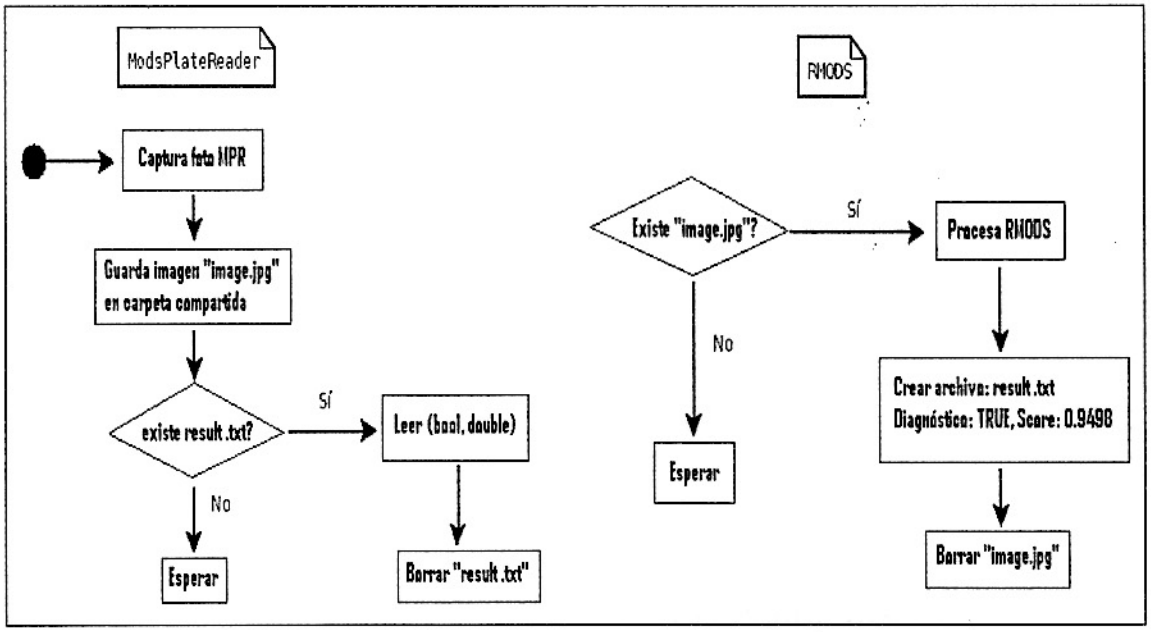


Figura 2.20: Diagrama de flujo de intercambio de mensajes

Una vez que el software RMODS determina que el archivo "image.jpg" existe, entonces realiza el procesamiento de la imagen y proporciona el resultado final en un archivo TXT con nombre clave "result.txt". En este archivo de texto se escribe el resultado del análisis en una variable booleana y un score de probabilidad, en una sola línea, separados por un espacio y en el orden mencionado. Finalmente, tras el procesamiento se destruye la imagen en la carpeta compartida. Por otro lado, como proceso concurrente, el software de control se encuentra preguntando continuamente por la existencia del archivo de texto "result.txt". Este archivo de texto será generado por RMODS en su momento, al finalizar el procesamiento de la fotografía. Una vez que "result.txt" es encontrado, se procede a capturar la información contenida para luego destruir el archivo de texto. El procedimiento implementado de sincronización se resume en el diagrama de flujo 2.20 sobre intercambio de mensajes.

2.2.7. Referencia inicial de posiciones

Dado que los posicionadores sólo cuentan con sensores discretos de fin de carrera y no con sensores absolutos de posición, se requiere un procedimiento de inicialización para ubicar el lector en una posición conocida. Esta rutina es la primera que se realiza al encender el equipo y es de importancia crítica dado que de no ejecutarse, podría ocasionar daños en el lector por choques con la estructura mecánica de soporte. Por consideraciones de construcción mecánica, decidimos utilizar los fines de carrera correspondientes a los extremos +X, +Y y –Z.

1. Initial_Position.h: Define la cabecera de la función de posicionamiento inicial de referencia.

void ***Initial_Position*** *(int* x, double* X, int* y, double* Y, int* z, double* Z, double* Vxy, double* Vz, int* Nx, int* Ny, double* PV, double* AV, LJ_HANDLE* lngHandle, LJ_ERROR* E1, long* E2, char* user_time)*

2. Initial_Position.cpp: A continuación se describe el desarrollo de este algoritmo.

Descripción: Este programa ubica a los tres posicionadores en posición inicial en base a los fines de carreras acoplados a ellos.

Argumentos: Todos los argumentos han sido explicados en secciones anteriores, excepto *user_time*, el cual es un puntero a array de tipo char con información del tiempo requerido para completar la operación de posicionamiento inicial.

Funcionamiento: En primer lugar, se mueve el posicionador en eje Z en dirección negativa en un lazo *while* hasta que el fin de carrera se active. Se eligió este orden para evitar que el microscopio choque con la mesa que soporta a la placa MODS. Una vez que el posicionador Z ha alcanzado la posición más baja, se procede con el movimiento en el plano XY. Los posicionadores XY son movidos simultáneamente en dirección +X, +Y hasta que los fines de carrera respectivos se activen (figura 2.21). Finalmente, las posiciones se actualizan a los valores extremos: *X=152.4, *Y=203.2 y *Z=0, donde las unidades están dadas en milímetros.

2.2.8. Rutina de lectura de control positivo

El siguiente procedimiento representa el primer proceso de lectura a llevarse a cabo durante un día cualquiera de análisis de muestras. En la parte teórica se hizo referencia a los controles internos (sección 1.2.6) desarrollados para validar placas. El primer control interno es el denominado Control Positivo (figura 2.22) y a continuación se detalla su funcionamiento.

1. Positive_Control.h: Cabecera del procedimiento de lectura de placa de control positivo.

void ***Positive_Control*** *(int* x, double* X, int* y, double* Y, int* z, double* Z, double* Zw, double* Vxy, double* Vz, int* Nx, int* Ny, char* F, double* PV, double* AV, double* X0, double* Y0, double* Z0, double* dX, double* dY, double* Dm, int* Tr, char* NP, bool* Opt, CXVideoOCX* m_XVideoOCX, LJ_HANDLE* lngHandle, LJ_ERROR* E1, long* E2, char* user_time, bool Manual, bool Man_Ctrl_Bool)*

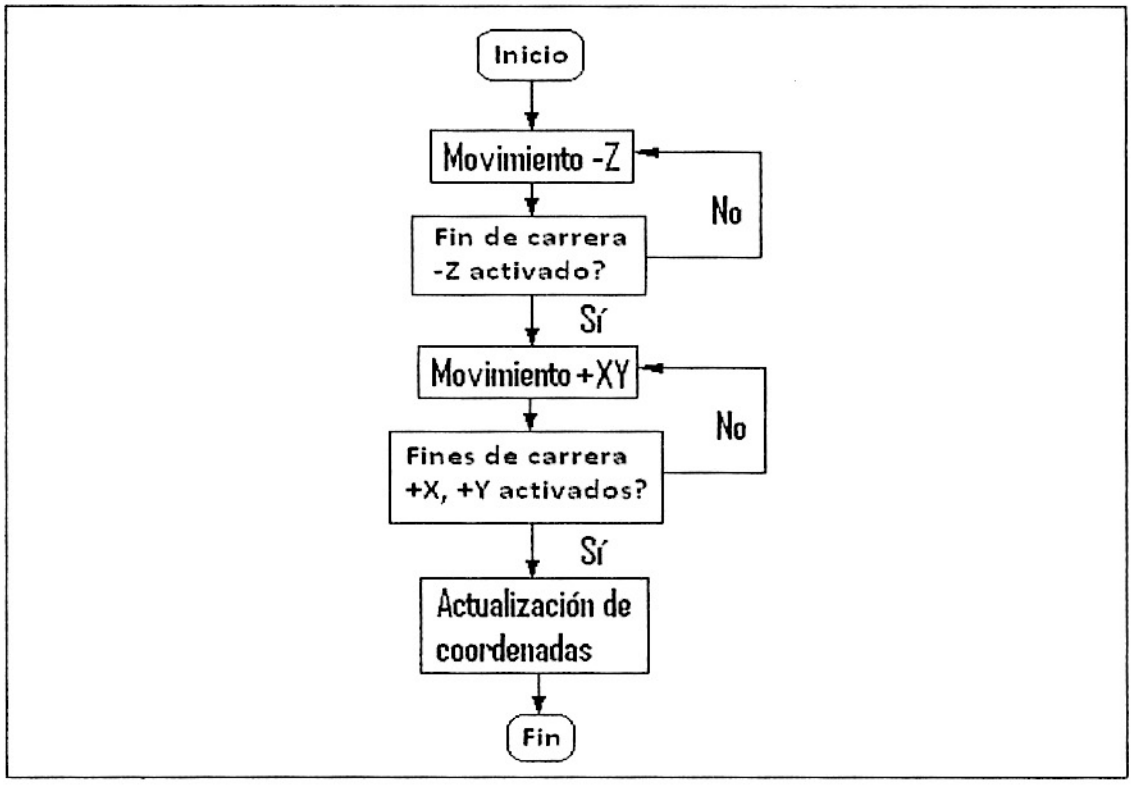


Figura 2.21: Diagrama de flujo de posicionamiento inicial

2. Positive_Control.cpp: A continuación se describe el desarrollo de este algoritmo de lectura.

Descripción: Implementación del proceso de lectura de placa de control positivo.

Argumentos: Todos los argumentos han sido explicados en secciones anteriores, excepto por los dos últimos que tienen relación con controles manuales habilitados para el usuario. *Manual* habilita el modo manual para este procedimiento y *Man_Ctrl_Bool* establece que la placa ha superado el control positivo en base a decisión del usuario. El argumento de entrada *Tr*, para este caso, brinda información contenida en la siguiente matriz: (1,5); (2,5); (3,5); (4,5); (1,6); (2,6); (3,6); (4,6) tal y como se indica en la figura 2.15 de codificación de índices para pozos.

Funcionamiento: Este bloque se inicia con la generación de carpetas para almacenamiento de fotografías y resultados. En la sección de enfoque automático se indicó que existe un filename en el cual se guardan las fotografías y que guarda un cierto formato. Además, se realizó la indicación sobre el mismo, para el caso de lectura de placas de un paciente. En este caso de control positivo, la placa no se refiere a un paciente y se debe realizar una corrección al filename mostrado con anterioridad. En este caso, el filename toma la siguiente forma:

filename = "[Base address]\MODS\[DD-MMM-YYYY]\Positive Control\[Well]\Sample [N].jpg"

En filename, se hace notar que se utiliza la palabra clave "Positive Control" luego de la fecha y que se emplea un código para los pozos. Este código para los archivos se indica a continuación:

Tabla 2.11: Codificación de pozos para control positivo

Pozo	Código	Resultado esperado
1	Control CTf1c5(+)	Positivo
2	Control CTf2c5(+)	Positivo
3	Control INHf3c5(-)	Negativo
4	Control RIFf4c5(-)	Negativo
5	Control CTf1c6(+)	Positivo
6	Control CTf2c6(+)	Positivo
7	Control INHf3c6(+)	Positivo
8	Control RIFf4c6(+)	Positivo

Se hace notar que esta tabla tiene los mismos resultados que la tabla 1.3 de controles internos. De esta manera, se termina indicando que la labor inicial de este programa es generar las ocho carpetas de la tabla 2.11 para los pozos de control positivo (el resto de la codificación para filename, sobre número de muestra, es tarea de enfoque automático macro). En adelante, se desarrolla el proceso de lectura de control positivo automático. El programa de lectura automática se inicia con un lazo while con un máximo de ocho iteraciones (cada iteración representa un pozo) las cuales pueden ser interrumpidas de detectarse un resultado distinto al esperado (tabla 2.11). Internamente, se hace una llamada a la función de posiciones de referencia "XY_Global.cpp" para ubicar virtualmente el centro del siguiente pozo de análisis. Luego de esto, se ingresa a un nuevo lazo *while* para hacer los barridos dentro de un pozo. En este caso, las condiciones son que las iteraciones se repitan siempre que el resultado del análisis sea negativo y que el número de iteración sea menor que el número de particiones del pozo.

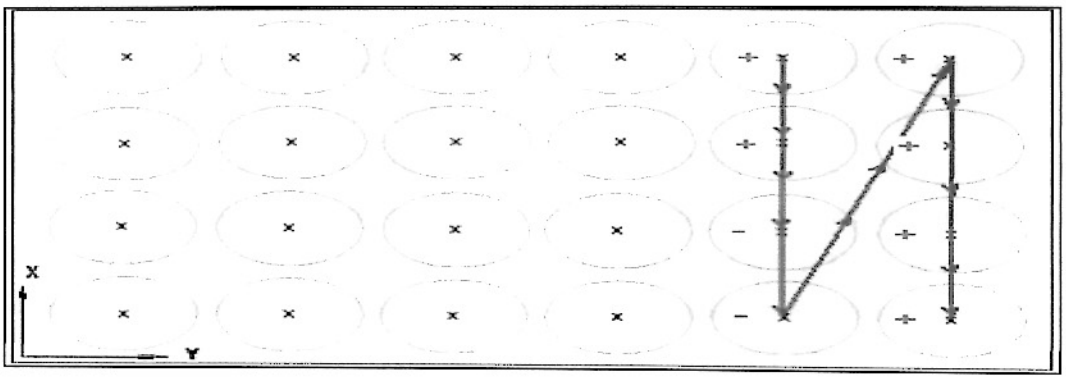


Figura 2.22: Lectura de placa de control positivo

Dentro de este lazo while se considera a la función "XY_Local.cpp" para realizar los barridos virtuales dentro del pozo. El paso siguiente, es evaluar la condición de que la posición virtual está efectivamente dentro de los límites del pozo. De ser el caso, se procede con las rutinas de enfoque automático y diagnosis como se explicó en detalle con anterioridad.

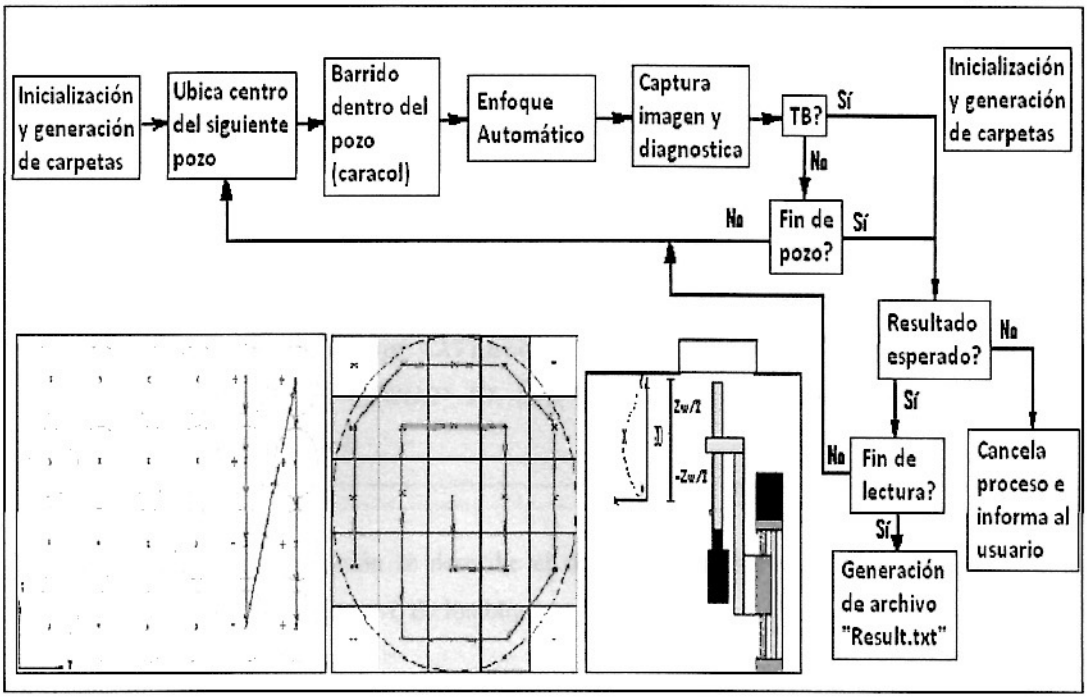


Figura 2.23: Diagrama de flujo de lectura de placa de control positivo

Tras culminar el primer lazo *while* se evalúan los resultados obtenidos de diagnosis y se compara con el patrón esperado. De encontrarse todo correcto, se almacena el resultado booleano y el score de resultados. El resultado final para una lectura satisfactoria, es la generación de archivo de texto "results.txt" en la carpeta especificada para control positivo. Este archivo muestra ocho líneas, una correspondiente a cada pozo. El contenido de estas líneas es: número de pozo, score obtenido y resultado obtenido como positivo o negativo.

El score para resultados negativos se considera como el máximo score negativo obtenido dentro del pozo. En caso de alguna falla encontrada, los lazos *while* son interrumpidos por sentencias *break* y se indica en pantalla la causa de la interrupción del código. En este caso no se genera un archivo de resultados TXT. Finalmente, a manera de resumen, se muestra el diagrama de flujo de la operación de lectura de placas de control positivo en la figura 2.23. Como se mencionó en la sección de argumentos, esta rutina de control cuenta con modo de ejecución manual, a fin de que un usuario experto tenga la opción de validar la placa de control con el consecuente ahorro de tiempo de análisis.

2.2.9. Rutina de análisis de placas de pacientes

Una vez que se ha validado el lote de placas del día, se procede con lectura de placas MODS que contienen muestras de pacientes. No obstante, se reserva una columna de control; en este caso, de control negativo, para validar una placa en particular (sección 1.2.6). A continuación se presenta el procedimiento de lectura de placas de pacientes.

1. Analysis.h: Rutina de lectura de placas con muestras de pacientes.

void ***Analysis*** *(int* **x**, double* **X**, int* **y**, double* **Y**, int* **z**, double* **Z**, double* **Zw**, double* **Vxy**, double* **Vz**, int* **Nx**, int* **Ny**, char* **F**, double* **PV**, double* **AV**, double* **X0**, double* **Y0**, double* **Z0**, double* **dX**, double* **dY**, double* **Dm**, int* **Tr**, char* **NP**, bool* **Opt**, CXVideoOCX* **m_XVideoOCX**, LJ HANDLE* **lngHandle**, LJ_ERROR* **E1**, long* **E2**, char* **user_time**, bool **Manual**, bool **Man_Ctrl_Bool**)*

2. Analysis.cpp: A continuación se describe el desarrollo de este algoritmo. Similarmente al caso anterior, se hace uso extensivo de los bloques funcionales implementados hasta aquí.

Descripción: Implementación de lectura de placas MODS con muestras de pacientes.

Argumentos: Todos los argumentos han sido explicados en apartados anteriores. En este caso, el argumento de entrada Tr nos lleva a la siguiente matriz: (1,3); (2,3); (3,3); (4,3); (1,1); (2,1); (3,1); (4,1); (1,2); (2,2); (3,2); (4,2); (1,4); (2,4); (3,4); (4,4); (1,5); (2,5); (3,5); (4,5); (1,6); (2,6); (3,6); (4,6), esto de acuerdo con la codificación de la figura 2.15 y teniendo en mente que la primera columna de análisis es la tercera.

Funcionamiento: Este bloque se inicia con la creación de carpetas donde se almacenarán las fotografías y los resultados obtenidos del análisis. Recordemos de la sección 2.2.5 de enfoque automático, que existe un cierto formato establecido para el filename donde se guardan las fotografías para los pacientes. Sin embargo, esta codificación sufre una pequeña variante cuando se trata de analizar columna de control negativo, a saber:

filename = "[Base address]\MODS\[DD-MMM-YYYY]\[Process]
Plate [X]\Negative Control\[Well]\Sample [N].jpg"

Se hace notar que se utiliza la palabra clave "Negative Control" en lugar de "Patient" para distinguir las direcciones.

Para el caso de pacientes, se menciona nuevamente el formato de filename:

filename = "[Base address]\MODS\[DD-MMM-YYYY]\[Process]\ Plate [X]\Patient [Y]\[Well]\Sample [N].jpg"

Finalmente, la codificación para el caso de placas de pacientes es como sigue:

Tabla 2.12: Codificación de pozos para análisis de placas

Pozo	Codificación para columna de Control Negativo	Codificación para columna de Análisis de Pacientes
1	Control CTf1c3(-)	CT1
2	Control CTf2c3(-)	CT2
3	Control INHf3c3(-)	INH
4	Control RIFf4c3(-)	RIF

En la codificación se recuerda una característica del pozo en cuestión (figura 1.1) y el resultado esperado para el análisis de un cierto pozo, cuando se trata de casos de control interno. El programa sigue con un lazo *while* que tiene un número máximo de 24 iteraciones, las cuales pueden ser interrumpidas en caso de no superarse la prueba de control negativo. Las primeras cuatro iteraciones definen la continuidad del proceso, en esta parte el algoritmo es idéntico al de control positivo. Una vez superada la prueba, se prosigue con el análisis de columnas de muestras de pacientes, para las cuales se pueden producir "saltos" entre columnas si es que en las dos primeras filas no se encuentra evidencia de TB (ver figura 2.24 y sección 1.2.4). El proceso de lectura por lo demás, es análogo al descrito en control positivo (sección 2.2.8) consistente en ubicación XY, barrido en caracol, enfoque automático y diagnosis de TB.

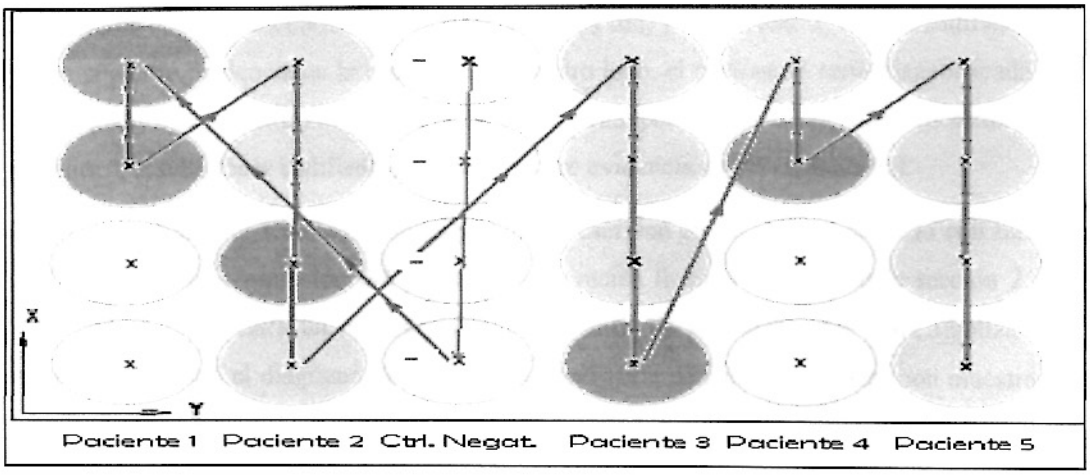


Figura 2.24: Lectura de placa de análisis de muestras de pacientes

Recordemos que se ha implementado el análisis de pozos de control en modo manual, con lo cual, se pasaría directamente al análisis automático de columnas de pacientes. Para esta parte de análisis de placas, se utilizó una codificación adicional para indicar el tipo de diagnóstico por paciente, tal como se muestra a continuación:

Tabla 2.13: Codificación de resultados de diagnosis

Codificación	Resultado del diagnóstico
0	No se ha encontrado evidencia de TB
1	Se ha encontrado evidencia de TB susceptible a INH y RIF
2	Se ha encontrado TB susceptible a RIF
3	Se ha encontrado TB susceptible a INH
4	Se ha encontrado TB multidrogoresistente
5	Resultado indeterminado, se recomienda repetir la prueba.

Como ejemplo de la codificación empleada para el diagnóstico, usaremos el caso mostrado en la figura 2.24, donde se ha representado con color gris oscuro resultados negativos, en gris claro, resultados positivos y en blanco pozos que no fueron analizados (asumimos que el control negativo se superó exitosamente en modo manual).

Según los resultados mostrados, el paciente 1 tendría codificación 0 (dado que no se encontró evidencia de enfermedad en las dos primeras filas), mientras que el paciente 2, tendría codificación 3 (no se encontró evidencia en la tercera fila) y el paciente 3, tendría codificación 2 (no se encontró evidencia en la cuarta fila). Por otro lado, el paciente 4 sería diagnosticado con codificación 5 (se encontró evidencia en la primera fila, pero no en la segunda), mientras que el paciente 5, resultó tener codificación 4 (se encontró evidencia en las cuatro filas).

Como punto final, los resultados del algoritmo se escriben en un archivo de texto con formato TXT con toda la información del procedimiento recién llevado a cabo. En la sección 2.4 de "Resultados" se muestra un ejemplo obtenido a partir de una placa real. Para finalizar esta sección, se muestra el diagrama de flujo para la operación de lectura de placas con muestras de pacientes en la figura 2.25.

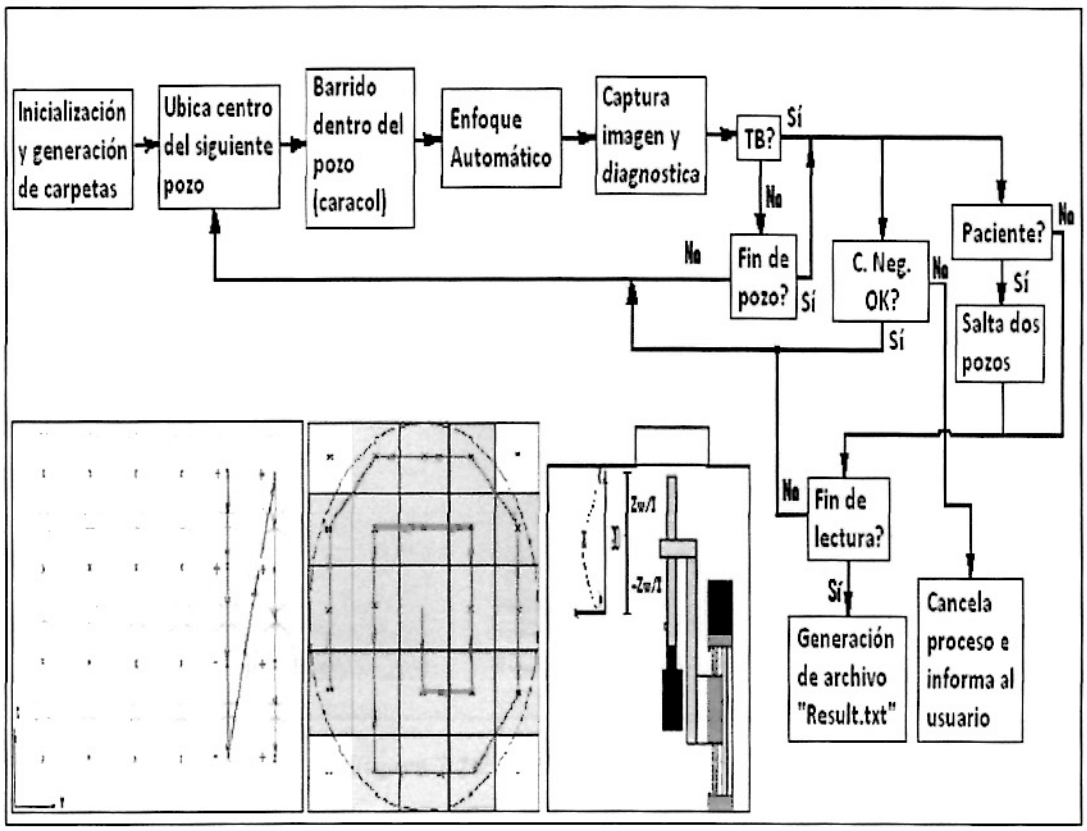


Figura 2.25: Diagrama de flujo de lectura de placa de muestras de pacientes

2.2.10. Interfaz gráfica de usuario MODS

La interfaz GUI muestra de manera amigable los parámetros, controles y procedimientos necesarios para que el usuario pueda leer placas MODS con bastante facilidad. Adicionalmente a los procedimientos de lectura automática, se ha considerado disponer de algunos controles manuales a fin de que el usuario pueda realizar alguna parte de las tareas de acuerdo con su criterio como ya se ha explicado en secciones anteriores.

Antes de continuar con la descripción de la interfaz GUI, se presentan algunas notas generales sobre el procedimiento de creación de la interfaz de usuario MODS.

El proceso de creación de la interfaz gráfica se inicia con la disposición de los componentes en la ventana de diálogo y creación de variables miembro para intercambiar datos dentro de la aplicación. Una vez creadas las etiquetas en Static Text para las funciones, se generan botones de aceptación para éstas y se asocia código a ellas.

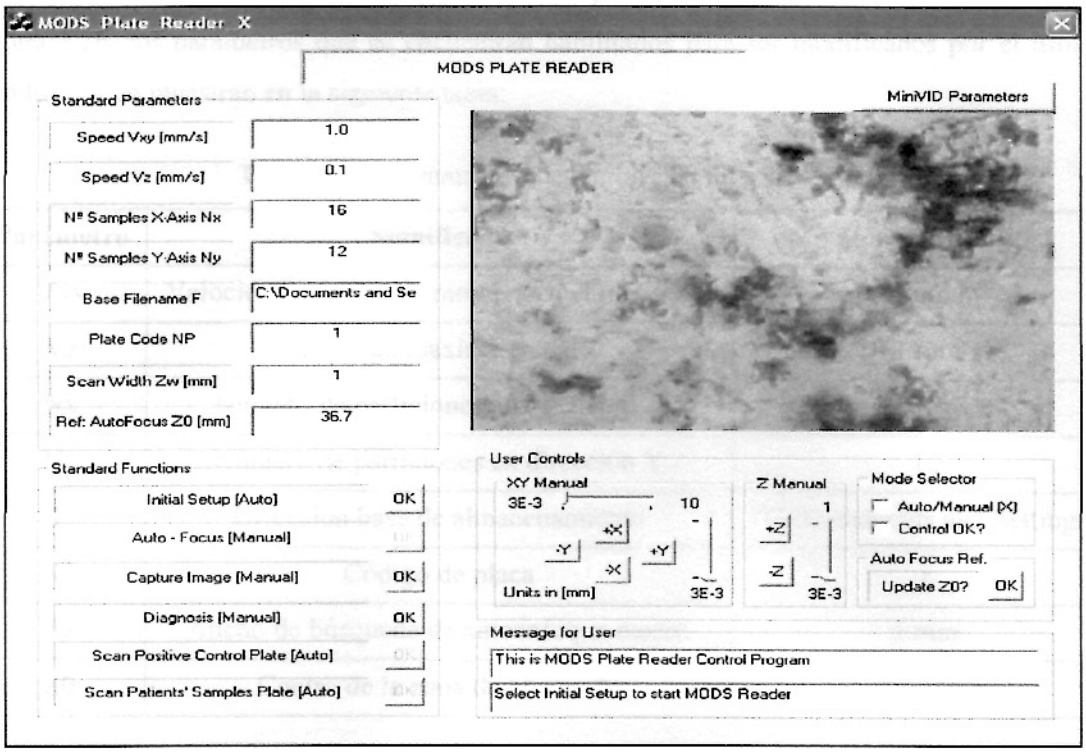


Figura 2.26: Interfaz de usuario MODS

Es importante recordar que las funciones se ejecutan con un click sobre el botón de OK una vez que se han ingresado los parámetros que el usuario desea emplear. Este botón de aceptación se encuentra al lado derecho de cada una de ellas. Con respecto al procedimiento de inicialización en interfaz de usuario, resumimos los pasos a seguir en la siguiente tabla.

Tabla 2.14: Instrucciones de inicialización para interfaz de usuario

Objetivo	Instrucción
Incluir librerías	Se agrega la lista completa de librerías desarrolladas en MODS_Plate_Reader_X.h, las cuales han sido descritas en este documento. Adicionalmente, en la carpeta "Resources" se debe agregar el archivo labjackud.lib con la implementación de las funciones relacionadas con la tarjeta de adquisición
Inicializar variables miembro	En MODS_Plate_Reader_XDlg.cpp se puede editar el valor por defecto de las variables miembro.
Abrir canal DAQ	OpenLabJack(LJ_dtU3, LJ_ctUSB, "1", TRUE, lngHandle)
Configuración por defecto en DAQ	ePut (*lngHandle, LJ_ioPIN_CONFIGURATION_RESET, 0, 0, 0)
Configurar cámara	Seguir pasos 2-6 de la tabla 2.10

Proseguimos nuestra descripción con el bloque de parámetros en GUI. Como se muestra en la figura 2.26, los parámetros que se encuentran habilitados para ser modificados por el usuario son los que se muestran en la siguiente tabla:

Tabla 2.15: Parámetros disponibles en interfaz GUI

Parámetro	Significado	Valor por defecto
Vxy	Velocidad máxima de motores en el plano XY	1 mm/s
Vz	Velocidad máxima en eje Z	0.1 mm/s
Nx	Número de particiones en dirección X	16
Ny	Número de particiones en dirección Y	12
F	Dirección base de almacenamiento	C:\Documents and Settings
NP	Código de placa	1
Zw	Ancho de búsqueda de autoenfoque macro	1 mm
Z0	Centro de la zona de búsqueda	36.7 mm

Estos parámetros pueden ser modificados sencillamente escribiendo en los Edit Boxes al lado de cada etiqueta. Internamente, antes de ejecutar cualquier procedimiento, estos valores son actualizados mediante la instrucción Update (TRUE). Por otro lado, las funciones implementadas se muestran en la tabla 2.16, como se muestra a continuación:

Tabla 2.16: Funciones disponibles en interfaz GUI

Función	Modo de funcionamiento	Referencia
Initial Setup	Auto	Sección 2.2.7
Auto-Focus	Manual	Sección 2.2.5
Capture Image	Manual	Sección 2.2.4
Diagnosis	Manual	Sección 2.2.6
Scan Positive Control Plate	Auto/Manual	Sección 2.2.8
Scan Patients' Samples Plate	Auto/Manual	Sección 2.2.9

Sobre el lado superior derecho, podemos observar el cuadro del control ActiveX, el cual nos permite ver el preview de la cámara, así como un botón de parámetros, el cual puede ser elegido en cualquier momento para corregir factores tales como tiempo de exposición, brillo, corrección a blanco automático, entre otros. Sobre el lado inferior derecho, se encuentran contenedores con la componente manual del equipo. Estos controles son bastante intuitivos, en el sentido que indican las direcciones de avance y la distancia que se desea avanzar. Los sliders toman como valor máximo 10mm y como mínimo un paso, equivalente a 3μm en dirección XY.

Para la dirección Z, el slider toma el valor máximo de 1mm y valor mínimo igual al anterior. Sobre este mismo contenedor, sobre el lado derecho podemos notar Checkboxes para control manual (X) o automático (O) del equipo, así como uno en la parte inferior, para validar las columnas de control. Finalmente, se cuenta con un botón que nos permite capturar el valor de Z y convertirlo en el valor de Z0 actualizado. Una nota importante es que los controles manuales se habilitan con el Checkbox de control manual/automático marcado con una X, esto es, deben ser habilitados por el usuario para poder ser utilizados. Sobre la parte inferior derecha, se puede observar los mensajes a usuario que indican el estado del proceso. Inicialmente, se nos muestra un mensaje de bienvenida, el cual advierte que el proceso de inicio del equipo es el de posicionamiento de referencia. Mientras no se ejecute Initial Setup, las funciones automáticas están deshabilitadas, las únicas disponibles son las manuales como se aprecia en la figura 2.26.

2.3. Procedimiento de lectura usando MODS Plate Reader

Una vez explicado el funcionamiento del software, sólo nos resta hacer algunas anotaciones para los usuarios finales, a fin de que puedan realizar lectura de placas sin inconvenientes. Las instrucciones (c), (e), (f) e (i) son opcionales, mientras que el resto son imperativas y deben ejecutarse en el orden indicado. A continuación, se lista en forma ordenada el conjunto de operaciones específicas necesarias para una lectura correcta de placas.

a) Encender la computadora del equipo, así como el switch del módulo de alimentación de circuitos e iluminación. Al hacer esto, debería observar que la lámpara se enciende automáticamente y los ventiladores comienzan a trabajar.

b) En el computador, ejecutar el programa "MODS_Plate_Reader_X" e iniciar la máquina virtual instalada para el software de reconocimiento. "RMODS" se ejecutará escribiendo "./conectorRMODS.sh" en un terminal y presionando Enter. En el panel de operador MODS_Plate_Reader_X debe poder observar el preview de la cámara.

c) El equipo se diseñó para que permanezca alineado en el tiempo. Sin embargo, debido a las continuas vibraciones esto podría alterarse ligeramente. A consecuencia de esto, se recomienda verificar el alineamiento del equipo periódicamente. Un procedimiento sencillo podría consistir en saturar el sensor de la cámara y conseguir que la zona más brillante coincida con el centro de la imagen, esto variando los tornillos que sujetan a la lámpara. Para saturar al sensor debe aumentar la corriente que circula por la lámpara girando el potenciómetro que se encuentra en el interior del compartimento de alimentación.

d) Ubicar la placa MODS de análisis, con su bolsa ziplock de protección, en el soporte diseñado para ella en la posición que se indica en la figura 1.1. Con los controles manuales ubicar una zona en la que no se observen cordones ni background y realizar la corrección de blanco de la cámara. Los parámetros de la cámara son disponibles presionando el botón "MinVID Parameters". Luego de esto, presionar el botón "Auto White Balance", esto debería mostrar el preview de la cámara en escala de grises.

e) A continuación, con los controles manuales ubicar alguna zona donde se encuentren cordones para ajustar los parámetros de la cámara MiniVID, tales como tiempo de exposición, contraste, etc. Si se desea se puede hacer una corrección de actualización de la zona central de búsqueda de cordones presionando el botón "Z0", en el contenedor "First Focus", una vez que se ha enfocado un plano de cordones adecuadamente.

f) Una vez ajustados los parámetros de la cámara, editar el panel de operador con los valores que se consideren adecuados para lectura. Los parámetros por defecto pueden ser modificados escribiendo nuevos valores en el contenedor "Standard Parameters".

g) Para ubicar al equipo en una posición conocida, el primer proceso se que se ejecuta será "Initial Setup", esto se consigue presionando el botón asociado a esta función en el contenedor "Standard Functions".

h) Una vez que el equipo se encuentra en su posición cero, el usuario debe definir si la lectura tendrá componente manual o si será íntegramente de tipo automático. Si se desea realizar el control del equipo de manera manual, debe seleccionar con un check la casilla indicada "Auto/Manual" en el contenedor "Mode Selector."

i) En modo de prueba se ha habilitado botones para realizar captura de imágenes y enfoque automático de manera independiente a lecturas. Para poder seleccionar estas funciones el control debe estar en modo manual.

j) Por otro lado, con respecto a lectura de placas, para validar los controles en modo manual, se debe pulsar el botón "Control OK?" en "Mode Selector".

k) De elegir la opción automática (por defecto) al presionar los botones correspondientes a "Scan Positive Control Plate" y "Scan Patients' Samples Plate" se iniciará los procesos de lectura de placa de control positivo y lectura de placa de pacientes, respectivamente.

l) Culminada la lectura de placa, recoger los resultados en la carpeta indicada en panel GUI.

2.4. Resultados

Con el equipo terminado de construir en su integridad y el software implementado en computadora se procedió con lectura de placas de manera rutinaria siguiendo el procedimiento recién descrito. En la figura 2.26 se muestra el resultado final en archivo de texto que se obtuvo a partir de la lectura de una placa de control positivo un día al azar, el código de la placa fue A2342 y el día de la lectura el 16 de mayo del presente año. El tiempo de lectura empleando la computadora del laboratorio fue de aproximadamente 4 horas y el archivo de almacenamiento fue el siguiente: "C:\Documents and Settings\MODS\16-MAY-2010\Positive Control\Results.txt".

RESULTS OBTAINED FROM THE MODS PLATES ANALYSIS

POSITIVE CONTROL

WELL	SCORE	RESULT
1	0.940	POSITIVE
2	0.889	POSITIVE
3	0.149	NEGATIVE
4	0.241	NEGATIVE
5	0.962	POSITIVE
6	0.918	POSITIVE
7	0.941	POSITIVE
8	0.908	POSITIVE

POSITIVE CONTROL COMPLETED SUCCESSFULLY

Figura 2.27: Resultado modelo para placa de control positivo

El archivo de texto contiene la información del tipo de placa, resultados booleanos y de score para cada uno de los pozos, así como un diagnóstico general de la lectura de la placa. En este caso el test de placa de control positivo fue superado exitosamente, recordemos que cuando se encuentra una falla y no se supera el test de control, la falla es indicada al usuario en pantalla y el archivo de texto no resulta ser creado. Para el caso de placas con muestras de pacientes, el archivo de texto es algo más extenso, dado que debe contener información de cada uno de los pozos, así como el diagnóstico del paciente y el resultado del test de control negativo como se explicó en la sección correspondiente a análisis de placas de pacientes. A continuación se muestra el archivo de texto con los resultados finales de la lectura de una placa al azar con muestras de pacientes, con código B4815 analizada el día 16 de mayo del presente año. El tiempo de lectura fue de aproximadamente 17 horas en este caso y el archivo de almacenamiento fue el siguiente: "C:\Documents and Settings\MODS\16-MAY-2010\Analysis of Samples\Plate B4815\Results.txt".

Results Obtained From The MODS Plates Analysis

Negative Control

Well	Score	Result
1	0.120	Negative
2	0.151	Negative
3	0.026	Negative
4	0.190	Negative

Negative Control Completed Successfully

Patient 1

Well	Score	Result
1	0.219	Negative
2	0.254	Negative
3	-	-
4	-	-

No Evidence Of TB

Patient 2

Well	Score	Result
1	0.193	Negative
2	0.142	Negative
3	-	-
4	-	-

No Evidence Of TB

Patient 3

Well	Score	Result
1	0.891	Positive
2	0.913	Positive
3	0.932	Positive
4	0.882	Positive

Evidence Of MDR TB Has Been Found

Patient 4

Well	Score	Result
1	0.952	Positive
2	0.923	Positive
3	0.091	Negative
4	0.852	Positive

Evidence Of TB Susceptible To INH Has Been Found

Patient 5

Well	Score	Result
1	0.941	Positive
2	0.870	Positive
3	0.893	Positive
4	0.972	Positive

Evidence Of MDR TB Has Been Found

Figura 2.28: Resultado modelo para placa de muestras de pacientes

En el caso mostrado se puede observar que la placa superó la prueba de control negativo, que mientras que los pacientes 1 y 2 fueron diagnosticados negativos, los pacientes 3 y 5 fueron diagnosticados con TB MDR y el paciente 4 con TB susceptible a INH.

Capítulo 3

Discusión

En este capítulo se discute la calidad de los resultados obtenidos usando el equipo de lectura automática trabajando de manera integral, desde la adquisición de imágenes hasta el diagnóstico final.

3.1. Validación de MODS Plate Reader

El proceso de validación del sistema resulta de comparar los resultados obtenidos por el equipo con un cierto referente confiable. Para ello, se ha considerado pertinente realizar la validación tomando como referencia el diagnóstico de un experto en el método MODS. Para esta parte se procedió con la adquisición de una serie de fotografías de muestras de diversas placas de pacientes, tanto positivas como negativas, diagnosticadas como tal previamente por un experto en condiciones ideales de laboratorio. La toma de fotografías se llevó a cabo entre los meses de abril y junio del presente año. Del lote de fotografías capturadas, se ha considerado un total de 200 imágenes (100 positivas y 100 negativas) al azar para la validación del equipo, no dando ningún tipo de preferencia sobre la selección de alguna en particular. La integridad de las fotografías se capturaron empleando MODS Plate Reader y enfoque automático.

1. Validación con referencia a experto

Se le entregó al experto un lote de 200 fotografías aleatorias para el diagnóstico respectivo según su experiencia, sólo pudiendo valerse de la fotografía mostrada. Los resultados se resumen en la tabla 3.1.

Tabla 3.1: Resultados de validación con referencia a experto

Resultado esperado	Diagnóstico Negativo	Diagnóstico Positivo	Diagnóstico en Duda	Total
Negativo	98	0	2	100
Positivo	0	98	2	100
Total	98	98	4	200

La interpretación de los resultados de la tabla 3.1 es que del lote de 100 fotografías negativas, 98 fueron diagnosticadas correctamente y 2 fueron consideradas en duda por una baja calidad de enfoque de la imagen, consiguiéndose el mismo resultado para el lote positivo. Estos resultados son muy satisfactorios dado que:

- En ninguno de los dos casos se obtuvo un resultado diferente al esperado, esto es, diagnosticar como positivo algo negativo o viceversa.
- Se obtuvo un factor de sensibilidad de 98% y una especificidad de 98%.
- Un bajo margen de resultados dudosos, el cual no fue capaz de llevar a un diagnóstico equivocado.

El hecho de haber encontrado diagnósticos dudosos resulta directamente de haber considerado un juego de fotografías al azar (elegidas sin preferencia), entre las cuales había algunas bastante borrosas. A continuación, se mencionan algunas de las causas probables observadas en laboratorio, que pueden motivar que el algoritmo de enfoque automático no consiga capturar una imagen bien enfocada:

- Falta de alineamiento entre lámpara y microscopio
- Formación de películas en la superficie interna de la bolsa ziplock de protección
- Imperfecciones o manchas en la superficie externa de la bolsa
- Bolsa aglomerada demasiado cerca de o en contacto con la lente objetivo del microscopio

Estas son algunas de las causas observadas que complican la labor del algoritmo de enfoque, las cuales se espera puedan ser corregidas en un futuro para mejorar los resultados obtenidos.

2. Prueba con el software de reconocimiento

El software de reconocimiento de TB no se utilizó para validar el lector automático, debido a que al momento de las pruebas aún se encontraba en etapa de depuración. Sin embargo, se consideró conveniente incluir los resultados encontrados usando la versión del programa disponible en ese momento para analizar el mismo juego de fotografías que en el caso anterior. Los resultados se resumen en la tabla 3.2.

Tabla 3.2: Resultados obtenidos con el software de reconocimiento

Resultado esperado	Diag. Neg.	Score N/Prom.	Score N/Crít.	Diag. Pos.	Score P/Prom.	Score P/Crít.	Total
Negativo	99	0.152	0.424	1	0.748	0.748	100
Positivo	3	0.256	0.310	97	0.913	0.865	100
Total	102	-	-	98	-	-	200

Esta tabla tiene una distribución similar a la empleada en el caso anterior de validación por un experto, con la consideración adicional de incluir los valores de score característicos del diagnóstico con software: score promedio y score crítico (valor máximo para diagnóstico negativo y mínimo para diagnóstico positivo). Como referencia, en el momento de la prueba, el modelo logístico para diagnostico de TB tenía un valor de score umbral de 0.720.

El software de reconocimiento se desarrolló tomando como base la experiencia de los investigadores, por lo que se esperaba que la prueba fuera satisfactoria también. Efectivamente, se obtuvieron valores altos de sensibilidad (99%) y especificidad (97%). Sin embargo, del total de muestras analizado, se encontró 1 falso positivo (diagnóstico positivo cuando no hay TB) y 3 falsos negativos (diagnóstico negativo de un paciente enfermo).

Este alto número de falsos negativos se debe en parte a la limitación expresa del software para trabajar con fotografías de muestras de alrededor de 10 días, tiempo en el cual es posible el reconocimiento de cordones aislados. Para la prueba realizada, al haberse elegido fotografías al azar, se encontró que habían fotos de muestras que se encontraban fuera del rango de trabajo del software (varios días antes o después de cumplirse los 10 días de cultivo). Este hecho fue superado exitosamente por el experto, quien obtuvo los resultados esperados, quedando su diagnóstico, en el peor de los casos, como dudoso por falta de enfoque, no siendo así el caso del diagnóstico del software.

3.2. Desempeño de MODS Plate Reader

Una vez analizados los aspectos referentes a calidad de resultados obtenidos por el lector de placas, el siguiente punto a tocar es referente a aspectos prácticos de rendimiento, con miras a conseguir un equipo comercialmente viable. Si bien es cierto el análisis más relevante constituye el de validación del equipo, el cual consiste fundamentalmente de enfoque automático y diagnóstico acertado, existen otros aspectos importantes a tomar en cuenta como velocidad de operación, costo y tamaño del equipo, los cuales serán analizados a continuación.

Del análisis de los componentes de hardware relacionados con velocidad de operación, se llega a la conclusión de que se requiere un computador bastante poderoso en términos de procesador y acceso a memoria para poder realizar el control de MPR de manera óptima. A la fecha se ha estado trabajando con una computadora de las siguientes características: Intel Dual CPU E2160 @1.80 GHz, 1 GB DRAM, la cual lamentablemente se encontraba sobre saturada (utilización de CPU igual a 100%) en gran parte del proceso de lectura.

En términos de software, se debe admitir que el tiempo de procesamiento del algoritmo de reconocimiento resulta todavía muy extenso, a pesar de los esfuerzos por optimizarlo. En el caso de lectura masiva de muestras, este proceso constituye un cuello de botella bastante apreciable. Un segundo factor que contribuye con esto, es el proceso de enfoque automático, debido a que debe esperar necesariamente algunos milisegundos para que la muestra deje de vibrar por el continuo movimiento y a que por defecto se ha considerado una distancia bastante grande de búsqueda (alrededor de 1mm), esto con el objetivo de maximizar la posibilidad de encontrar el mejor plano de enfoque de cordones. Un tercer factor, en menor medida lo constituye la tarjeta DAQ, la cual podría ser reemplazada por una de mayor velocidad en un futuro.

Con respecto al control de movimiento, no podemos pasar por alto las deficiencias intrínsecas del sistema operativo empleado. Debe tenerse en consideración que el control de velocidades no es tan preciso al estar trabajando en un sistema operativo de uso ordinario. Esto es, Windows no asigna la prioridad necesaria al proceso de control de MPR y las llamadas a sistema toman un tiempo no determinístico de ejecución, con lo cual no se tiene una confianza de que el tiempo que se establece para una iteración realmente se esté cumpliendo. A fin de establecer realmente una velocidad; o indirectamente, el tiempo que dura una iteración de manera precisa, se requiere un sistema operativo en tiempo real (RTOS), el cual nos permitiría tener control absoluto de los tiempos de ejecución, así como la posibilidad de paralelizar la ejecución de programas, a saber movimiento X e Y de forma concurrente.

La evolución de este equipo requiere contar con hardware especializado para procesamiento de imágenes digitales, dado que un computador de propósito general no se encuentra optimizado para este tipo de tareas. Dadas las deficiencias actuales recién mencionadas, una alternativa interesante para el sistema de control de MPR se plantea de la siguiente manera:

- Unidades de microcontroladores (MCU) dedicadas para el control de cada uno de los motores de los posicionadores.
- Tarjeta GPU dedicada para el procesamiento de imágenes de reconocimiento. Mediante el empleo de la herramienta CUDA de nVidia es posible usar una extensión de lenguaje C para codificar algoritmos en GPUs del fabricante nVidia. CUDA explota las ventajas de las GPUs frente a las CPUs de propósito general utilizando el paralelismo que ofrecen sus múltiples núcleos, que permiten el lanzamiento de un altísimo número de hilos simultáneos, reduciendo considerablemente los tiempos de cómputo final.
- Empleo de una computadora de escritorio con razonable capacidad de procesamiento, memoria y alta capacidad de almacenamiento (del orden de TB) para las fotografías adquiridas. Esta se limitaría a servir como interfaz gráfica y unidad de almacenamiento en disco, así como enviar órdenes a los MCU sobre los set-points para los motores y a recibir los resultados de procesamiento de imágenes de la tarjeta GPU para el reporte final.

El costo estimado del equipo en la actualidad en términos de hardware empleado es de aproximadamente US$ 4000. La presencia de la tarjeta adicional de procesamiento, dependiendo de las características de su selección, incrementaría el costo considerablemente, en un estimado de US$ 1000. Finalmente, el tamaño del equipo se podría reducir con el empleo de posicionadores de menor distancia de trabajo, con las medidas mínimas para cubrir el área de una placa y mediante el empleo de la configuración L para la componente óptica, con ayuda de un espejo intermedio.

Capítulo 4

Conclusiones

El objetivo planteado de construcción de un equipo de lectura automática de placas MODS ha sido cumplido. Este lector no sólo realiza la lectura de placas como un sistema experto, sino que contribuye con un incremento considerable de la bioseguridad debido a que reduce la interacción de los operarios con las placas. A continuación, se presenta un resumen del contenido de este trabajo de tesis.

En el capítulo 1 se inició la descripción de la metodología MODS para lectura de placas de tuberculosis, dando a conocer sus características principales, beneficios e información de background relevante para el diseño del equipo MODS Plate Reader.

En el capítulo 2 se describió con bastante detalle los elementos de construcción del equipo en términos de software y hardware, destacando continuamente la estrecha relación de cada componente con los fundamentos del proceso de lectura. El capítulo finalizó con el procedimiento a seguir para lectura de placas, diagnóstico y control general del equipo, así como un formato propuesto para el reporte de los resultados obtenidos de muestras de pacientes.

En el capítulo 3 se realizó un análisis de validez de los resultados obtenidos por el equipo con referencia al diagnóstico de personal experto, así como una prueba con el software de reconocimiento, llegándose a determinar un alto grado de sensibilidad y especificidad en ambos casos. La sensibilidad alcanzada para el caso de experto (%) como referencia, fue ligeramente más alta que la que se alcanzó con el software de reconocimiento (%). Se observa un resultado similar, para el caso de la especificidad (% con operador experto y % con software de diagnóstico).

Finalmente, a pesar de pequeñas consideraciones que se puedan tener para mejorar el rendimiento del equipo a futuro y teniendo en consideración los resultados satisfactorios de validación, consideramos que el objetivo de construir un lector de placas automático basado en la metodología MODS ha sido cumplido a cabalidad.

Bibliografía

[1] "Guía del usuario MODS". Universidad Peruana Cayetano Heredia, 2008.

[2] Arias M, Mello FC, Pavon A, Marsico AG, Alvarado-Galvez C. "Clinical evaluation of the microscopic-observation drug-susceptibility assay for detection of tuberculosis". Clin Infect Dis 44: 674–680, 2007.

[3] Caviedes L, Lee TS, Gilman RH, Sheen P, Spellman E. "Rapid, efficient detection and drug susceptibility testing of Mycobacterium tuberculosis in sputum by microscopic observation of broth cultures". J Clin Microbiol 38: 1203–1208, 2000.

[4] Moore DA, Evans CA, Gilman RH, Caviedes L, Coronel J. "Microscopic-observation drug-susceptibility assay for the diagnosis of TB". N Engl J Med 355: 1539–1550, 2006.

[5] Moore DA, Mendoza D, Gilman RH, Evans CA, Hollm Delgado MG. "Microscopic observation drug susceptibility assay, a rapid, reliable diagnostic test for multidrug-resistant tuberculosis suitable for use in resource-poor settings". J Clin Microbiol 42: 4432–4437, 2004.

[6] Oberhelman RA, Soto-Castellares G, Caviedes L, Castillo ME, Kissinger P. "Improved recovery of Mycobacterium tuberculosis from children using the microscopic observation drug susceptibility method". Pediatrics 118: e100–106, 2006.

[7] Park WG, Bishai WR, Chaisson RE, Dorman SE. "Performance of the microscopic observation drug susceptibility assay in drug susceptibility testing for Mycobacterium tuberculosis". J Clin Microbiol 40: 4750–4752, 2002.

[8] Shiferaw G, Woldeamanuel Y, Gebeyehu M, Girmachew F, Demessie D. "Evaluation of Microscopic Observation Drug Susceptibility Assay for Detection of Multidrug-Resistant Mycobacterium tuberculosis". J Clin Microbiol 45: 1093–1097, 2007.

[9] Mirko Z, Velasco A, Comina G, Coronel J, Fuentes P, Luna C, Sheen P, Gilman R, Moore D. "Development of Low-Cost Inverted Microscope to Detect Early Growth of Mycobacterium tuberculosis in MODS Culture", PLoS ONE, 2010.

[10] Yazdanfar S, Kenny K, Tasimi K, Corwin A, Dixon E, Filkins R. "Simple and robust image-based autofocusing for digital microscopy", Virtual Journal for Biomedical Optics, Vol. 3, Iss. 7, Optical Society of America, 2008.

[11] Murphy D. "Fundamentals of light microscopy and electronic imaging". Editorial: Wiley-Liss, pp 1-13, 300-324, 2001.

[12] Hughes A. "Electric motors and drives: Fundamentals, types and applications". Editorial: Elsevier, pp 45-81, 305-339, 2006.

[13] Bolton W. "Instrumentation and Control Systems". Editorial: Elsevier, pp 74-79, 2004

[14] Mayne R. "Introduction to Windows and Graphics Programming with Visual C++ .NET". Editorial: World Scientific, pp 102-131, 2005.

[15] Romero A. "VirtualBox 3.1: Beginner's Guide". Editorial: Packt, pp 43-75, 107-138, 173-210, 2010.

[16] "LabJack U3 User's Guide". LabJack Corporation, 2008.

[17] "XVideoOCX Help Guide", Marvelsoft, 2005.