PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE BACILOSCOPIA AUTOMATIZADO DE CUATRO MUESTRAS DE ESPUTO PARA EL DIAGNÓSTICO DE TUBERCULOSIS

Tesis para optar al grado de Magíster en Ingeniería Mecatrónica

Presentado por:

JOAQUIN GONZALEZ VILLARREAL

ASESOR: Mg. Willy Eduardo Carrera Soria

Junio 2019

Lima - Perú

RESUMEN

En los últimos años se han estado desarrollando equipos médicos que automatizan los procesos que involucran el diagnóstico de tuberculosis (TBC), pues esta enfermedad continua siendo un problema de gran importancia. Según la organización mundial de la salud (OMS), la tuberculosis continúa siendo un problema de carácter global. En nuestro país se presentan alrededor de 27000 casos de TBC anualmente con una incidencia de 88.8 casos nuevos por cada 100 mil habitantes. Es así que el Perú aun ocupa el 1° lugar en América Latina en el número de casos de tuberculosis resistentes. El hacinamiento en los sectores desfavorecidos ha fomentado la transmisión de la tuberculosis. Este hacinamiento sumado a la falta de equipamiento de rápido diagnóstico accesible a las poblaciones más necesitadas ha dado lugar a que la enfermedad continúe siendo un problema. En el Perú el proceso de diagnóstico es completamente manual, la no existencia de un equipo de diagnóstico da lugar a gran variabilidad en los resultados, los cuales dependerán de la experiencia y concentración del personal de laboratorio. Es por ello que es necesario encontrar herramientas que permitan reducir la incidencia de la tuberculosis en nuestro país. Sin embargo, los sistemas de detección automatizados y de mayor confiabilidad poseen valores elevados en el mercado. La presente tesis propone el desarrollo de un sistema de baciloscopía automatizada de diagnóstico de TBC. La tesis involucra el re-diseño e implementación del sub-sistema de tinción de cuatro muestras y el diseño e implementación del sub-sistema de microscopia.

A mis padres que a lo largo de toda mi vida me han apoyado y apoyan. A mi esposa que me apoya en el día a día. A mi hija que me alegra los días.

AGRADECIMIENTOS

A mi asesor el Profesor Willy Carrera Soria por su apoyo, confianza y motivación para realizar la presente tesis.

A todos los miembros que han pertenecido al grupo SIBA durante el desarrollo de la presente tesis.

A mi familia por el apoyo incondicional que siempre me brindan.

A mi hermana Jessenia Gonzalez Villarreal quien me ayudó a documentar a través de fotos y videos las pruebas realizadas.

Al personal de la Sección de Electricidad y Electrónica por las facilidades dadas para usar los instrumentos de medición.

ÍNDICE DE CONTENIDO

Pág.

RESUMEN II

AGRADECIMIENTOS IV

ÍNDICE DE CONTENIDO V

ÍNDICE DE FIGURAS X

INTRODUCCIÓN 1

1. CAPÍTULO 1:
DIAGNÓSTICO DE LA PROBLEMÁTICA 4

1.1 La tuberculosis en el Perú 4

1.2 Aspectos socio-económicos de la población afectada. 8
1.2.1 Aspecto Social 8
1.2.2 Aspecto Económico 10

2. CAPÍTULO 2:
ESTADO DEL ARTE 13

2.1 Alternativas de Diagnóstico mediante baciloscopía 13

2.2 Alternativas de Diagnóstico mediante cultivo 15

2.3 Alternativas de Diagnóstico molecular. 16

3. CAPÍTULO 3:
PROBLEMÁTICAS Y COMPONENTES A MEJORAR DE LA MAQUETA FUNCIONAL SIBA (TINCIÓN) 20

3.1 Proceso de baciloscopia 20

3.2 Problemáticas de la maqueta funcional 21
3.2.1 Calentamiento de muestras 22
3.2.2 Alimentación 22
3.2.3 Dispensado de reactivos y agua 23
3.2.4 Desplazamiento de muestras 23

3.2.5 Interfaz de usuario 24

3.3 Problemas identificados en prototipos anteriores 24

3.4 Módulo de microscopía. 25

4. CAPÍTULO 4:
REDISEÑO DEL MÓDULO AUTOMÁTICO DE TINCIÓN DE CUATRO MUESTRAS DE ESPUTO 27

4.1 Alcances 27

4.2 Diagrama de bloques 28

4.3 Estructura mecánica del módulo de tinción 29
4.3.1 Área de inserción de muestras 30
4.3.2 Calentamiento de muestras 31
4.3.3 Dispensado de líquidos y expulsión de residuos 31
4.3.4 Plataforma móvil 31
4.3.5 Área de componentes electrónicos 32

4.4 Hardware Electrónico del módulo de tinción 33
4.4.1 Sensor de posición 33
4.4.2 Sensor de temperatura 34
4.4.3 Bombas y excitador de las bombas: 36
4.4.4 Actuador de calefacción y su excitador 39
4.4.5 Motor a pasos y su excitador: 41
4.4.6 Fuente de alimentación 42
4.4.7 Esquema de conexiones. 43

4.5 Software de la plataforma Electrónica Arduino para el módulo de tinción. 44
4.5.1 Subrutina de localización de referencia 45
4.5.2 Subrutina de inserción de muestras 46
4.5.3 Subrutina de posicionamiento 47
4.5.4 Subrutina de dispensado de reactivos 48
4.5.5 Subrutina de calentamiento 48
4.5.5.1 Descripción del control de temperatura de la maqueta funcional 48
4.5.5.2 Validación y rediseño del control de temperatura 49
4.5.6 Subrutinas para el control del motor a paso 51

5. CAPÍTULO 5:
DISEÑO E IMPLEMENTACIÓN DEL MÓDULO DE MICROSCOPÍA. 52

5.1 Alcances 53

5.2 Diagrama de bloques 53

5.3 Estructura mecánica 54

5.4 Diseño e implementación del módulo. 55
5.4.1 Hardware Electrónico: 55
5.4.1.1 Sensores y acondicionadores de señal 55
5.4.1.1.1 Cámara digital 55
5.4.1.1.2 Sensor de posición 57
5.4.1.2 Actuadores y excitadores 58
5.4.1.2.1 Led de potencia 58
5.4.1.2.2 Excitador del Led de potencia 59
5.4.1.2.3 Motores para el desplazamiento de las muestras 60
5.4.1.2.4 Excitadores de Motores para el desplazamiento de las muestras 61
5.4.1.3 Unidad de procesamiento 63
5.4.1.4 Pantalla táctil 63
5.4.1.5 Fuente de alimentación 64
5.4.2 Software: 65
5.4.2.1 Software para la plataforma Electrónica Arduino 65
5.4.2.2 Software para la plataforma Raspberry 66

5.5 Interfaz de usuario. 69

5.6 Software de autoenfoque 71

5.7 Implementación de un algoritmo de detección y diagnóstico (conteo) de patrones BAAR. 73

6. CAPÍTULO 6: RESULTADOS Y DISCUSIÓN 76

6.1 Resultados. 76
6.1.1 Resultados del módulo de tinción 76
6.1.1.1 Factores determinantes 76
6.1.1.1.1 Inclinación de las muestras 76
6.1.1.1.2 Tuberías de dispensado de fucsina: 77
6.1.1.1.3 Temperatura del calentamiento 78
6.1.1.1.4 Posiciones de las muestras 82
6.1.1.1.5 Oxidación de los recipientes 84
6.1.2 Resultados del módulo de microscopia 85
6.1.2.1 Plataforma móvil 85
6.1.2.2 Software de autoenfoque 85
6.1.2.3 Software de conteo de bacilos 87

7. CONCLUSIONES 89
- **Conclusiones del módulo de tinción: 89**
- **Conclusiones del módulo de microscopía: 89**

8. RECOMENDACIONES Y TRABAJOS FUTUROS 91

BIBLIOGRAFÍA 92

ANEXOS 95

A. Diagrama de circuito impreso (Módulo de tinción)..... 95

B. Programa de procesamiento de imagines para la obtención de valores de temperatura..... 95

C. Programa para Arduino(Módulo de Tinción)..... 100

D. Programa de autoenfoque diseñado en Matlab. 119

E. Programa de detección y conteo implementado en Python. 121

F. Programa para arduino (Modulo de microscopio). 125

G. Diagrama de circuito impreso (Módulo de Microscopio) 147

H. Control de temperatura para diversas posiciones del sensor de referencia del módulo de tinción. 148

I. Programa de la interfaz gráfica y del proceso de diagnóstico 149

- Diseño de la interfaz: 149

- Programa de la interfaz: 159

- Programa que recorre las muestras: 161

ÍNDICE DE TABLAS

Tabla 1.1: Casos nuevos e incidencia de tuberculosis por departamentos, Perú año 2013 y 2014. 5

Tabla 1.2: Tasa de morbilidad por TB en trabajadores de Salud de Hospitales de Lima y Callao, Año 2013- 2014. 7

Tabla 1.3: Panorama general de los costos entre el 2005 al 2010. 11

Tabla 1.4: Distribución de los gastos del manejo clínico según tipo. 12

Tabla 1.5: Costo promedio y costo basado en actividades para el análisis de una muestra en el diagnóstico de TBC. 12

Tabla 3.1: Tiempo estimado para un proceso automatizado de tinción. 21

Tabla 3.2: Problemas de interés encontrados en versiones anteriores del SIBA..... 25

Tabla 4.1: Características del sensor de posición fin de carrera..... 33

Tabla 4.2: Sensor LM35..... 35

Tabla 4.3: Selección de bombas..... 37

Tabla 4.4: Selección de micropasos (8A) y Control de giro y parada del motor a pasos(8B)..... 41

Tabla 4.5: Consumo de corriente..... 43

Tabla 5.1: Requerimientos y características de la cámara digital 56

Tabla 5.2: Requerimientos y características del sensor de posición fin de carrera..... 57

Tabla 5.3: Requerimientos y características del LED de potencia. 59

Tabla 5.4: Requerimientos y características de los motores a pasos. 61

Tabla 5.5: Control de giro y parada del motor a pasos. 62

Tabla 5.6. Características de la pantalla táctil..... 64

Tabla 5.7 Requerimientos de las fuentes de alimentación..... 65

Tabla 6.1: Posiciones del cabezal en milímetros respecto a la ubicación del sensor de fin de carrera. 83

Tabla 6.2: Variación de ml de fucsina respecto al intervalo de dispensado. 83

ÍNDICE DE FIGURAS

Figura 1.1: Sintomáticos respiratorios identificados e Incidencia de Tuberculosis. 6

Figura 1.2 : Red Nacional de Laboratorios de Salud Pública 8

Figura 1.3: Encuesta sobre la comunicación de los empleados sobre sus diagnósticos a sus jefes. 10

Figura 2.1: Panorama de los equipos desarrollado en base a métodos de diagnósticos por amplificación del ADN 17

Figura 3.1: Características de la Bomba - RS 360 RH 23

Figura 4.1: Diagrama de bloques del Subsistema de tinción 29

Figura 4.2: Disposición de las áreas del proceso en la estructura mecánica30

Figura 4.3: Interfaz mecánica con el usuario 30

Figura 4.4: Dispensado y desague de reactivos 31

Figura 4.5: Estructura de la plataforma movil..... 32

Figura 4.6: Soporte de componentes electrónicos 33

Figura 4.7: Interruptor de fin de carrera 33

Figura 4.8: Diagrama esquemático del circuito de sensado de posición 34

Figura 4.9 Respuesta de la temperatura medida en el sensor por la termocupla FLUKE 179 vs el sensor LM35. 35

Figura 4.10: Posición del sensor de temperatura..... 36

Figura 4.11: Respuesta del MOSFET IRF530 38

Figura 4.12: Excitadores de las bombas 38

Figura 4.13: Foco de luz halógena..... 39

Figura 4.14: Distancia entre la campana y las muestras (Y= 12.3mm) 39

Figura 4.15: Distancia entre centro del foco de luz halógena y la muestra (Y=34.1mm) 40

Figura 4.16: Vista inferior de la distribución de la campana y muestras. 40

Figura 4.17: Diagrama esquemático del detector de cruce por cero. 41

Figura 4.18: Diagrama esquemático del circuito de control de ángulo de disparo. 41

Figura 4.19: Esquemático del excitador del motor de paso 42

Figura 4.20: Fuente conmutada del sistema. 43

Figura 4.21: Esquema de conexiones..... 44

Figura 4.22: Diagrama de flujo general del proceso de tinción 45

Figura 4.23: Subrutina de localización de referencia 46
Figura 4.24: Subrutina de inserción de muestras..... 47
Figura 4.25: Diagrama de flujo de la rutina de posicionar plataforma 47
Figura 4.26: Dispensado de reactivos..... 48
Figura 4.27: Respuesta del sistema de calentamiento vs Sensor de referencia. 50
Figura 4.28: Control del calentamiento 51
Figura 4.29: Subrutinas del control del motor a paso..... 51
Figura 5.1: Diagrama de bloques del módulo de microscopía 54
Figura 5.2: Estructura o planta del módulo de microscopía 55
Figura 5.3: Cámara digital Pro C920 WEBCAM..... 56
Figura 5.4: Conexión entre la cámara Webcam y la Raspberry pi 3 56
Figura 5.5: Interruptor de fin de carrera 57
Figura 5.6: Diagrama esquemático del circuito de sensado de posición 58
Figura 5.7: Led de potencia 59
Figura 5.8: Diagrama esquemático del circuito excitador del LED de potencia 60
Figura 5.9: Motor a pasos XY42STH34-0354A..... 61
Figura 5.10: Esquemático del excitador del motor de paso del eje X 62
Figura 5.11: Esquemático del excitador del motor de paso del eje Z..... 62
Figura 5.12: Esquemático del excitador del motor de paso del eje Y 62
Figura 5.13: Conexión entre la tarjeta Arduino Nano y la Raspberry 3 63
Figura 5.14: Interfaz de usuario y su conexión con la Raspberry 64
Figura 5.15: Fuente switching del sistema. 65
Figura 5.16: Diagrama de flujo general del software del microcontrolador .. 66
Figura 5.17: Diagrama de flujo del proceso general 67
Figura 5.18: Diagrama de flujo del proceso de microscopía 68
Figura 5.19: Interfaz de usuario: Inicio del proceso. 69
Figura 5.20: Interfaz de usuario: visualización del teclado táctil. 70
Figura 5.21: Interfaz de usuario: búsqueda de registros. 70
Figura 5.22: Vista superior de la plataforma que contiene hasta 4 muestras. 71
Figura 5.23: Comparación del espectro para una misma imagen enfocada (Azul) y no enfocadas (rojo y verde). Se nota el cambio de las amplitudes en la porción de alta frecuencia del espectro. 72
Figura 5.24: Imágenes enfocadas y desenfocadas correspondientes al espectro mostrado en la figura 5.23. 72

Figura 5.25: Operación de eliminación de agujeros. 74

Figura 5.26: Etiquetado de objetos. a) Posterior al filtro de color, b) Posterior al filtro de puntos extremos, c) Posterior a la eliminación de agujeros 74

Figura 5.27: Imagen original e imagen final 75

Figura 5.28: Diagrama de flujo de detección y conteo de bacilos 75

Figura 6.1: Fracción de un campo de una muestra coloreada por el módulo de tinción..... 76

Figura 6.2: Tuberia de dispensado de fucsina 77

Figura 6.3: Muestra con fucsina proveniente del equipo (izquierda) y fucsina no contaminada (derecha) 78

Figura 6.4: Muestras teñidas con exceso de calentamiento (Izquierda) y Muestras teñidas con calentamiento adecuado (Derecha). 79

Figura 6.5: Respuesta del sensor de temperatura LM35 y temperaturas medidas en las muestras para una referencias de 60°. 79

Figura 6.6: Respuesta del sensor de temperatura LM35 y temperaturas medidas en las muestras para una referencias de 70°. 80

Figura 6.7: Temperatura del sensor de referencia durante el calentamiento 80

Figura 6.8: Temperatura del sensor de referencia durante el precalentamiento 81

Figura 6.9: Temperatura del sensor de referencia vs temperatura en la fucsina (prueba 1). 82

Figura 6.10: Temperatura del sensor de referencia vs temperatura en la fucsina (prueba 2). 82

Figura 6.11: Recubrimiento de fucsina de las muestras posterior al calentamiento..... 84

Figura 6.12: Oxidación del recipiente de fucsina 85

Figura 6.13: Resultados de enfoque normalizado para el primer grupo de imágenes 86

Figura 6.14: Resultados de enfoque normalizado para el segundo grupo de imágenes. 86

Figura 6.15: a) Imagen 10 de la primera secuencia de imágenes, Figura 6.15 b) Imagen 11 de la primera secuencia de imágenes. 87

Figura 6.16: Detección incorrecta de bacilos. 88

Figura 0.1: Diagrama esquemático. 95

Figura 0.2: Esquemático impreso del módulo de tinción. 147

Figura 0.3: Respuesta del sensor LM35 y termocupla con una 148

Figura 0.4: Respuesta del sensor LM35 y termocupla con una referencia de 70°C 148

Figura 0.5: Respuesta del sensor LM35 y termocupla con una referencia de 80°C 149

ET LUX IN TENEBRIS LUCET
MCMXVII

LISTA DE SÍMBOLOS

TBC : Tuberculosis
ZN : Ziehl Neelsen
NAAT : Nucleic acid amplification test
IDH : Índice de desarrollo humano
OMS : Organización mundial de la salud
SIBA : Sistema de baciloscopía automatizado

INTRODUCCIÓN

La tuberculosis es una enfermedad infectocontagiosa con gran índice de morbilidad y mortalidad a nivel mundial donde más del 95% de las muertes registradas a causa de esta enfermedad ocurren en países en vías de desarrollo.

En los últimos años se han estado desarrollando equipos médicos que automatizan los procesos que involucran el diagnóstico de tuberculosis (TBC), pues esta enfermedad continua siendo un problema de gran importancia. Según la organización mundial de la salud (OMS) la tuberculosis continúa siendo un problema de carácter global. En el año 2014 a nivel mundial se presentaron 9.6 millones de casos de los cuales 1,5 millones de personas fueron afectadas mortalmente. En el año 2017, en nuestro país se presentaron alrededor de 27578 casos nuevos de TBC con una incidencia de 88.8 casos nuevos por cada 100 mil habitantes (Organization, W. H. ,2018). Es así que, según el informe mundial sobre la tuberculosis de la OMS del 2016, el Perú aun ocupa el 1° lugar en América Latina en el número de casos de tuberculosis multidrogorresistente (MDR) y extremadamente resistente (XDR). El hacinamiento en los sectores desfavorecidos ha fomentado la transmisión de la tuberculosis. Este hacinamiento sumado a la falta de equipamiento de rápido diagnóstico accesible a las poblaciones más necesitadas ha dado lugar a que la enfermedad continúe siendo un problema. En el Perú el proceso de diagnóstico es completamente manual, la no existencia de un equipo de diagnóstico da lugar a gran variabilidad en los resultados, los cuales dependerán de la experiencia y concentración del personal de laboratorio. Es por ello que es necesario encontrar herramientas que permitan reducir la incidencia de la tuberculosis en nuestro país. Sin embargo, los sistemas de detección automatizados y de mayor confiabilidad poseen valores elevados en el mercado.

Los métodos de diagnósticos de TBC pueden ser clasificados en tres grupos: baciloscopía, cultivo, y detección genética directa en las muestras. Las técnicas de baciloscopía convencionales aunque no son las más sensibles son las más usadas debido a su rapidez, estas técnicas involucran una etapa de tinción y otra de microscopia. Es así que la tinción Ziehl Neelsen (ZN) es el método de tinción más usado en baciloscopía por ser efectivo y simple; además de ser el método de tinción recomendado por la OMS para los países en vías de desarrollo. En el Perú, el proceso de tinción está conformada por tres sub-procesos principales: en primer lugar se extiende la muestra sobre un porta-objetos (frotis), seguidamente se realiza la tinción de bacilos ácido-alcohol resistente (BAAR) el cual es realizado por el método ZN a base de fucsina fenificada, azul metileno, alcohol ácido y agua. Finalmente, para el diagnóstico, se examina la muestra teñida a través de un microscopio, donde se realiza el conteo de bacilos tuberculosos presentes en la muestra. Este proceso de diagnóstico de tuberculosis implica tiempo y exposición del personal encargado, quienes pueden realizar tan solo 70 diagnósticos de muestras por día.

El método de tinción Ziehl Neelsen es realizado de manera manual por expertos técnicos de laboratorio quienes inevitablemente realizan el proceso con amplia variabilidad, lo que puede afectar las actividades posteriores al diagnóstico. Los errores de estos diagnósticos pueden conducir a la no detección de pacientes con TBC (falsos negativos), quienes continuarán la cadena de transmisión en la comunidad. Adicionalmente, estas personas serán diagnosticadas tardíamente con posibles complicaciones. De otro lado los falsos positivos dan lugar a tratamiento inútiles de personas no tuberculosas. En el caso de la ausencia de tratamiento de personas tuberculosas sumadas al hacinamiento ocasiona la alta incidencia de la enfermedad en las zonas más pobres de la sociedad mientras que en el caso de los falsos positivos da lugar a que personas no tuberculosas sean tratadas innecesariamente causando daños en sus sistemas inmunológicos. Es por ello que la garantía o aseguramiento de un diagnóstico estandarizado, rápido y confiable es esencial.

En consecuencia el objetivo de la presente Tesis es rediseñar e implementar el sub-sistema automático de tinción para cuatro muestras de esputo en simultáneo y diseñar e implementar el sub-sistema de microscopía. A continuación se listan los objetivos específicos que serán desarrollados en los capítulos siguientes:

I. Rediseño del sistema mecánico del módulo de tinción.

II. Rediseño, implementación y/o modificación del hardware y software requerido de acuerdo a los cambios mecánicos realizados en el módulo automático de tinción.

III. Diseño e implementación del subsistema de desplazamiento y autoenfoque de muestras del módulo de microscopía.

IV. Implementación de un algoritmo de detección (conteo) de patrones BAAR.

V. Implementación de una interfaz de usuario.

En el capítulo 1 se presenta la problemática. En el capítulo 2 se desarrolla el estado del arte, mientras que en el capítulo 3 se presentan los problemas encontrados en versiones anteriores de los equipos desarrollados. Posteriormente, los objetivos I y II se abordan en el capítulo 4, mientras que los objetivos III, IV y V se abordan en el capítulo 5. Finalmente, en el capítulo 6 se concluye con las recomendaciones, observaciones y posibles trabajos futuros.

CAPÍTULO 1:
DIAGNÓSTICO DE LA PROBLEMÁTICA

En este capítulo se presenta la problemática general de la tuberculosis en el Perú. Se presentan algunos indicadores socioeconómicos que permiten justificar el estudio y la importancia de dar soluciones de rápido diagnóstico y de costo accesible.

1.1 La tuberculosis en el Perú

A nivel nacional el 57% de los enfermos de TBC se encuentran en las regiones de Lima y Callao, lo que quiere decir que el 43% restante de los casos se encuentra disperso en las demás regiones, correspondiendo un promedio del 2% por región. En los últimos años cinco departamentos (Madre de Dios, Ucayali, Loreto, Lima, e Ica) reportaron el 73% de todos los casos nuevos notificados en el país (Tabla 1.1). Es así que, la enfermedad se concentra principalmente en los departamentos de la costa central y la selva.

Tabla 1.1: Casos nuevos e incidencia de tuberculosis por departamentos, Perú año 2013 y 2014[1].

Departamento	N° Casos nuevos Año 2013	N° Casos nuevos Año 2014	Incidencia TB 2013	Incidencia TB 2014	% de casos nuevos TB 2013	% de casos nuevos TB 2014	% de casos TB nuevos acumulado 2013	% de casos TB nuevos acumulado 214
Perú	27505	27350	90.3	88.8				
Madre de dios	234	270	178.8	201.3	0.9%	0.8%	0.9%	0.8%
Lima*	16265	16618	154.6	155.5	59.1%	60.7%	60.0%	61.5%
Tacna	458	437	137.4	127.2	1.7%	1.6%	61.6%	63.1%
Ucayali	771	598	159.4	122.1	2.8%	2.3%	64.4%	65.4%
Loreto	1272	1113	124.9	108.2	4.6%	4.2%	69.0%	69.6%
Ica	751	761	97.3	97.6	2.7%	2.9%	71.8%	72.4%
Moquegua	144	141	81.5	78.9	0.5%	0.5%	72.3%	73.0%
La libertad	1257	1245	69.3	68.0	4.6%	4.6%	76.9%	77.5%
Arequipa	771	766	61.2	60.4	2.8%	2.8%	79.7%	80.3%
Lambayeque	825	721	66.5	57.9	3.0%	2.6%	82.7%	82.9%
Junin	792	711	59.5	54.9	2.9%	2.6%	85.5%	85.5%
San Martin	333	436	40.7	52.6	1.2%	1.6%	86.8%	87.2%
Ancash	628	584	55.3	51.1	2.3%	2.1%	89.0%	89.3%
Tumbes	83	116	35.9	49.4	0.3%	0.5%	89.3%	89.8%
Huanuco	396	411	46.7	48.1	1.4%	1.4%	90.8%	91.2%
Cusco	697	603	53.6	45.4	2.5%	2.2%	93.3%	93.4%
Ayacucho	281	246	41.7	36.1	1.0%	0.9%	94.3%	94.3%
Pasco	85	96	28.4	31.8	0.3%	0.3%	94.6%	94.6%
Puno	441	444	31.7	31.7	1.6%	1.7%	96.3%	96.3%
Piura	514	501	28.3	27.4	1.9%	1.8%	98.1%	98.1%
Amazonas	92	114	21.9	27.1	0.3%	0.4%	98.5%	98.5%
Huancavelica	89	108	18.3	22.0	0.3%	0.3%	98.8%	98.9%
Apurímac	106	97	23.3	21.2	0.4%	0.4%	99.2%	99.2%
Cajamarca	220	213	14.5	14.0	0.8%	0.8%	100.0%	100.0%

En la Revista Peruana de Medicina Experimental y Salud Publica (**Laboratorio de Referencia Nacional de Micobacterias, 2017)** se menciona que las personas que requieren ser diagnosticadas o pacientes con sintomáticos respiratorios (SR) han aumentado sostenidamente. Esto muestra que si bien la tasa de incidencia se está reduciendo, las personas a quienes se les realiza el diagnóstico han aumentado. Es así que, en los últimos 5 años los identificados con SR han alcanzado su máximo valor en el 2015 con 1’774,000 SR identificados a pesar de que en este mismo año

[1] Fuente: Informes operacionales ESNPCT-DGSP/MINSA - Vigilancia epidemiológica de TB-MINSA/DGE Perú

se registró la tasa más baja en los últimos 25 de incidencia con 87.6 casos nuevos de TB por cada 100 mil habitantes.

En la figura 1.1 se muestran las tendencias mencionadas:

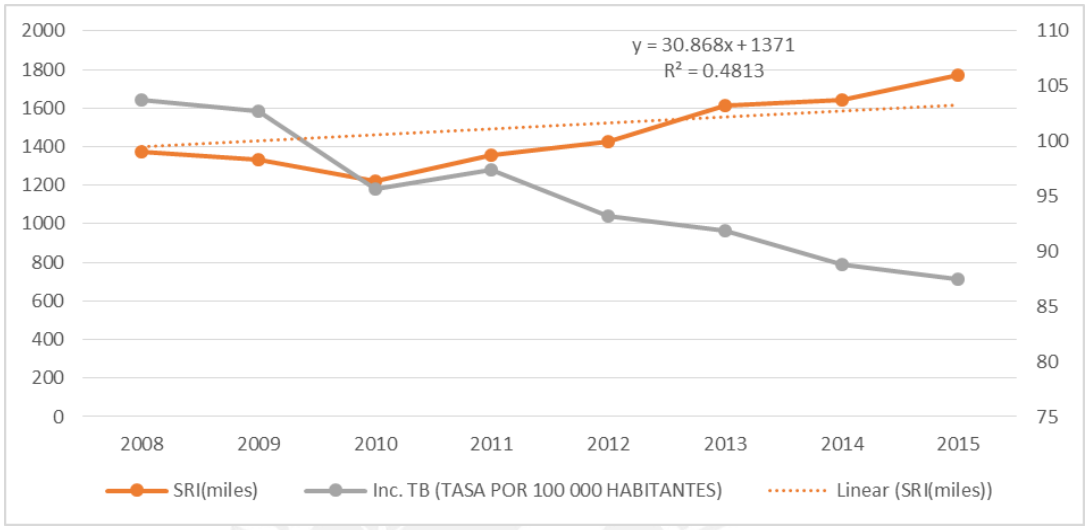


Figura 1.1: Sintomáticos respiratorios identificados e Incidencia de Tuberculosis. [2]

Adicionalmente es importante mencionar que Lima no solo es el departamento con más casos de tuberculosis en el país, sino también es el tercer departamento después de Madre de Dios y Ucayali con la incidencia más alta (Tabla 1.1); el mayor porcentaje de casos se concentran en la provincia metropolitana de Lima. Los distritos que reportan tasas más altas de TB pulmonar frotis positivos (TBPFP), es decir por encima de 100 casos por cada 100 mil habitantes, son San Juan de Lurigancho, Rímac, La Victoria, El Agustino, Ate, San Anita y Barranco. Es claro que los hospitales que se encuentran alrededor de estos distritos serán los que presentan la mayoría de los casos. Esto se ve reflejado en los datos notificados en el año 2013 y 2014, donde se estimó una tasa de morbilidad por TBC en trabajadores de salud (TS) en 12 hospitales de Lima y Callao (Tabla 1.2); siendo los Hospitales Emergencias Grau de EsSalud, Hospital Cayetano

[2] Fuente: Revista Peruana de Medicina Experimental y Salud Publica Junio 2017

Heredia, Arzobispo Loayza y Dos de mayo los de mayor tasa para el 2013; para el año 2014 fueron los hospitales Hipólito Unanue, Cayetano Heredia (3.3), Dos de Mayo y Sergio Bernales (2.7).

Tabla 1.2: Tasa de morbilidad por TB en trabajadores de Salud de Hospitales de Lima y Callao, Año 2013- 2014. [3]

Nombre de Hospitales	Institución	Casos TB			N° de TS		Tasa*1000	
		2013	2014	Total	2013	2014	2013	2014
Hosp. Cayetano Heredia	MINSA	15	8	**23**	2296	2457	6.5	3.3
Hosp. Edgardo Rebagliati Martins	ESSALUD	9	12	**21**	5374	5629	1.7	2.1
Hosp. Arzobispo Loayza	MINSA	11	7	**18**	2779	3149	4.0	2.2
Hosp. Dos de Mayo	MINSA	10	7	**17**	2575	2428	3.9	2.9
Hosp. Hipólito Unanue	MINSA	6	10	**16**	2372	2271	2.5	4.4
Hosp. Guillermo Almenara Irigoyen	ESSALUD	10	7	**17**	3526	3698	2.8	1.9
Hosp. Daniel A. Carrion - Callao	MINSA	7	6	**13**	2030	2122	3.4	2.8
Hosp. Emergencias Grau	ESSALUD	8	3	**11**	960	1137	8.3	2.6
Hosp. Sergio Bernales	MINSA	3	4	**7**	1419	1477	2.1	2.7
Hosp. Alberto Sabogal Soluguren	ESSALUD	3	2	**5**	2030	1821	1.5	1.1
Hosp. Maria Auxiliadora	MINSA	0	4	**4**	1928	1964	0.0	2.0
Hosp. San Jose - Callao	MINSA	2	2	**4**	781	812	2.6	2.5
Total Nacional		**226**	**241**	**467**	**223803**	**241732**	**1.0**	**1.0**

En consecuencia, es muy conveniente el hecho de poder trabajar en base a las muestras del Hospital Dos de Mayo pues es uno de los hospitales donde se presentan gran cantidad de casos debido a su particular ubicación rodeada de los distritos donde se presentan la mayor incidencia y cantidad de casos. Adicionalmente, es importante mencionar que el equipo al ser portátil tendrá la posibilidad de ser transportado a las diferentes regiones del País. Sin embargo, experimentalmente será validado en el Hospital Dos de Mayo.

Finalmente, es necesario indicar que existe una relación de laboratorios de referencia (Red Nacional de Laboratorios de Salud Pública) en los que se realizan los análisis de baciloscopía y podrían usar experimentalmente el

[3] Fuente y elaboración: Vigilancia epidemiológica de TB- DGE/MINSA, Base de datos de Recursos Humanos- Observatorio de Recursos Humanos en Salud DGRH/MINSA

sistema automatizado. Esta red de Laboratorios está conformada por 25 Laboratorios de Referencia Regional (LRR) y 04 Laboratorios Referenciales (LR) (Ministerio de Salud, I. N. S. , 2014, MINSA - I. N. S, 2018). Además, se cuenta con un Laboratorio de Referencia Nacional (LRN), el cual realiza pruebas de diagnóstico de tuberculosis mediante el método automatizado en cultivo líquido BACTEC MGIT-960. En la figura 1.2 se muestra la distribución de la red de laboratorios a lo largo del país.

Figura 1.2 : Red Nacional de Laboratorios de Salud Pública [4]

1.2 Aspectos socio-económicos de la población afectada.

1.2.1 Aspecto Social

Respecto al aspecto social, en nuestro país los estratos sociales más bajos son los más afectados. A pesar de que las provincias con mayor índice de

[4] Fuente: Portal de la Red de Laboratorios en Salud Pública (portal.ins.gob.pe.).

desarrollo humano (IDH) son las que presentan la mayor incidencia de casos de tuberculosis, el hacinamiento de los casos en determinados distritos evidencia la desigualdad social y su impacto en la carga de tuberculosis. De tal forma que en los distritos de Lima con mayores desventajas respecto a IDH son los que tienen mayor incidencia de tuberculosis. Concentrándose el 50% de la carga de tuberculosis de la provincia de Lima en el 40% de los distritos más desfavorecidos en términos de IDH, y solo el 10% en el 20% de los distritos con mayor IDH (Ministerio de Salud, Febrero, 2016).

Adicionalmente, se ha podido notar que padecer de tuberculosis genera en nuestro país estigma y discriminación. La percepción de los pacientes de la discriminación o estigmatización puede dar lugar a que en algunos casos no quieran comunicar su condición a sus familiares o amigos. Incluso algunos pacientes han mencionado que para evitar ser estigmatizados o discriminados han descontinuado el tratamiento (Raúl Alonso Timana Ruiz, 2014). En las encuestas aplicadas en el informe Impacto Socioeconómico de La Tuberculosis en el Perú del 2010 a los pacientes con TB se encontró que casi la mitad de los pacientes con TB sensible le han comunicado de su enfermedad a sus jefes y a sus compañeros de trabajo calificando el trato recibido de ellos como un trato que denotó comprensión. La figura 1.3 muestra que la mayoría de pacientes ha preferido no comunicar su situación a su empleador. Los que si comunicaron tuvieron en la mayoría de casos una respuesta positiva.

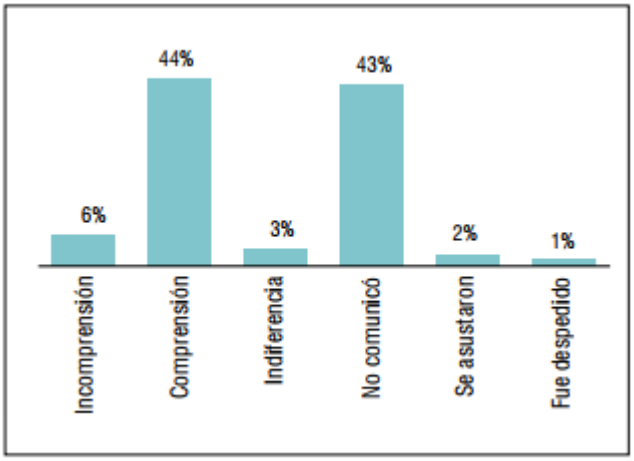


Figura 1.3: Encuesta sobre la comunicación de los empleados sobre sus diagnósticos a sus jefes. [5]

1.2.2 Aspecto Económico

El estudio del impacto socioeconómico realizado por el MINSA (Raúl Alonso Timana Ruiz, 2014) concluye que el costo económico de la tuberculosis en el Perú para el 2010 ascendió a 80,087 millones de US$, correspondiendo al costo directo a 42,120 millones de US$ (52.6%) y un costo indirecto a 37,968 millones de US$ (47.4%). El costo directo es financiado principalmente por el Estado (19,933 millones) que considera los aportes de la seguridad social y el financiamiento de los cooperantes. Sin embargo, el costo para las familias representa una parte importante del costo económico total, 20,629 millones de US$ (25,7%), constituido por los costos directos o gastos de bolsillo y los costos indirectos, que incluyen los años de vida perdidos por discapacidad, el costo de los días dejados de trabajar o estudiar y el costo del tiempo de los familiares o voluntarios. A esto se suma el costo para la sociedad compuesta por los años de vida perdidos por muerte (costo indirecto) de 23,544 millones de US$ (29.4%). En la Tabla 1.3 se puede apreciar el panorama general de los costos entre el 2005 y 2010.

[5] Fuente: MINSA, Informe del Impacto Socioeconómico de La Tuberculosis en el Perú del 2010

Tabla 1.3: Panorama general de los costos entre el 2005 al 2010.[6]

	2005	2006	2007	2008	2009	2010	Total	Promedio	2005 - 2010
								Monto	%
Costo directo	31 676	31 317	30 383	35 430	34 926	42 120	205 852	34 308	50,2%
Gastos del Estado	14 648	9 133	11 964	16 229	13 868	19 933	85 775	14 296	20,9%
Aportes seguridad social	4 821	2 719	3 335	4 119	3 874	4 168	23 036	3 839	5,6%
Financiado cooperantes	8 479	13 336	9 737	8 556	9 970	11 813	61 891	10 315	15,1%
Gastos de bolsillo[1/]	3 728	6 129	5 347	6 526	7 214	6 206	35 150	5 858	8,6%
Costo indirecto	34 481	31 056	31 613	35 191	34 270	37 967	204 578	34 096	49,8%
Mortalidad	22 417	19 924	19 637	22 489	21 646	23 544	129 857	21 643	31,6%
Discapacidad	4 438	3 342	3 492	3 778	3 588	4 782	23 420	3 903	5,7%
Días dejados de trabajar	6 920	7 030	7 590	7 955	7 851	8 589	45 935	7 656	11,2%
Tiempo de familiares	706	760	894	969	985	1 052	5 366	894	1,3%
Total	66 157	62 373	61 996	70 621	69 196	80 087	410 430	68 404	100%

El proyecto de Finanzas y gobernanza en salud (HFG) de la Agencia de los Estados Unidos para el Desarrollo Internacional (USAID), ha realizado un informe en el 2014 en el cual se concluye, a partir de una evaluación económica parcial de tipo costo de enfermedad, que el costo total aproximado para TBC es de 27’443,865 dólares, de los cuales corresponde para TBC Pulmonar 23’666,252 de dólares, TBC Extrapulmonar 1’501,742 dólares, TBC con complicaciones 935,552 dólares y para TBC Multidrogoresistente 1,340,319 dólares. Los 27’443,865 de dólares representan el 14.1% del presupuesto ejecutado del año 2014 en el Programa Presupuestal 016 TBC –VIH/SIDA.

En la tabla 1.4 se puede apreciar que el costo total correspondiente a diagnóstico es 1’302,884 de dólares (4.7 %), tratamiento 24’205,776 de dólares (88.2%) y para seguimiento 1’935,206 de dólares (7.1%). La población de estudio fue en base a pacientes con TBC afiliada al Seguro Integral de Salud (SIS).

[6] Fuente: Base de datos del estudio – Gastos en tuberculosis 2005-2010

Tabla 1.4: Distribución de los gastos del manejo clínico según tipo.[7]

Condición Específica	Población Objetivo	Costo Diagnóstico	Costo Tratamiento	Costo Seguimiento	Costo total
Tuberculosis pulmonar	11,888	926,726	20,902,936	1,836,590	23,666,252
Tuberculosis extrapulmonar	1,180	240,207	1,239,134	22,401	1,501,742
Tuberculosis con complicaciones	559	114,079	751,967	69,505	935,552
Tuberculosis multidrogoresistente	180	21,871	1,311,739	6,709	1,340,319
Total	**13,808**	**1,302,884**	**24,205,776**	**1,935,206**	**27,443,865**
		4.7%	**88.2%**	**7.1%**	**100.0%**

Finalmente, A continuación se presenta el resultado de una evaluación por costos promedio y costeo en base al método ABC (Activity Based Costing) realizado en un hospital de Minas Gerais - Brazil (I. de Almeida, 2017), para diferentes métodos de diagnóstico (baciloscopía, cultivo y diagnóstico a través del ADN). En cuanto al método ABC se realiza considerando una actividad como el común denominador para el cálculo del costo unitario por actividad, en lugar de la cantidad real consumida. El resultado será el costo por análisis pero basado en costeo por actividades. El costo promedio y el costo basado en la actividad se muestran en la Tabla 1.5. Se puede apreciar que no solo el costo del dispositivo médico es alto sino también el costo por análisis de muestra.

Tabla 1.5: Costo promedio y costo basado en actividades para el análisis de una muestra en el diagnóstico de TBC.[8]

Método de diagnóstico	Costo Promedio	Costeo basado en las actividades
Baciloscopia centrifugada de Ziehl Neelsen (ZN) y Auramina (AU)	U $ 10.06	U $ 5.61
Baciloscopia directa por ZN	U $ 7,42	U $ 4,15
Cultivo en un medio sólido Loweinstein-Jensen	U $ 27.38	U $ 16.50
kit Detect TB®LabTest	U $ 115.74	U $ 73.46

[7] Fuente: Proyecto HFG-Perú, USAID

[8] Fuente: Evaluation of the Mean Cost and Activity Based Cost in the Diagnosis of Pulmonary Tuberculosis in the Laboratory Routine of a High-Complexity Hospital in Brazil.

CAPÍTULO 2:
ESTADO DEL ARTE

En el presente capítulo se expone una breve descripción de los métodos de diagnóstico de tuberculosis. Se realiza una revisión y descripción breve de los equipos de diagnóstico automatizado de tuberculosis que actualmente están siendo desarrollados y los que están siendo introducidos al mercado. Los equipos comerciales de diagnóstico en la mayoría de los casos han sido enfocados en técnicas de detección genética; no obstante, existen diversas iniciativas en proceso de desarrollo de prototipos basados en técnicas de cultivo y baciloscopía. En las siguientes líneas se hará una revisión de las iniciativas y métodos existentes respecto al equipamiento de diagnóstico por detección genética, baciloscopía y cultivo, poniendo una particular atención a los equipos que se encuentran disponibles en nuestro país.

2.1 Alternativas de Diagnóstico mediante baciloscopía

Una alternativa a la Baciloscopía con tinción ZN es la microscopía por fluorescencia con tinción Auramina-Rodamina, que obtiene mejores resultados de sensibilidad y especificidad con respecto al método de tinción ZN. Sin embargo, debido al elevado costo que se requiere para adquirir un microscopio de fluorescencia (microscopio con una lámpara de vapor de mercurio), se continúa optando por la tinción ZN en los países en vías de desarrollo (González-Martin, 2017).

Actualmente existen herramientas de tinción automatizadas y procesadores de portaobjetos a fin de obtener un alto rendimiento y la tinción adecuada de las muestras. El equipo RAL Diagnostics (socio de bioMérieux, Francia) es un ejemplo, este ofrece un sistema de tinción automático para el método convencional ZN o los métodos asociados a microscopía fluorescente. El producto ofrece una consistente tinción a través de la automatización. La máquina es aplicable en centros de microscopía de alto rendimiento y en instalaciones más pequeñas a fin de reducir el tiempo dedicado al preparado de muestras. La tinción automatizada ha demostrado ser un factor en el ahorro de recursos significativos y la mejora de los resultados de tinción en laboratorios con una gran carga de trabajo. El equipo de tinción RAL STAINER procesa 10 muestras al mismo tiempo. Todos los reactivos de tinción se encuentran dentro de los depósitos de reactivos en la máquina. El aire interno se filtra para prevenir la liberación de productos químicos. La herramienta se comercializa desde el 2013 y cuesta alrededor de US $15,000. El costo por muestra teñida en tinción ZN y para reactivos de auramina es de US $ 0,30 y US $ 0,50 respectivamente.

El Aerospray® TB Series 2 de ELITechGroup (Suiza) es otra plataforma de tinción automatizada que también se puede usar para métodos de tinción convencionales o fluorescentes. Las diapositivas se preparan a través de reactivos aplicados desde las boquillas atomizadoras, limitando los volúmenes utilizados. Esta plataforma puede procesar por lotes ya sea 12 o 30 diapositivas y requiere cinco minutos de tiempo de procesamiento. Según el ELITechGroup el sistema de 30 diapositivas puede procesar hasta 400 diapositivas por hora.

La Sección de Electricidad y Electrónica de la facultad de Ciencias e Ingeniería de la Pontificia Universidad Católica del Perú ha estado desarrollando sistemas de preparación automática de muestras de esputo para el diagnóstico de TBC (PAME) a fin de automatizar el proceso de tinción ZN. En versiones anteriores, la tinción se realizaba para una sola muestra por proceso, lo que hace que el tiempo de análisis de varias

muestras sea excesivo; posteriormente, en otra versión se consideró un sistema que realice la preparación de siete muestras en simultáneo, lo cual involucra que este sea voluminoso y pesado para formar parte de un equipo portátil de diagnóstico de TBC , ya que se sabe que a nivel nacional el 54% de los enfermos de TBC se encuentran en las regiones de Lima y Callao, lo que quiere decir que el 46% restante de los casos se encuentra disperso en las demás regiones, correspondiendo un promedio del 2% por región. En consecuencia, actualmente el grupo que desarrolla el Sistema de Baciloscopía Automatizado (SIBA) cuenta con un prototipo funcional cuyo objetivo es que sea utilizado en todas las regiones de nuestro país. El mencionado prototipo de diagnóstico de TBC, cuenta con un módulo de tinción de 4 muestras en simultáneo y un módulo microscopia de 4 muestras.

2.2 Alternativas de Diagnóstico mediante cultivo

Según la revisión de la organización internacional Unitaid (Unitaid, 2017), no hay nuevas actualizaciones con respecto al diagnóstico basado en el cultivo de TBC. Debido a los altos riesgos asociados con la infección en el laboratorio en comparación con los métodos que simplemente manipulan el material infectado (pe. baciloscopía). Por lo tanto, la mayoría de los cultivos se realizan en instalaciones de referencia o en algunos laboratorios intermedios con instalaciones apropiadas de bioseguridad y personal capacitado. Si bien el cultivo puede tomar mucho tiempo para generar diagnóstico (generalmente hasta 4 semanas), es un ensayo altamente sensible, que también se puede usar para una posterior determinación de la resistencia a los medicamentos en los casos de cultivo positivo. Actualmente, tres fabricantes ofrecen sistemas automatizados en cultivo liquido: el equipo BacT / ALERT® 3D de bioMérieux (Francia); la plataforma basada en un tubo indicador de crecimiento micobacteriano BACTEC ™ (MGIT ™) de BD (EE. UU.); y la plataforma Mycolor TK de Salubris (Turquía).

Además de los sistemas comerciales de cultivo líquido se encuentra el método MODS (microscopic observation drug susceptibility assay) de menor costo y no comercial, que también permite realizar ensayos de resistencia a los medicamentos. El método MODS es una técnica basada en metodología de cultivo. Se basa en un cultivo en medio líquido middlebrook 7H9 monitorizado a través de la observación de un microscopio invertido. Se observan la formación de cuerdas, que son visibles antes que las colonias, en una media de 8 días. Tecnológicamente esta es una opción barata y aplicable a países de alta incidencia y bajos recursos (González-Martin, 2017). En el Perú, la Universidad Peruana Cayetano Heredia (UPCH) y la Universidad Nacional de Ingeniería (UNI) han desarrollado un sistema MODS Plate Reader que puede diagnosticar la tuberculosis multidrogo resistente (TB-MDR) en menos de quince segundos; no obstante, los métodos de cultivo requieren por lo menos de 2 a 3 semanas para amplificar la muestra . El sistema contempla un lector de placas automatizado de observación microscópica para cultivos MODS. El lector maneja automáticamente las placas MODS y después de ejecutar el algoritmo de autoenfoque se obtienen imágenes digitales de las estructuras microscópicas características de Mycobacterium tuberculosis. Una vez obtenida la imagen se realiza el diagnóstico a través de un software de reconocimiento de patrones MODS. De esta forma el lector automatizado reduce el tiempo de trabajo y el manejo de cultivos de M. tuberculosis por personal de laboratorio (COMINA, 2011).

2.3 Alternativas de Diagnóstico molecular.

La aplicación de NAAT (En inglés: nucleic acid amplification test) ha revolucionado las pruebas de diagnóstico rápidas y precisas para la mayoría patógenos. La tecnología generalmente implica un equipo PCR para la amplificación de una sección específica de ADN genómico o ARN celular de un patógeno. El método de detección que identifica la presencia del material genético patógeno varía según los productos; en muchos casos se usa fluorescencia, mientras que en pruebas más sencillas se usan luminiscencia

o se realiza a través de la identificación visual de líneas en la captura de las amplificaciones. Las NAAT se usan con frecuencia para identificar alelos genéticos conocidos que confieren resistencia a los medicamentos.

En la figura 2.1 se muestra el panorama de desarrollo de estos equipos comerciales, los instrumentos marcados de verde son productos que cuentan con la recomendación de la OMS. Los equipos que se encuentra en la primera fila son diseñados para laboratorios de referencias, los de la segunda para laboratorios intermedios y los de la última para laboratorios básicos.

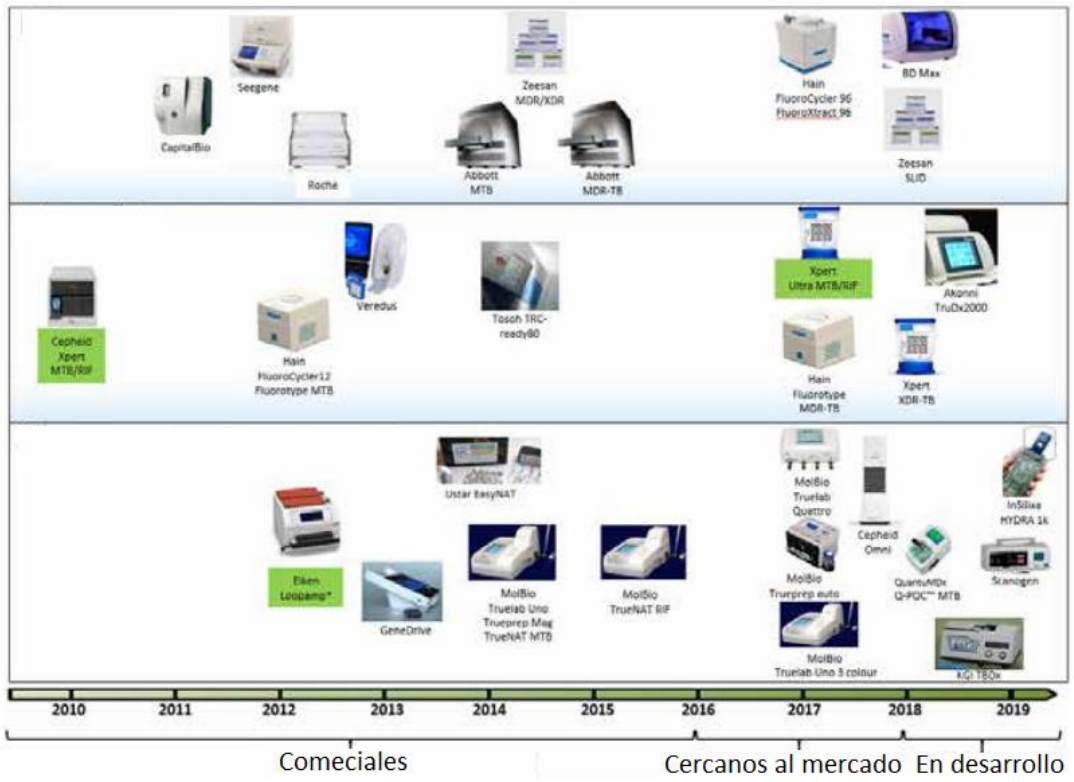


Figura 2.1: Panorama de los equipos desarrollado en base a métodos de diagnósticos por amplificación del ADN[9]

En nuestro país las instituciones que cuentan con equipos de diagnóstico rápido de tuberculosis por análisis molecular se muestran en la tabla 2.1.

[9] Diagnostics Technology Landscape 5th Edition, May 2017

Recientemente el hospital ESSALUD Hospital III de Emergencias Grau adquirió el Gx4 Xpert MTB / RIF. En el caso del INS este realiza pruebas moleculares GenotypeCM y GenotypeAS (Ministerio de Salud, I.N.S., 2018).

Tabla 2.1: Instituciones que cuentan con equipos de diagnóstico molecular para TBC

TIPO DE INTITUCIÓN	INTITUCIÓN	EQUIPO
ESSALUD	Hospital III de Emergencias Grau	Gx4 Xpert MTB / RIF
INS	Laboratorio de Referencia Nacional de Biotecnología y Biología Molecular	GenoType

La empresa Cepheid, Inc. (Sunnyvale, California, EE.UU.) desarrolló la técnica Xpert MTB/RIF, que es un test automatizado para el diagnóstico de tuberculosis basado en la detección de ácidos nucleicos específicos del bacilo de Koch, con cartuchos en una muestra orgánica. A diferencia de las pruebas convencionales de amplificación de ácidos nucleicos (NAAT), el Xpert MTB/RIF simplifica la identificación de ADN micobacteriano, automatizando los 3 procesos requeridos para pruebas moleculares basadas en PCR: extracción, amplificación y detección (VALLEJO V).

La técnica Xpert MTB / RIF es una prueba que puede detectar el complejo de M.tuberculosis y la resistencia a la rifampicina dentro de las dos horas posteriores de iniciada la prueba, con una intervención técnica mínima. Este método cuenta con una muy alta sensibilidad y especificidad. Adicionalmente, el reactivo de la muestra utilizado para licuar el esputo tiene potentes propiedades desinfectantes (con capacidad de matar las bacterias de la tuberculosis) y elimina en gran medida los problemas de bioseguridad durante el procedimiento de prueba. Estas características permiten que la tecnología sea extraída de un laboratorio de referencia y utilizada más cerca del paciente.

El equipo de Xpert MTB / RIF requiere de una fuente de alimentación eléctrica ininterrumpida y estable, control de temperatura y calibración anual de los módulos de cartucho (N. Karen R Steingart). El Xpert tiene una alta especificidad para el diagnóstico de tuberculosis, que en la serie evaluada por el Instituto Nacional del Tórax de Chile fue de 95% en las muestras respiratorias y 94% en las no respiratorias (considerando como standard la positividad del cultivo). Considerando el costo de este examen, no debería ser indicado a todos los pacientes a quienes se les solicita una baciloscopía, sino sólo en aquellos con sospecha fundada de tuberculosis en los cuales la baciloscopía sea negativa. Al ser un método muy sensible no está indicado para enfermos que hayan tenido una tuberculosis, porque por ser una técnica de PCR, cualquier residuo de DNA micobacteriano que haya quedado del pasado será amplificado muchas veces y puede dar un resultado falso positivo (VALLEJO V). El dispositivo más usado es el GX4, que cuenta con 4 módulos (permite realizar 16-20 pruebas por turno de 8 horas), cuyo costo con descuento para países en vías de desarrollo asciende a aproximadamente a US$ 17 000 FOB (Pan American Health Organization, 2017).

CAPÍTULO 3:
PROBLEMÁTICAS Y COMPONENTES A MEJORAR DE LA MAQUETA FUNCIONAL SIBA (TINCIÓN)

La sección de electricidad y electrónica de la PUCP se encuentra desarrollando el Sistema de Baciloscopía Automatizado (SIBA), el cual cuenta con una maqueta funcional para el módulo de tinción, en esta maqueta se han identificado problemas y componentes que pueden ser mejorados. Se realiza la descripción actual de la maqueta con sus problemas y componentes a mejorar.

En las siguientes líneas se presentan los problemas encontrados en la maqueta. Para ello, primero se realiza una breve revisión del proceso de tinción por ZN en baciloscopía.

3.1 Proceso de baciloscopia

El proceso de diagnóstico mediante baciloscopía está conformado por tres sub-procesos principales: en primer lugar se realiza la preparación y

extendido de la muestra sobre un porta-objetos (frotis), seguidamente se realiza la tinción de bacilos ácido-alcohol resistente (BAAR) el cual es realizado por el método ZN a base de fucsina fenificada, azul metileno, alcohol ácido y agua. Finalmente, se realiza la observación microscópica. En el presente capitulo se analiza la automatización de la tinción.

De acuerdo al estándar según el Organismo Andino de Salud (ORGANISMO ANDINO DE SALUD – CONVENIO HIPÓLITO UNANUE, 2018), se presentan los subprocesos y los tiempos de reposo de los reactivos en el proceso de baciloscopia basada en tinción Zhien Nielsen. Es este caso se está considerando un tiempo de calentamiento de 5 minutos y 75 segundos (con control de la temperatura), pues la norma recomienda como mínimo 5 minutos. Adicionalmente, es importante recalcar que en estos 11 minutos se realizan 4 teñidos, pues el prototipo es capaz de teñir cuatro muestras en paralelo. En la Tabla 3.1 se presentan los tiempos considerados para el diseño del módulo de tinción.

Tabla 3.1: Tiempo estimado para un proceso automatizado de tinción. [10]

	Sub-Proceso	Tiempo	Tiempo en segundo
Tinción	Aplicación de fucsina	30s	30
	Calentamiento	5min 75seg	375
	Enjuague 1	30s	30
	Aplicación del decolorante	2min	120
	Enjuague 2	30s	30
	Aplicación de azul	1min	60
	Enjuague 3	30s	30
	Total	11.15 min	675

3.2 Problemáticas de la maqueta funcional

[10] Fuente: Laboratorio de micobacterias-MINSA

El módulo de tinción puede ser subdivido en 5 sistemas de acuerdo a su funcionalidad: sistema de calentamiento, alimentación de energía, dispensado de reactivos, desplazamiento de muestras e interfaz de usuario. A continuación, se presentan los problemas según el proceso del módulo de tinción.

3.2.1 Calentamiento de muestras

Respecto al sistema de calentamiento, este componente crítico del módulo de tinción requiere una validación adecuada y detalla por lo que su análisis se abordará en los capítulos siguientes. Si bien funcionalmente el control de temperatura proporcional es el adecuado, es necesario un análisis más profundo que tome en cuenta la distribución del calor en todo el equipo y en las mismas muestras de tal forma que el aumento de temperatura no afecte el funcionamiento de otros componentes como las tarjetas**,** circuitos electrónicos, dispensadores de reactivos entre otros. Por ahora, cabe mencionar que el dispositivo sensor de temperatura es el LM35 y los elementos calefactores son dos lámparas de luces halógenas de la marca OPALUX de 220V - 350W. Los elementos que concentran el calor y lo direccionan hacia las muestras son dos piezas de material cerámico.

3.2.2 Alimentación

Se encontró que en el diseño de la maqueta se considera una fuente conmutada que recibe una señal de 220 VAC y entrega una señal de 12 VDC – 2 A, y una fuente regulada que recibe el voltaje de la fuente mencionada y entrega una señal de 5 VDC a 0.7 A. Teniendo en cuenta la optimización del espacio y consumo de energía del equipo portátil, tener dos fuentes separadas es un problema, será preferible realizar las mediciones de corrientes con los equipos en sus condiciones más extremas de funcionamiento y dimensionar una sola fuente con salidas de 12V/2A y 5V/0.7A.

3.2.3 Dispensado de reactivos y agua

Para el dispensado de reactivos se consideraron Bombas - RS 360 RH. Estas tuvieron un deterioro bastante rápido cuando se usaron para suministrar los reactivos, incluso al mes de uso quedaron inútiles. Este desgaste se produce a causa de que el alcohol desgastaba el material de la bomba. Adicionalmente, el motor de estas bombas requiere una alta cantidad de corriente (entre 1.76A a 8.6A, ver figura 3.1). Finalmente, el fabricante de las bombas recomienda alimentarla con tensiones entre 6 a 9V. En el diseño de la maqueta no se consideró una fuente de alimentación de 5V, la bomba se estuvo alimentado con 12V. En consecuencia, será necesario dimensionar una bomba adecuada.

Figura 3.1: Características de la Bomba - RS 360 RH [11]

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN-m	g-cm	W	mN-m	g-cm	A
RS-360SH-2885	3~9	7.2V CONSTANT	12500	0.36	10380	1.76	7.00	71.3	7.59	41.2	420	8.60
RS-360SH-10500	12~25	12V CONSTANT	3500	0.050	2590	0.14	2.74	28.0	0.74	10.6	108	0.41

Otro problema encontrado son los actuadores de las bombas. Debido a que las bombas no necesitaran trabajar en ambos sentidos, el uso del módulo driver L298 es innecesario; además de ocupar mayor espacio, este módulo posee una caída de tensión de alrededor 1.2V, lo cual dejaría a las bombas funcionando por debajo de su voltaje recomendado. Será necesario rediseñar e implementará un driver que sea parte de una sola tarjeta electrónica y tenga menores caídas de tensión.

3.2.4 Desplazamiento de muestras

El desplazamiento de la muestras se realiza con un movimiento suave de tal forma que se mantiene la tensión superficial cuando se encuentran los reactivos sobre las muestras. Sin embargo, se ha identificado que en el

[11] Fuente: Pagina web del distribuidor del fabricante (Mabuchi motors,2011)

enjuague el dispensado no es capaz de romper la tensión superficial del agua sobre las muestras. Es preciso no tener residuos de agua pues diluyen el reactivo dispensado luego del enjuague. En consecuencia, es necesario encontrar un mecanismo que permite la eliminación del agua remanente posterior al enjuague.

3.2.5 Interfaz de usuario

Actualmente la maqueta funcional cuenta con un botón de inicio de proceso. El módulo deberá tener un indicador que permita visualizar si el equipo se encuentra en funcionamiento.

3.3 Problemas identificados en prototipos anteriores

A continuación**,** se presentan las dificultades que durante el desarrollo de los anteriores prototipos se han presentado y están pendientes por validar. En la tabla 3.2 se presentan estas dificultades con el fin de poder validarlas.

Tabla 3.2: Problemas de interés encontrados en versiones anteriores del SIBA

Posibles problemas	Posibles procesos y/o componentes involucrados
No se puede identificar la cantidad de reactivos disponibles.	- Recipientes de reactivos. - Visualizadores de reactivos.
El tiempo de tinción de muestras por unidad de tiempo puede ser optimizado.	Dispensado de reactivos y agua: - Retardo de transporte del fluido. - Mayor volumen de agua en el dispensado. Calentamiento de muestras: - Tiempos altos de calentamiento.
El recubrimientos de las muestras no es uniforme (no se cubren todas las muestras al 100%).	Dispensado de reactivos.
No se tiene una medición real de la temperatura en la muestra.	Calentamiento de muestras: - Ubicación del sensor - Ajuste de los parámetros del algoritmo de control.
La velocidad de enfriamiento del la campana que contiene la lámpara de calefacción afecta la siguiente serie de tinción.	Calentamiento de muestras: - Velocidad de enfriamiento del foco de luz halógena. - Velocidad de enfriamiento de la pieza cerámica.

3.4 Módulo de microscopía.

Respecto al módulo de microscopía dentro del grupo de trabajo existen miembros que se encargaron del diseño del sistema mecánico así como también expertos en imágenes que han realizado el diseño del software de

conteo de bacilos. Por lo que el desarrollo de la presente tesis involucra la integración del sistema mecánico con la electrónica que controla dicho sistema y que a la vez interactúa con el software de conteo. Para ello hubo la necesidad de realizar algunos ajustes en la mecánica del microscopio como también en el software diseñado. Es importante resaltar que el software diseñado en MATLAB será implementado en el lenguaje de programación Python. Esta implementación es parte también de la presente tesis. Finalmente, dado que no existe un software de autoenfoque diseñado para el sistema de microscopía, este software ha sido diseñado he implementado como parte de la presente tesis.

CAPÍTULO 4:
REDISEÑO DEL MÓDULO AUTOMÁTICO DE TINCIÓN DE CUATRO MUESTRAS DE ESPUTO

En el presente capítulo se muestra el alcance del módulo de tinción; posteriormente, se presenta su diagrama de bloques para luego exponer el re-diseño del hardware y software electrónico. Finalmente, se expone el estudio del sistema mecánico del prototipo. Para ello se establecen los requerimientos, alternativas, selección y el diseño propuesto para los elementos analizados.

4.1 Alcances

El rediseño mecánico contempla el análisis del subsistema de dispensado de reactivos y del subsistema de calentamiento del módulo de tinción. Se analiza el dispensado de reactivos poniéndose énfasis en la distribución óptima y uniforme en la muestra. El análisis del sistema de calentamiento abarca la validación del control de temperatura en la muestra. A partir de

este resultado se plantea y realiza el rediseño del software y hardware, proponiendo sugerencias para futuros trabajos.

4.2 Diagrama de bloques

La figura 4.1 muestra el diagrama de bloques que esquematiza la función de los componentes durante el proceso. El módulo está conformado por una plataforma móvil, un subsistema de calentamiento, un subsistema de alimentación y la interfaz de usuario. La plataforma se desplaza al punto de inserción de muestras de tal forma que el usuario coloca los 4 portaobjetos. Posteriormente, la plataforma se desplaza a la posición de inyección de fucsina, seguidamente las muestras son trasladas hacia la posición de calentamiento de las muestras. Una vez calentadas las muestras la plataforma móvil se traslada a la posición de enjuague. Posterior al primer enjuague se aplica el alcohol ácido y se realiza un segundo enjuague. Finalmente, se aplica el azul de metileno y seguidamente se realiza un último enjuague.

Los elementos que intervienen en el proceso pueden ser agrupados de la siguiente forma:

- Planta: conformada por la estructura mecánica del subsistema de tinción
- Hardware conformado por todos los componentes electrónicos; y
- Software que se programa en el microcontrolador.

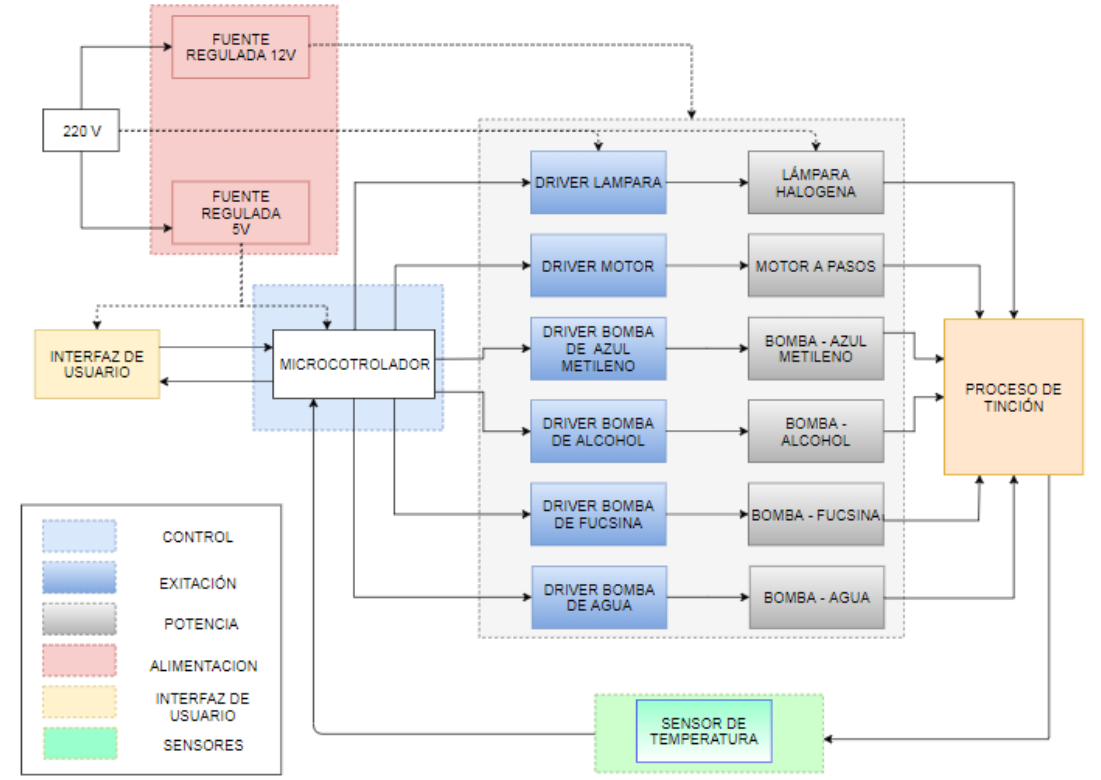


Figura 4.1: Diagrama de bloques del Subsistema de tinción

4.3 Estructura mecánica del módulo de tinción

El sistema mecánico debe permitir automatizar el proceso. El módulo de tinción consta de: una plataforma móvil, el área de inserción de muestras, el de tinción de muestras, el de calentamiento de muestras y el de alojamiento de los componentes electrónicos. El área de inserción de muestras debe contar con espacio libre que permita facilitar la ubicación de muestras. El área de tinción de muestras debe considerar la estructura requerida para alojar las mangueras y boquillas por donde fluyen los reactivos (fucsina, alcohol ácido, azul metileno y agua). Seguidamente, el área de calentamiento de muestras debe considerar las piezas cerámicas que permitan el calentamiento de las muestras y una estructura para la ubicación del sensor de temperatura. La plataforma móvil permite la ubicación de las muestras en las distintas áreas. En la figura 4.2 se identifican las áreas del módulo de tinción.

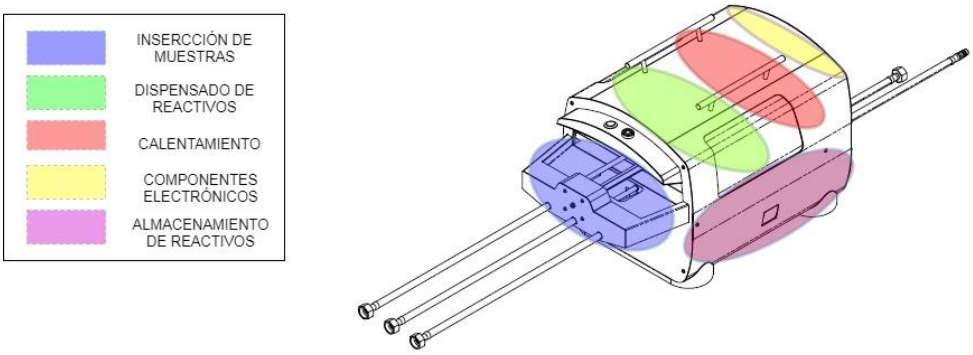


Figura 4.2: Disposición de las áreas del proceso en la estructura mecánica

Adicionalmente, se debe considerar el ingreso de los reactivos y la salida de los residuos. El llenado a través de estas tuberías deber ser de forma rápida y accesible. En la figura 4.3 se muestra el prototipo resaltando las tuberías de ingreso y salida de líquidos.

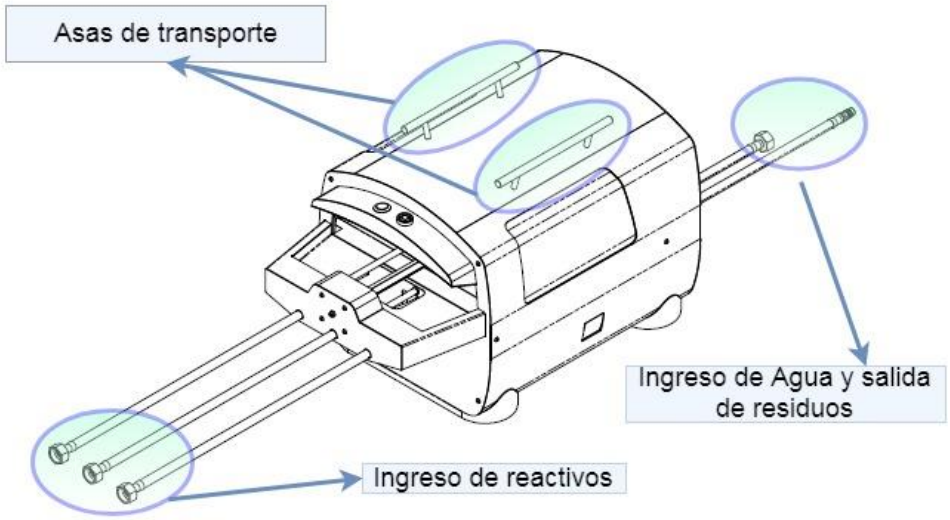


Figura 4.3: Interfaz mecánica con el usuario

4.3.1 Área de inserción de muestras

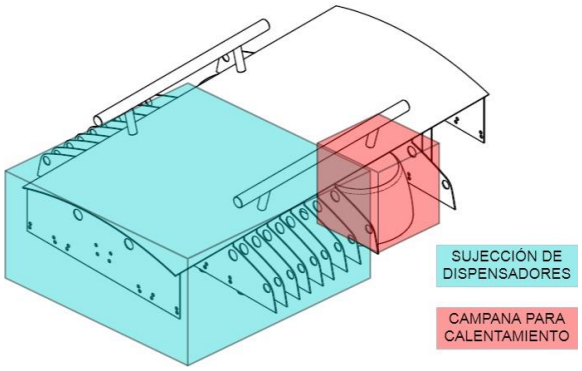
El área de inserción de muestras está disponible para que el operario coloque las muestras y una vez realizado todo el proceso se puedan extraer; esta se encuentra en la parte delantera del equipo.

4.3.2 Calentamiento de muestras

El calentamiento de muestras se realiza a través de dos lámparas de luz halógena, siendo dos piezas cerámicas las encargadas de direccionar el calor. Una de la piezas cerámicas (campana) se puede apreciar en la figura 4.4 A.

4.3.3 Dispensado de líquidos y expulsión de residuos

El dispensado de líquidos se realiza por caída de los fluidos a través de 4 boquillas dispensadoras ubicadas en la parte superior de la estructura tal y como se muestra en la figura 4.4 A. Los depósitos de los reactivos se encuentran debajo de la estructura de desagüe que se muestra en la figura 4.4 B. Se plantea un movimiento suave de la plataforma, mientras se realiza el dispensado de reactivos. Este movimiento será de tal forma que se cubra totalmente la muestra con el reactivo dispensado.



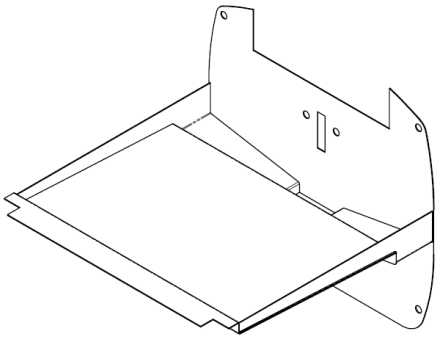


Figura 4.4 A soporte superior **Figura 4.4 B Desague**

Figura 4.4: Dispensado y desague de reactivos

4.3.4 Plataforma móvil

La plataforma móvil es la encargada de sujetar las muestras que se desplazan a través de un eje. El medio de sujeción son pinzas metálicas. La plataforma es transportada por una faja acoplada a la base, dos varillas y cuatro cojinetes soportan este movimiento. El movimiento de la plataforma luego del enjuague será brusco de tal forma que se rompa la tensión

superficial del agua sobre las muestras. La figura 4.5 muestra la estructura de la plataforma.

La polea que genera el desplazamiento lineal posee 16 dientes separados 2mm entre dientes. Una vuelta completa del eje del motor, es decir un giro de 360°, equivaldría a 32mm. El motor es un Nema 17 de 200 pasos por revolución, es decir 1.8° por paso; y como se está trabajando a 1/8 de paso esto genera un movimiento lineal de 0.02mm.

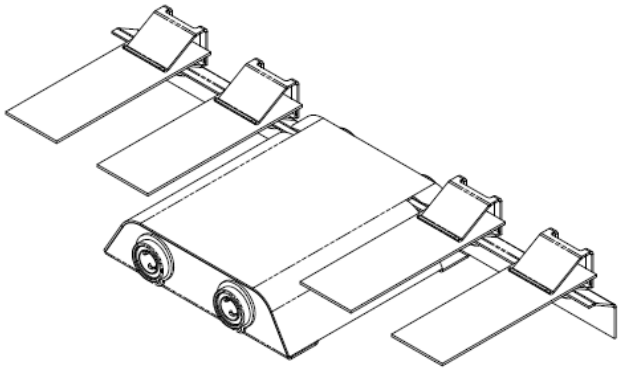


Figura 4.5: Estructura de la plataforma movil

4.3.5 Área de componentes electrónicos

El área de componentes electrónicos deberá de tener un fácil acceso para optimizar la ejecución de reparación y mantenimiento. En la figura 4.6 se muestra la estructura posterior que sujeta los componentes electrónicos.

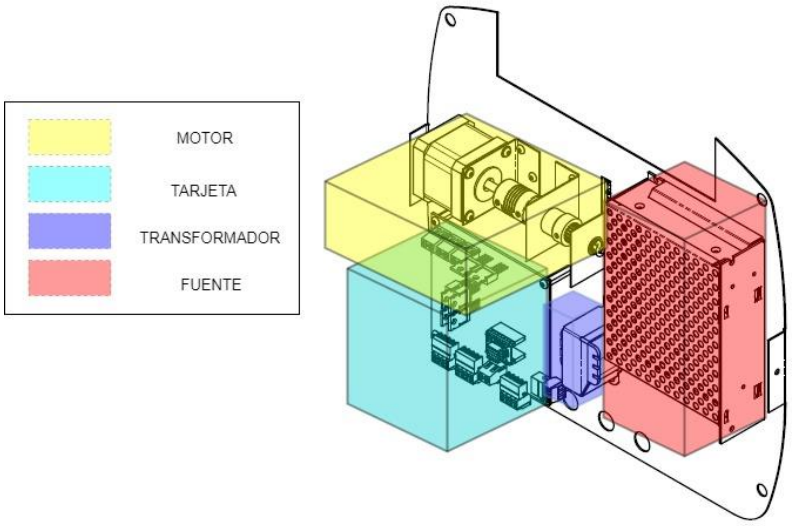


Figura 4.6: Soporte de componentes electrónicos

4.4 Hardware Electrónico del módulo de tinción

El subsistema electrónico se ha implementado de acuerdo al diagrama de bloques.

4.4.1 Sensor de posición

Se requiere de un dispositivo que indique la posición de referencia al inicio del proceso. Para ello el sistema cuenta con un sensor de posición del tipo interruptor fin de carrera como lo muestra la figura 4.7. La tabla 4.1 muestra los requerimientos y las características de este sensor.

Figura 4.7: Interruptor de fin de carrera

Tabla 4.1: Características del sensor de posición fin de carrera.

Características	Valores del fin de carrera
Precisión menor o igual a 0.1 cm.	[+/- 0.05 cm]
Bajo tiempo de respuesta.	[125 µs]

Este sensor presenta dos estados On-Off, se mantiene abierto cuando la plataforma móvil que transporta los portaobjetos no está en contacto con este y se cierra cuando la estructura lo mantenga presionado. La figura 4.8 muestra el diagrama esquemático del circuito (resistencia de pull-up) de sensado de posición.

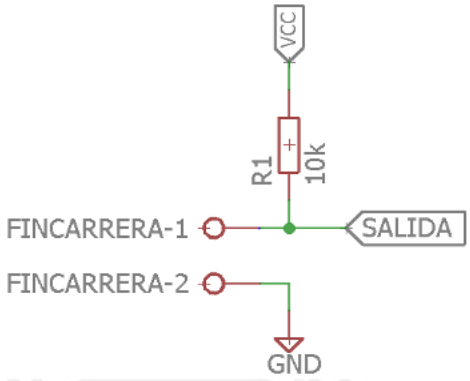


Figura 4.8: Diagrama esquemático del circuito de sensado de posición

4.4.2 Sensor de temperatura

Se requiere medir la temperatura en la muestra. Es de notar que al tener una plataforma móvil y de muestras extraíbles, el sensor ideal tendría que ser un sensor de temperatura de no contacto. Se probó el sensor infrarrojo mlx90614 con resultados desalentadores, pues el equipo posee una fuente de calentamiento, cuya luz irradiada afecta irremediablemente la medición de este sensor. En consecuencia se opta por continuar midiendo la temperatura a través de un sensor LM35, este sensor posee un voltaje de salida de 10mV / °C. El rango de monitoreo de temperatura es de 0 °C a 100 °C, lo cual define una tensión de salida en el rango de 0 mV a 1000 mV. La tabla 4.2 muestra las principales características del sensor. No se ha considerado un amplificador debido a que es permisible una exactitud de 1°C, entonces teniendo en cuenta que el sensor tiene una exactitud de +/- 0.5°C; es decir 5mV y considerando un microcontrolador con una resolución de 4.9mV por unidad se puede leer sin problemas todo valor del sensor con la exactitud requerida.

Tabla 4.2: Sensor LM35

Sensor	LM35
Característica	Característica
Rango de temperatura	[-55 °C,150 °C]
Exactitud	[+/- 0.5 °C]
Consumo de corriente	[<60 uA]

Se realiza una validación del control de temperatura con un set point de 90°, ubicando la termocupla y el sensor LM35 sobre el portaobjetos. Los resultados se muestran en la Figura 4.9.

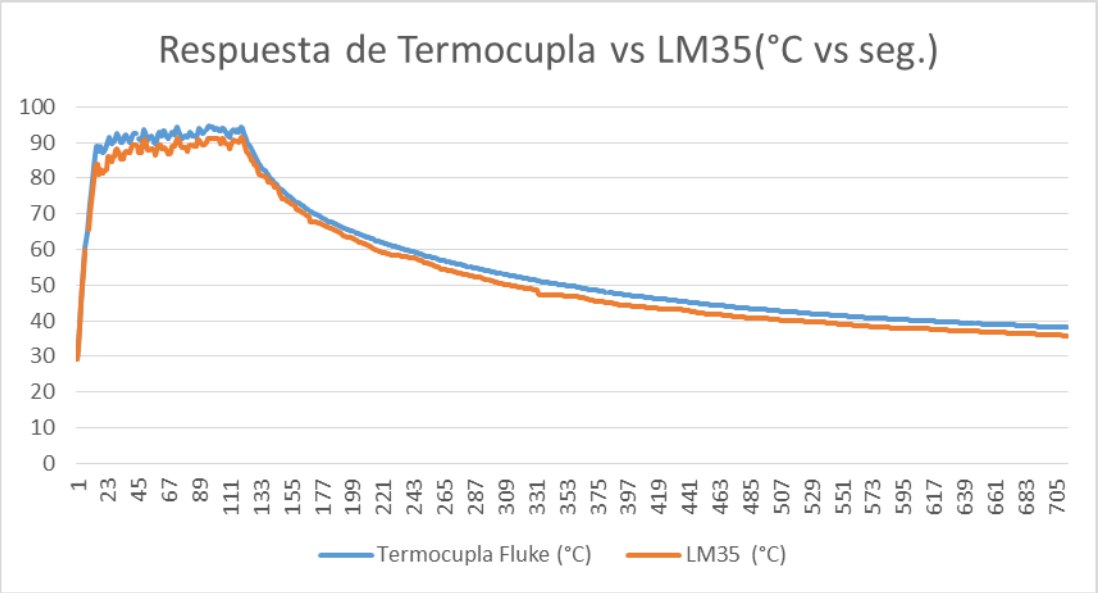


Figura 4.9 Respuesta de la temperatura medida en el sensor por la termocupla FLUKE 179 vs el sensor LM35.

Las respuestas difieren levemente por lo que la problemática está en la ubicación del sensor de referencia.

Se propone colocar el sensor debajo del portaobjetos y debajo de un pedazo de vidrio como protección. En un principio se propuso colocar pasta térmica entre el vidrio de protección y el sensor. Sin embargo, debido a que el vidrio

posee un calor específico mayor al de la fucsina, la temperatura del vidrio se incrementa rápidamente sobrepasando la temperatura de la muestra. En la figura 4.10 se muestra la distancia entre el portaobjetos del sensor de referencia y los portaobjetos del análisis.

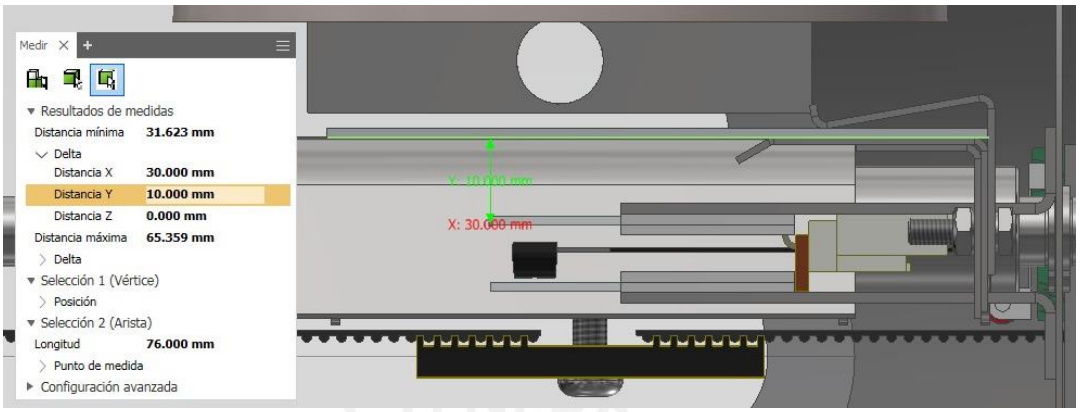


Figura 4.10: Posición del sensor de temperatura

Se evaluará si la posición propuesta del sensor representa la temperatura de la muestra, lo ideal será que el comportamiento sea similar al de la figura 4.9, donde cabe esperar una diferencia constante entre ambas.

4.4.3 Bombas y excitador de las bombas:

I. Bombas

En la tabla 4.3 se muestra el resultado de la evaluación de las bombas que se utilizaron para remplazar a las bombas RS-360SH utilizadas en los equipos PAME, estas bombas poseen un consumo de corriente elevada. Alimentadas a su voltaje nominal de 7.2V pueden llegar a tener un consumo de 8.6A a plena carga.

Las pruebas para la selección de las bombas de 12V inicialmente se realizaron con agua. Se considera un flujo aceptable si la bomba es capaz de dispensar agua por las cuatro boquillas de salida. Entonces, únicamente las bombas cuyo dispensado con agua eran aceptables para el sistema fueron probadas con fucsina. La fucsina tiene mucha mayor densidad que el agua, entonces si la bomba no es capaz de bombear agua no será capaz de

bombear fucsina o azul metileno. Esta prueba se realizó en la maqueta funcional.

Tabla 4.3: Selección de bombas

	Flujo de agua	Flujo de fucsina	Consumo de corriente con carga (agua)	Consumo de corriente con carga (fucsina)
Bomba peristáltica 5000RPM , 12V	Insuficiente (goteo)	-	-	-
Bomba sumergible DC-808	Aceptable(La salida es por los 4 orificios)	Aceptable	0.5 A	0.7 A
Bomba para limpiaparabrisas de autos LCTY	Insuficiente(solo se dispensa por 4 salidas)	-	-	-
Bomba RS-385-PUMP100.	Aceptable (La salida es por los 4 orificios)	Aceptable	0.4 A	0.5 A

En el caso de la bomba sumergible se tuvo que sellar la entrada de líquido, dado que la absorción seria a través de una tubería y al ser una bomba sumergible permitía el acceso no solo por el orificio de entrada. El sellado se realizó con resina epóxica.

Una vez migradas las bombas de la maqueta al prototipo final, a causa del cambio de las tuberías (tamaño y diámetros), la bomba DC-808 al ser una bomba de flujo y no de presión no dispensaba por los 4 orificios y se tuvo que descartar. Finalmente, se realizaron pruebas de dispensado y flujo con la Bomba RS-385 en el prototipo final. En el capítulo 6 se presentan los resultados obtenidos.

II. Excitador de la bomba

La tensión de alimentación de las bombas es de 12V y su corriente nominal de 300 mA, el excitador considerado es un MOSFET IRF530 que soporta una tensión V_{DS} de 100V y una corriente de hasta 3A (V_{GS}=5V), ver gráfica 4.8. Según esta gráfica, se puede ver que para corrientes cercanas a 0.5A y una tensión V_{GS} de 4.75V (se realizó la medición de la salida digital de un

arduino) se tendrá una I_D aproximada 0.15A, estando el MOSFET en región óhmica. La figura 4.11 exhibe las curvas de respuesta del MOSFET.

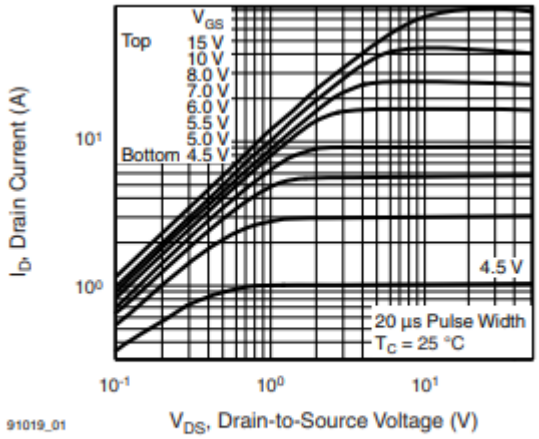


Fig. 1 - Typical Output Characteristics, T_C = 25 °C

Figura 4.11: Respuesta del MOSFET IRF530[12]

Así mismo, se utiliza los diodos 1N4004 en paralelo a las cargas a fin de proteger el MOSFET ante posibles transitorios. Se coloca una resistencia pull-down R1 = 100 KΩ entre los terminales G-S del MOSFET para evitar el ruido eléctrico, es decir minimizar los efectos de las capacitancias parasitas. El circuito para cada bomba se muestra en la figura 4.12.

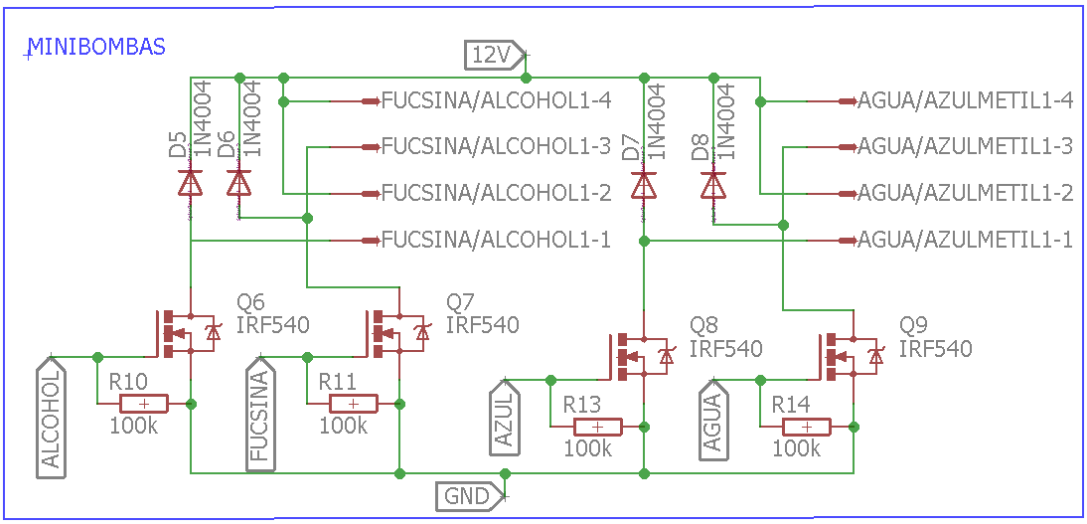


Figura 4.12: Excitadores de las bombas

[12] Fuente: Hoja de datos del MOSFET IRF530.

4.4.4 Actuador de calefacción y su excitador

Cada una de las campanas contiene una lámpara de luz halógena que irradia calor a dos muestras. En la figura 4.13 se muestra la lámpara de luz halógena utilizada (OPALUX 220V-350 W). En la figuras 4.14, 4.15 y 4.16 se pueden apreciar con mayor detalle la disposición y distancia entre la campana, la lámpara halógena y las muestras. Se aprecia que el recipiente tiene forma de campana pues se requiere direccionar el calor a las muestras.

Figura 4.13: Foco de luz halógena

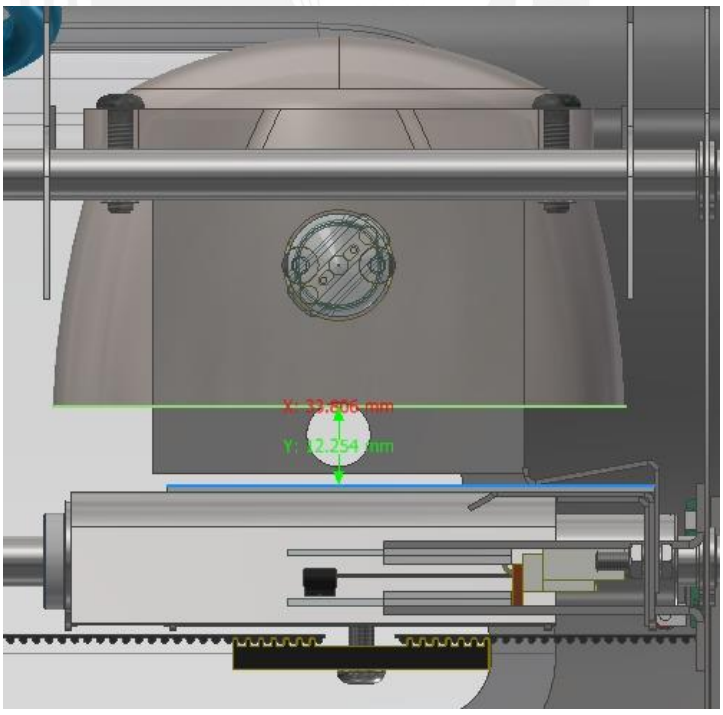


Figura 4.14: Distancia entre la campana y las muestras (Y= 12.3mm)[13]

[13] Imagen proporcionada por Andrés Garcés Beltrán.

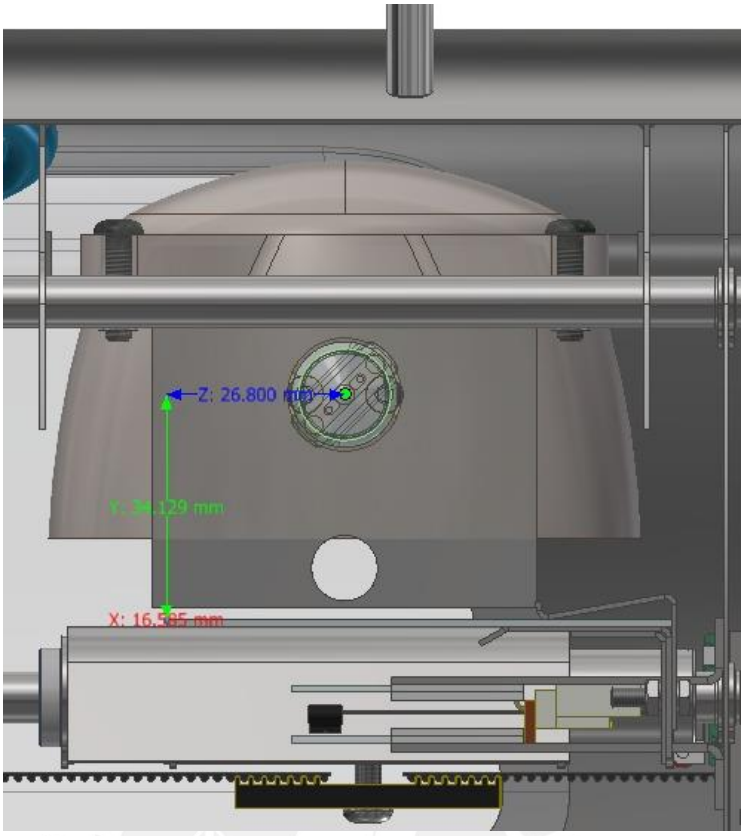


Figura 4.15: Distancia entre centro del foco de luz halógena y la muestra (Y=34.1mm)[14]

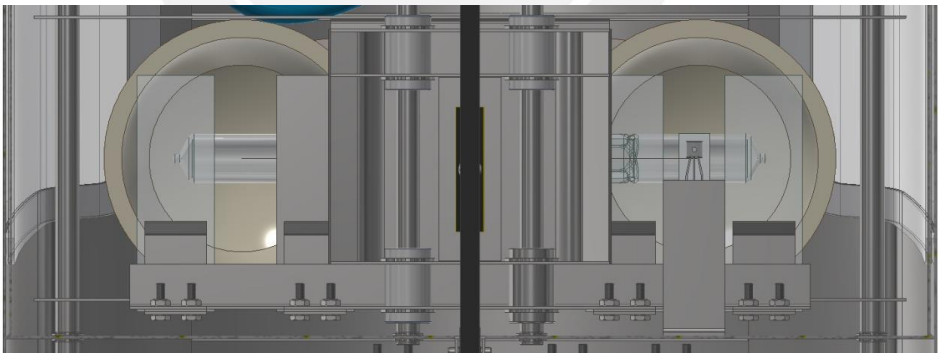


Figura 4.16: Vista inferior de la distribución de la campana y muestras.[15]

El excitador contempla la variación del ángulo de disparo de la señal senoidal que acciona las lámparas. Para poder realizar la variación del

[14] Nota 15.
[15] Imágenes proporcionada por Andrés Garcés Beltrán.

ángulo de disparo se necesita identificar una referencia que en este caso la da un circuito de cruce por cero de la señal senoidal del tranformador de 220V/12V.

A continuación se muestra el circuito de cruce por cero en la figura 4.17 y el del control del ángulo de disparo en la figura 4.18. Se utiliza un TRIAC BT136 que soporta una corriente de hasta 4 A.

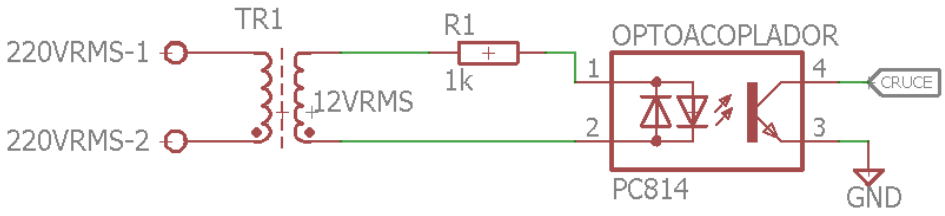


Figura 4.17: Diagrama esquemático del detector de cruce por cero.

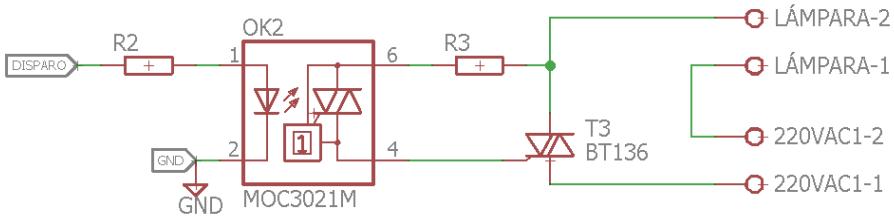


Figura 4.18: Diagrama esquemático del circuito de control de ángulo de disparo.

4.4.5 Motor a pasos y su excitador:

Para el desplazamiento de las muestras se utiliza un motor a pasos bipolar XY42STH34-0354A donde el movimiento del motor se puede realizar a través de micropasos. El driver utilizado es el A4988 que permite el control del sentido de giro y velocidad del motor por medio de los pines de control de micropasos, número de pasos y dirección (MS1, MS2, MS3, STEP y DIR respectivamente). La tabla lógica es mostrada en la tabla 4.4.

Tabla 4.4: Selección de micropasos (8A) y Control de giro y parada del motor a pasos(8B).

MS 1	MS2	MS3	Pasos
0	0	0	Paso completo
1	0	0	1/2 paso
0	1	0	1/4 de paso
1	1	0	1/8 de paso
1	1	1	1/16 de paso

STEP	DIR	GIRO
Pulso	0	Horario
Pulso	1	Antihorario
-	x	Parado

La figura 4.19 muestra el diagrama esquemático del excitador de los motores a paso para los ejes X, Z e Y respectivamente. Los motores a pasos se energizan con 12V. El driver A4988 posee dos entrada de alimentación, una de 5V para el control y otra de 12V para la potencia.

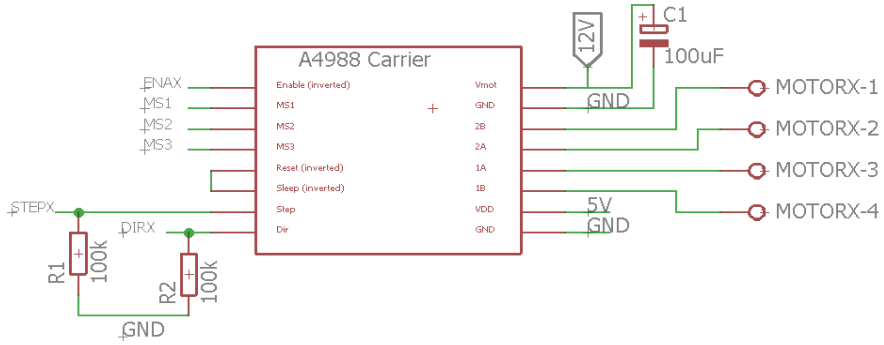


Figura 4.19: Esquemático del excitador del motor de paso

Para el caso del motor paso a paso se realizó las pruebas de funcionamiento mecánico y electrónico dando en la validación resultados exitosos. Por lo que no será necesario realizar ninguna modificación en el hardware electrónico. Sin embargo, ajustes en el micro-paso a fin de optimizar la velocidad serán necesarios. Debido a que se tiene acceso desde el microcontrolador a los pines de control de los micropasos, solo será necesario modificarlo vía software.

4.4.6 Fuente de alimentación

El sistema cuenta con la fuente de alimentación conmutada RD65 de 5V/ 6 A y 12V / 3A para los dispositivos de potencia. La fuente es la que se muestra en la figura 4.20.

Figura 4.20: Fuente conmutada del sistema.

En la tabla 4.5 se muestra el consumo de corriente máxima para las dos fuentes de alimentación. Se puede verificar que la fuente de alimentación cubre los requerimientos.

Tabla 4.5: Consumo de corriente

Tensión de alimentación	Corriente Máxima (mA)
5 V	15.8
12V	175(Equipo estático) 191(Motor en movimiento) 390(Dispensado de fucsina) 590(Dispensado de agua) Sacudido de muestras(260)

4.4.7 Esquema de conexiones.

Con el fin de poder visualizar la interacción y establecer la ubicación de los componentes electrónicos, eléctricos y mecánicos se presenta en la figura 4.21 el esquema de conexiones.

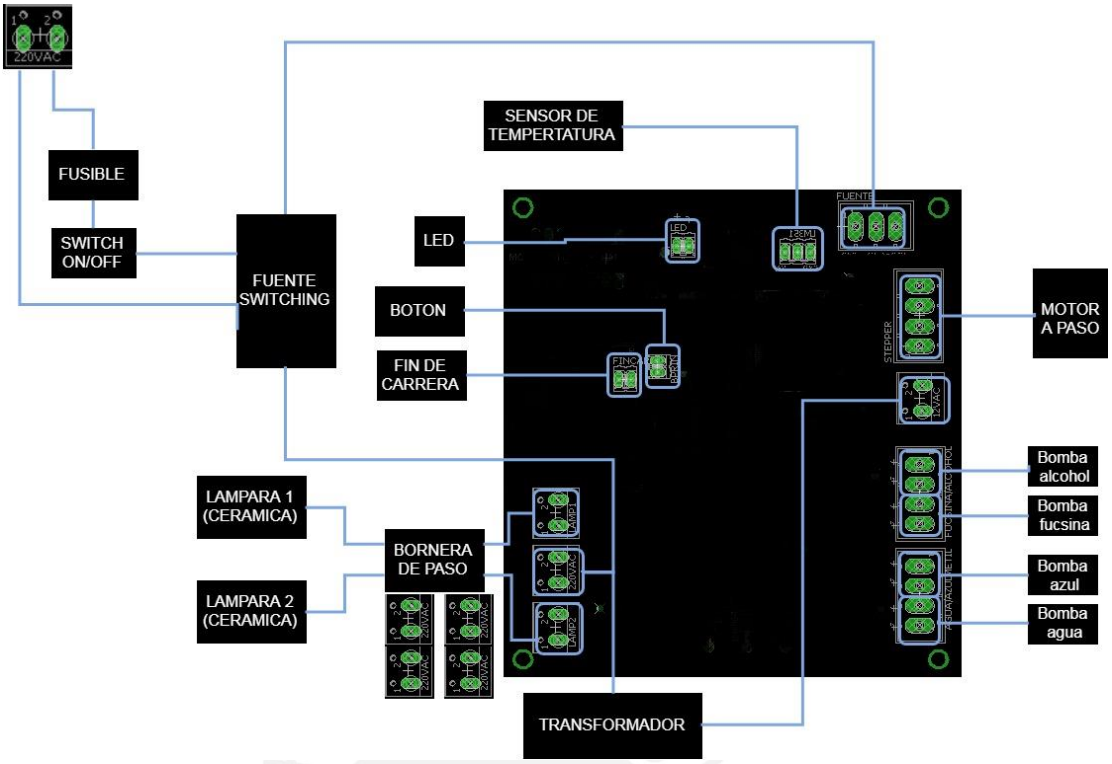


Figura 4.21: Esquema de conexiones

4.5 Software de la plataforma Electrónica Arduino para el módulo de tinción.

A continuación en la figura 4.22 se muestra el diagrama de flujo de todo el proceso de tinción. Seguidamente se presentan cada uno de los bloques que conforman el proceso general. En el Anexo C se encuentran todos los programas referentes a la programación en el microcontrolador Atmega328; es decir, la implementación de los diagramas de flujo.

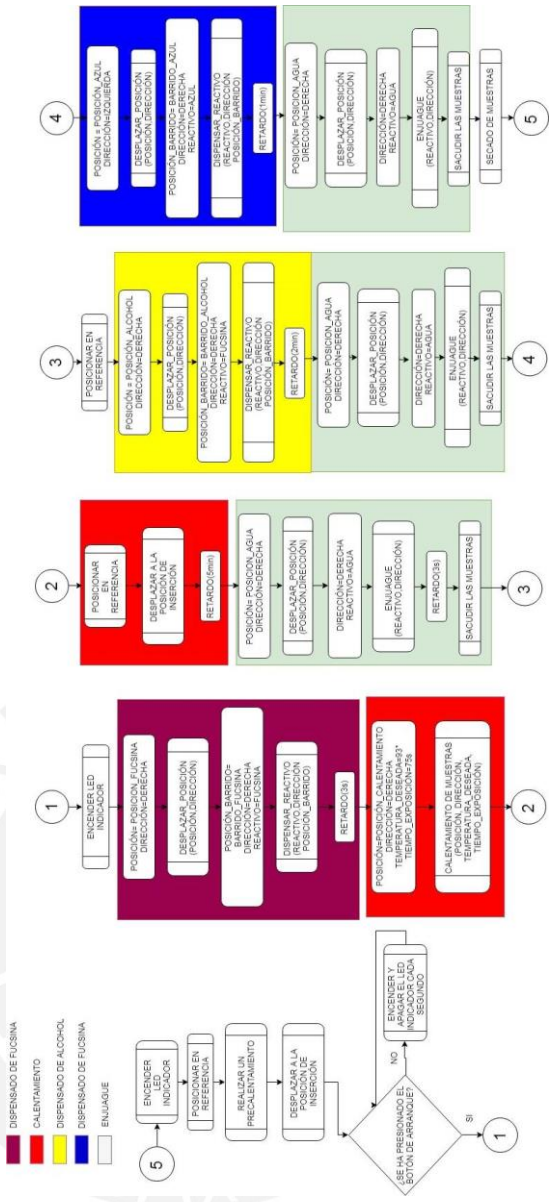


Figura 4.22: Diagrama de flujo general del proceso de tinción

Como se puede apreciar en la figura 4.22, básicamente el diagrama de flujo describe el proceso de tinción: inserción de muestras, tinción con fucsina, calentamiento de muestras, enjuague, dispensado de alcohol acido, enjuague, tinción con azul metileno y el enjuague final. A continuación se describen brevemente las principales funciones del software.

4.5.1 Subrutina de localización de referencia

Una vez encendido el módulo, con el fin de localizar una referencia la plataforma se desplaza hacia la posición del fin de carrera, una vez que se

detecta que la plataforma ha activado el sensor se define esta posición como la posición cero del desplazamiento de la plataforma. La figura 4.23 muestra el diagrama de flujo correspondiente.

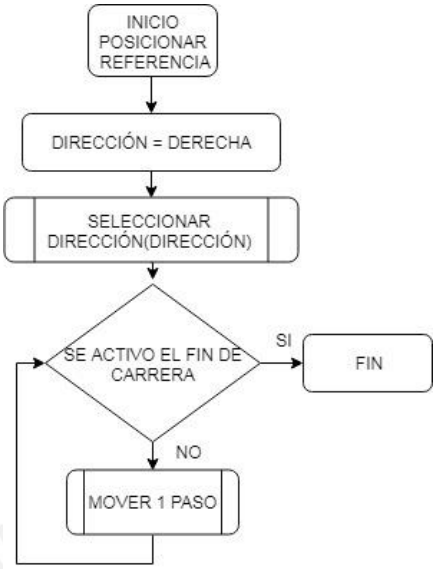


Figura 4.23: Subrutina de localización de referencia

4.5.2 Subrutina de inserción de muestras

Esta subrutina permite ubicar la plataforma en el área de inserción. En esta posición se espera hasta que el operario presione y suelte el pulsador de arranque para iniciar con el proceso de tinción. La figura 4.24 muestra el diagrama de flujo de la subrutina inserción de muestras.

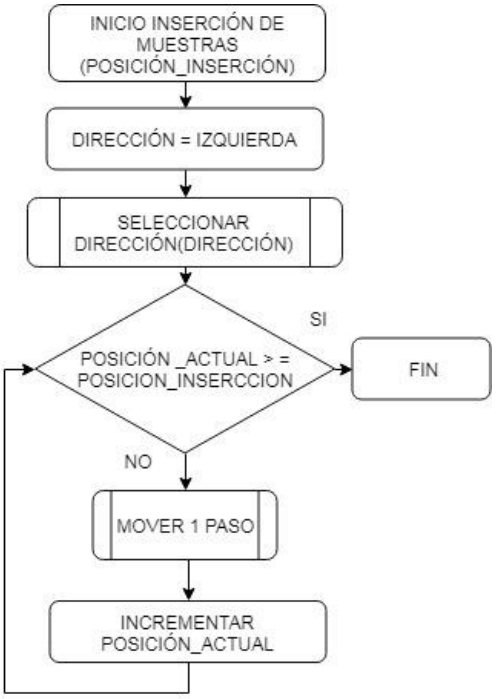


Figura 4.24: Subrutina de inserción de muestras

4.5.3 Subrutina de posicionamiento

Se realiza el posicionamiento de la plataforma para el dispensado de reactivos, agua y el calentamiento. La figura 4.25 muestra el diagrama de flujo de la subrutina de posicionamiento.

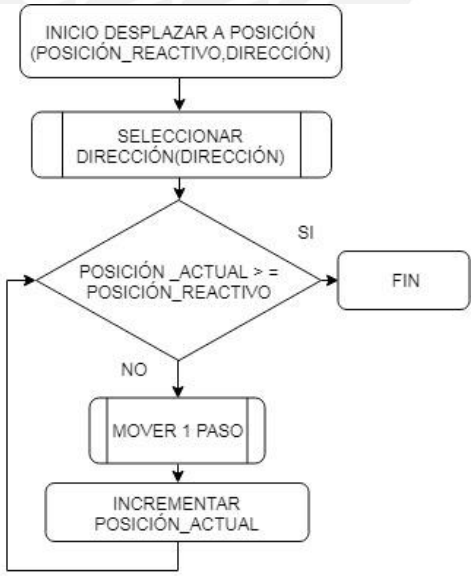


Figura 4.25: Diagrama de flujo de la rutina de posicionar plataforma

4.5.4 Subrutina de dispensado de reactivos

Para realizar el dispensado de los reactivos se utiliza la subrutina de dispensado de reactivo (DISPENSAR_REACTIVO); en primer lugar, se define la posición deseada, el sentido del movimiento y el reactivo a dispensar. Seguidamente, se activa la bomba del reactivo para luego invocar a la subrutina de desplazar a posición deseada (DESPLAZAR_POSICIÓN) para realizar el barrido y finalmente desactivar la bomba. El barrido se utiliza a fin de dispensar el reactivo sobre las muestras mientras la plataforma se mueve suavemente de tal forma que se recubre la muestra manteniendo la tensión superficial. En el caso del enjuague se activa la bomba de agua por 25s tras los cual se realizan tres barridos en ambos sentidos. Este doble barrido se ejecuta mediante movimientos bruscos de ida y vuelta a fin eliminar los residuos de agua. A continuación, la figura 4.26 se muestran los diagramas de flujo de las subrutinas de dispensado, enjuague y sacudido de las muestras.

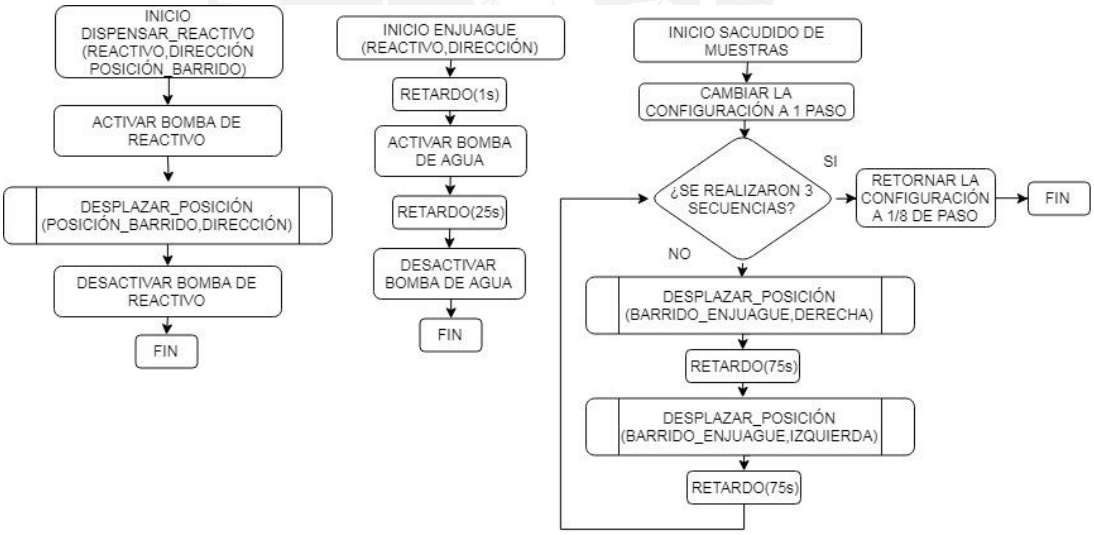


Figura 4.26: Dispensado de reactivos

4.5.5 Subrutina de calentamiento

4.5.5.1 Descripción del control de temperatura de la maqueta funcional

La maqueta funcional cuenta con el algoritmo de control para calentar las muestras descrito en (Medina, B. ,2018). En este algoritmo: en primer lugar, se lee la temperatura actual por medio del sensor de temperatura y se

calcula el error mediante la diferencia entre la temperatura deseada y la temperatura actual medida; seguidamente, si el error es mayor a 2 °C se establece un ángulo de disparo mínimo definido en el programa principal; caso contrario, se realiza un algoritmo de control proporcional ya que el ángulo de disparo (señal de control) será proporcional al error. Para este control se había considerado que la temperatura de la muestra tiene una relación lineal respecto a la temperatura del sensor de referencia. Sin embargo, como se verá en las siguientes líneas la respuesta de la planta no es lineal.

4.5.5.2 Validación y rediseño del control de temperatura

El sensor de temperatura migra de posición de la maqueta funcional al prototipo final. El sensor de temperatura se ha colocado para medir la temperatura del aire debajo de la campana cerámica y por encima de las muestras. Se han empleado cuatro multímetros FLUKE 179 con termocuplas sobre los portaobjetos sin fucsina a fin de medir la temperatura a la cual están expuestas las muestras. Para realizar la medición de temperatura de forma sincronizada se ha grabado un video que posteriormente pasa por un software de procesamiento a fin de extraer los valores del visualizador del multímetro. En el anexo B se encuentra el software de procesamiento modificado a partir del código de (Rosebrock, A. , 2017). Una de las pruebas de medición de temperatura y control de temperatura con set point de 60°C se muestra en la figura 4.27.

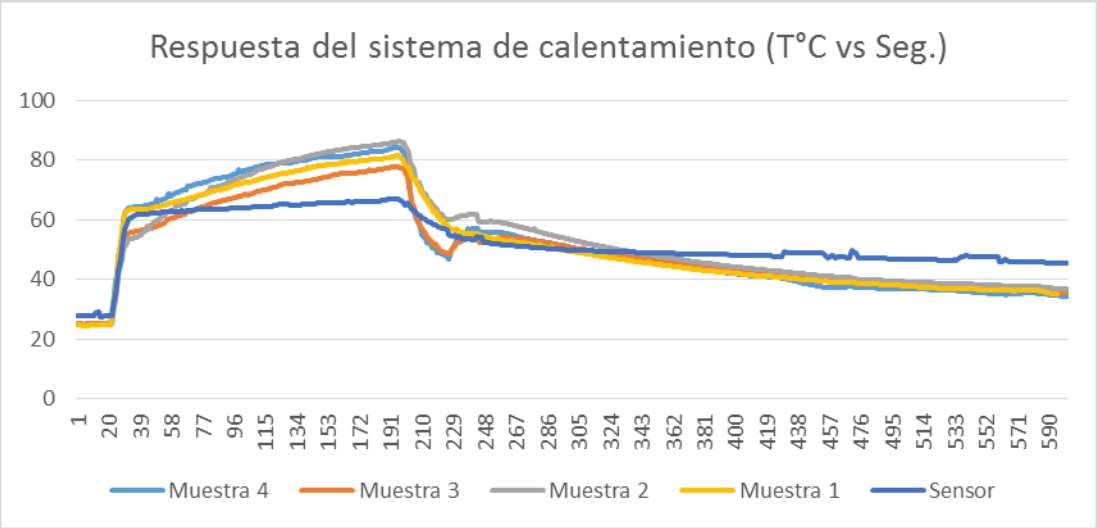


Figura 4.27: Respuesta del sistema de calentamiento vs Sensor de referencia.

Se puede apreciar que las cuatro muestras tienen una dinámica similar con temperaturas máximas de 80°C aproximadamente, pero la temperatura en el sensor difiere teniendo un valor máximo de 65°C. Se realizaron pruebas reubicando el sensor de referencia en diversas posiciones debajo de la campana (Ver anexo H). En estas pruebas se observó que la temperatura de los portaobjetos sin muestra ni reactivos crece continuamente. Posteriormente se realizan pruebas con el portaobjetos cubierto de fucsina, y dado que la variable a controlar es temperatura se utiliza un control ON/OFF. Este control de temperatura se realiza con un margen de cinco grados de error hacia abajo, con el fin de evitar que se sobrepase la referencia, pues el proceso de calentamiento es lento. En otras palabras, cuando se fija la referencia en 75°C, las lámparas reciben la potencia máxima hasta que el sensor llega a 70°C con Ton =7.33ms (máxima potencia u ON) y sin ángulo de disparo (lámpara apagada u OFF). La figura 4.28 muestra el diagrama de flujo del control ON/OFF. En el capítulo 6 se presentan los resultados finales del control ON/OFF implementado.

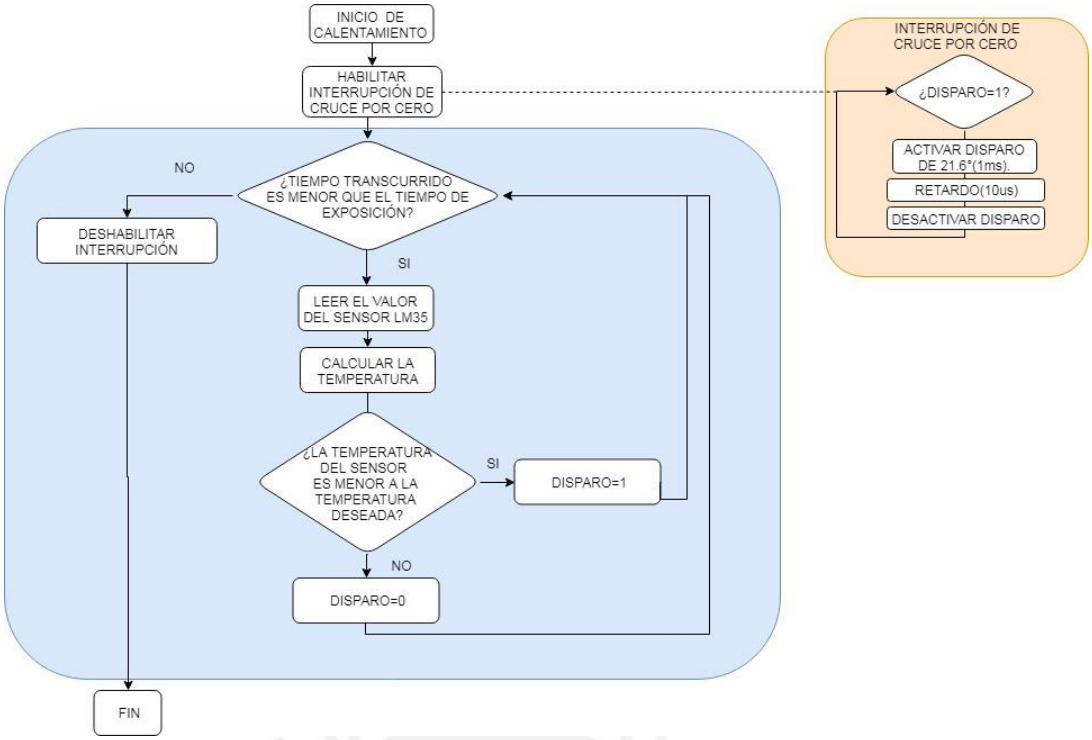


Figura 4.28: Control del calentamiento

4.5.6 Subrutinas para el control del motor a paso

Estas subrutinas muestran el control del motor a pasos tanto de la dirección como de los pulsos que definen el movimiento del motor. En la figura 4.29 se muestran las subrutinas.

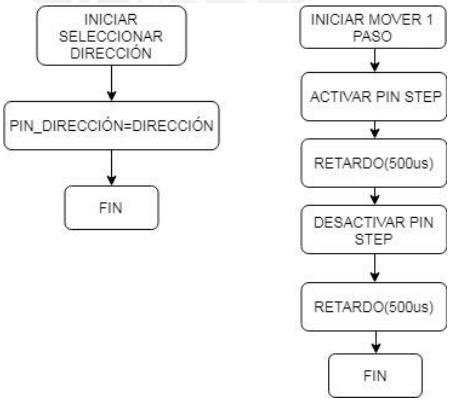


Figura 4.29: Subrutinas del control del motor a paso

CAPÍTULO 5:
DISEÑO E IMPLEMENTACIÓN DEL MÓDULO DE MICROSCOPÍA.

En el presente capítulo se muestra los alcances del módulo de microscopio; posteriormente, se presenta el diagrama de bloques para luego exponer el diseño del hardware y software electrónico. Finalmente, se expone el estudio del sistema mecánico del prototipo. Para ello se establecen los

requerimientos, alternativas, selección y el diseño propuesto para los elementos analizados del módulo de microscopio.

5.1 Alcances

El diseño e implementación del subsistema del desplazamiento y autoenfoque del módulo de microscopía, abarca el estudio de la mecánica de transporte de las muestras del microscopio, el diseño y la implementación del algoritmo de automatización del enfoque y el diseño e implementación del sistema de iluminación. Adicionalmente, se implementa un algoritmo de conteo de bacilos y una interfaz gráfica.

5.2 Diagrama de bloques

El sistema está conformado por la planta, hardware y software. El hardware contempla una plataforma móvil, un sistema de iluminación, un sistema de visión, una interfaz de usuario y la fuente de alimentación. El software contempla los programas del microcontrolador y los programas de enfoque, conteo e interfaz en la Raspberry. La plataforma se desplaza a un punto de inserción de muestras de tal forma que el usuario coloca los 4 portaobjetos con las muestras provenientes del subsistema de tinción, agregando una gota de aceite para microscopio a cada muestra. Posteriormente, la plataforma se desplaza a la posición del ocular del microscopio, donde la cámara realiza el autoenfoque y conteo de bacilos para cada muestra. En la posición del ocular se encuentra el sistema de iluminación que permite que las muestras puedan ser escaneadas por una cámara a través del microscopio. Finalmente, los resultados son presentados al usuario través de una interfaz gráfica. La figura 5.1 muestra el diagrama de bloques que esquematiza el proceso.

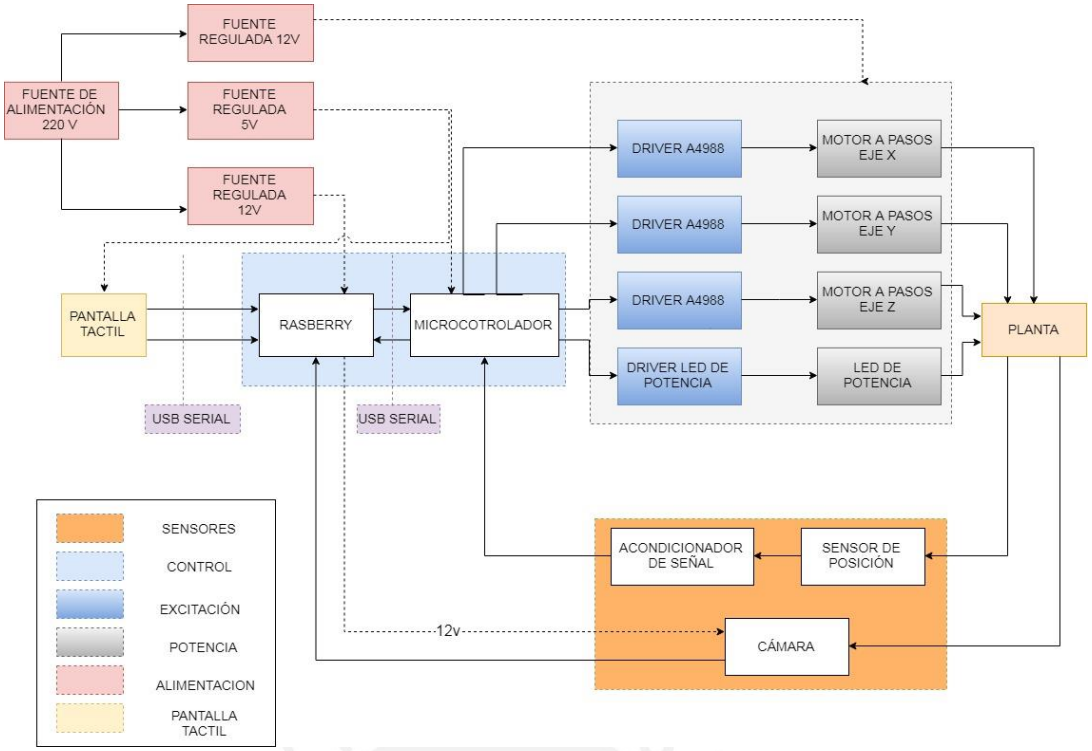


Figura 5.1: Diagrama de bloques del módulo de microscopía

5.3 Estructura mecánica

El sistema mecánico se encuentra conformado por 3 estructuras principales: el soporte principal del sistema, una estructura móvil que contiene y desplaza las muestras y la estructura del microscopio. La plataforma móvil original del microscopio ha sido reemplazada por una plataforma de 4 muestras (estructura móvil). Se realizó un corte al microscopio original al fin de acoplar la parte óptica (estructura del microscopio) a la estructura principal. Sobre esta estructura principal se realiza la sujeción de la pantalla táctil, la tarjeta electrónica y demás componentes electrónicos (motores y LED). Estos componentes se encuentran ocultos solo con acceso para realizar reparaciones y mantenimientos. El microscopio está alineado con el sistema de iluminación de tal forma que la luz emitida por el LED de potencia pasa a través de una perforación en la plataforma, iluminando la bandeja que contiene las muestras. La cámara que permite la digitalización de las muestras tiene un soporte que la fija a la estructura del microscopio. Esta estructura se encuentra fija sobre la estructura principal siendo la plataforma

la única que se desplaza en todos los ejes, tanto para el ingreso de las muestras como para el autoenfoque. La figura 5,2 muestra el sistema mecánico del prototipo.

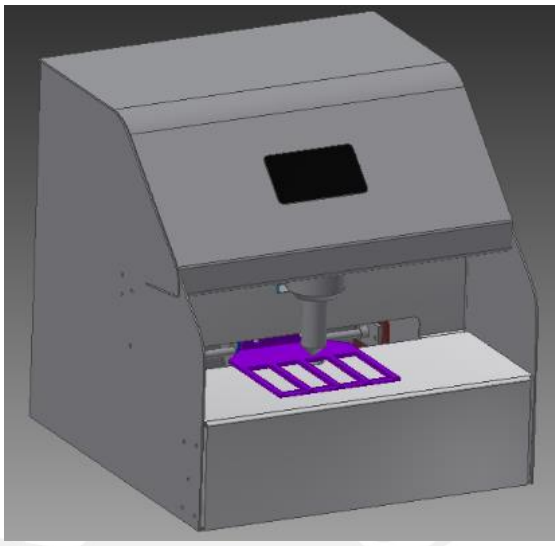


Figura 5.2: Estructura o planta del módulo de microscopía

5.4 Diseño e implementación del módulo.

5.4.1 Hardware Electrónico:

5.4.1.1 Sensores y acondicionadores de señal

El módulo automático de microscopía para cuatro muestras de esputo en simultáneo cuenta con los siguientes sensores:

- Una cámara para la adquisición de las imágenes de los cuatro portaobjetos con las muestras de esputo.
- Un sensor de posición para obtener la ubicación de referencia en la que se encuentran las muestras.

5.4.1.1.1 Cámara digital

El sistema utiliza la cámara digital C920E WEBCAM de la figura 5.3. La tabla 5.1 muestra los requerimientos y las características de la cámara. Esta se encuentra acoplada al tubo del microscopio a través de una estructura creada digitalmente en impresión 3D.

Figura 5.3: Cámara digital Pro C920 WEBCAM

Tabla 5.1: Requerimientos y características de la cámara digital

Requerimientos	C920E WEBCAM	
	Característica	¿Cumple?
Resolución	1920×1080	Sí
Marcos por segundo (frame rate)	30	Sí
Campo de visión	78°	Sí
Alimentación	USB	Sí

La cámara se alimenta a través del puerto usb de la placa de la Raspberry Pi 3 tal cual se muestra en la figura 5.4.

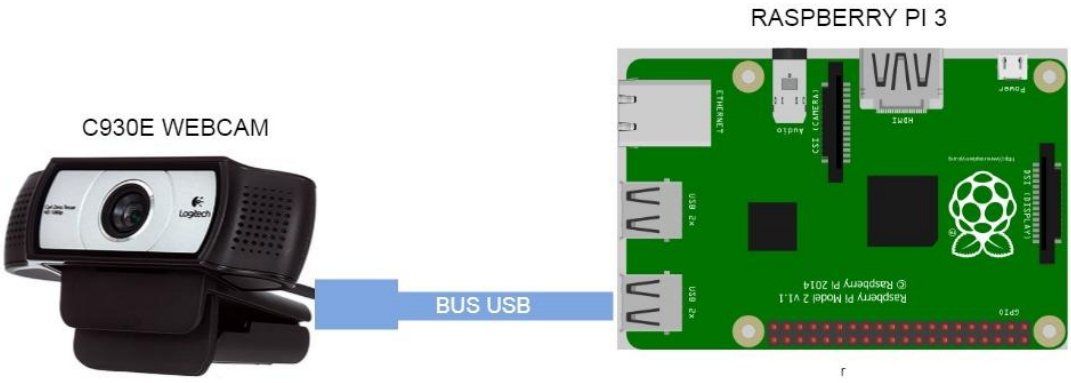


Figura 5.4: Conexión entre la cámara Webcam y la Raspberry pi 3

5.4.1.1.2 Sensor de posición

El sistema cuenta con un sensor de posición del tipo interruptor fin de carrera como lo muestra la figura 5.5. La tabla 5.2 muestra los requerimientos y las características de este sensor.

Figura 5.5: Interruptor de fin de carrera

Tabla 5.2: **Requerimientos y características del sensor de posición fin de carrera.**

Requerimientos	Fin de carrera	
	Característica	¿Cumple?
Precisión menor o igual a 0.1 cm.	[+/- 0.05 cm]	Sí
Bajo tiempo de respuesta.	[125 μs]	Sí

Este sensor presenta dos estados On-Off, se mantiene abierto cuando la plataforma móvil que transporta los portaobjetos no está en contacto con este sensor y se cierra cuando la estructura lo mantenga presionado. La figura 5.6 muestra el diagrama esquemático del circuito de sensado de posición.

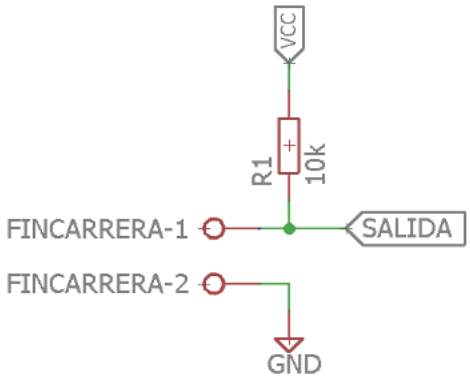


Figura 5.6: Diagrama esquemático del circuito de sensado de posición

Este circuito cuenta con una resistencia pull-up que brinda una salida de 0 V cuando el switch está presionado, caso contrario la salida será de 5V. La resistencia R1 debe ser alta de manera que consuma una baja corriente, por lo tanto se opta por escoger un valor de 10 KΩ.

5.4.1.2 Actuadores y excitadores

El módulo automático de microscopía cuenta con los siguientes actuadores:

- Un LED de alta potencia para el subsistema de iluminación.
- Tres motores que permiten el desplazamiento de las muestras en los tres ejes.

5.4.1.2.1 Led de potencia

El sistema utiliza como fuente de iluminación un LED de potencia como se muestra en la figura 5.7. En la tabla 5.3 se muestran los requerimientos y las características del LED de potencia.

Figura 5.7: Led de potencia

Tabla 5.3: Requerimientos y características del LED de potencia.

Requerimientos	LED de potencia	
	Característica	¿Cumple?
Intensidad luminosa – Lumens	Supera los 500lm	Sí
Alimentación	12V o 5V	Sí
DC current	900 mA	Si
Temperatura de color	6000-6500K	Si
Intensidad de iluminación	700lm	Si
Debe ser compacto.	2 cm de radio	Sí

5.4.1.2.2 Excitador del Led de potencia

La tensión de alimentación del LED es de 12V y su corriente nominal de 300mA, el excitador es el MOSFET de potencia IRF530 que soporta una tensión V_{DS} de 100 V y una corriente de hasta 14 A. Además, se utiliza una resistencia pull-down R1 de 100 KΩ entre los terminales G-S del MOSFET para evitar el ruido eléctrico.

Por otro lado, la corriente del LED se controla a través de la modulación por ancho de pulso (PWM) de la señal LUZ la cual será activada por el

microcontrolador. En la figura 5.8 se muestra el diagrama esquemático del circuito excitador del LED de potencia.

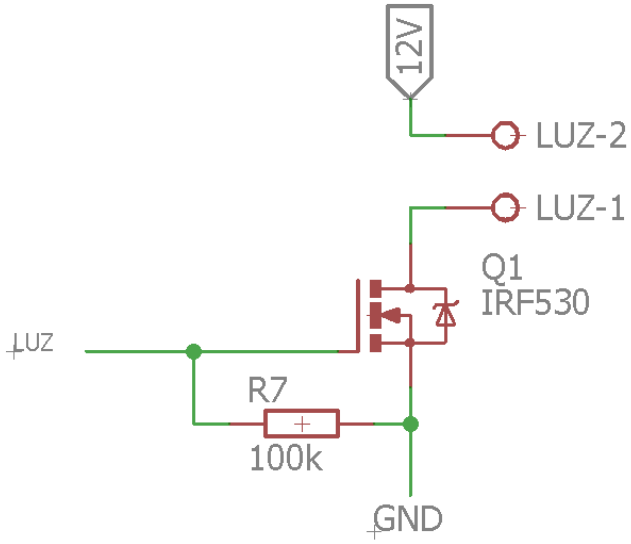


Figura 5.8: Diagrama esquemático del circuito excitador del LED de potencia

5.4.1.2.3 Motores para el desplazamiento de las muestras

El sistema mecánico consta de 3 motores a paso de los cuales se han realizado los cálculos que se describen a continuación, con el fin de poder tener la seguridad de poder desplazar la plataforma a través de los campos de las muestras visualizados por el microscopio.

El motor puede girar hasta 1/16 de un paso de 1.8°. Es decir, puede rotar 0.1125°. El paso entre hilos del tornillo sin fin usado es de 2 mm (360°) para los ejes X e Y. Por lo tanto, un giro de 0.1125° corresponde a un desplazamiento lineal de 0.000625mm. Para pasar al siguiente campo de 0,2 mm (Aumento 100x del objetivo), el motor debe girar 36° o 20 pasos. En el caso del eje z hay un hilo de 1 mm, por lo tanto, una rotación de 0.1125° corresponde a 0.0003125mm.

Para el desplazamiento de las muestras el sistema utiliza el motor a pasos XY42STH34-0354A que se muestra en la Figura 5.9. En la tabla 5.4 se muestra los requerimientos y las características para cada eje del motor. Los

desplazamientos en el plano XY no necesitan ser reajustados a cada momento por lo que se configura solo para un micropaso según la tabla 5.5. En el caso del movimiento del eje Z, es necesario reajustar los micropasos según los requerimientos del algoritmo de autoenfoque.

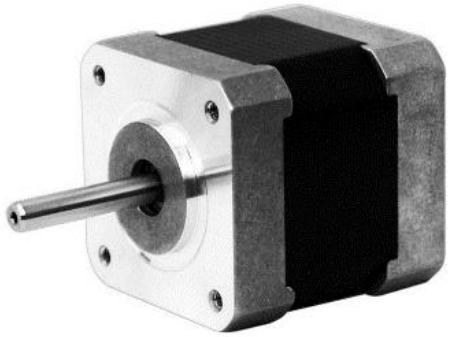


Figura 5.9: Motor a pasos XY42STH34-0354A

Tabla 5.4: Requerimientos y características de los motores a pasos.

Requerimientos	XY42STH34-0354ª	
	Característica	¿Cumple?
Torque (mayor a 24.53 mN.m)	[300 mN.m]	Sí
Velocidad mínima (76.4 rpm)	[600 r.p.m.]	Si
Potencia mecánica (mayor a 2.57 W)	[3 W]	Sí
Bajo consumo de corriente.	[250 mA]	Sí
Tensión de alimentación (12 V)	[12 V]	Sí
Requiere sensor de posición	Sensado de posición por pasos	Si

5.4.1.2.4 Excitadores de Motores para el desplazamiento de las muestras

A continuación, las figuras 5.10 5.11 y 5.11 muestran los diagramas esquemáticos de los excitadores de los motores de paso para los ejes X, Z e Y respectivamente. Los motores a pasos se energizan con 5V y 12V/250mA para el circuito de control y potencia respectivamente.

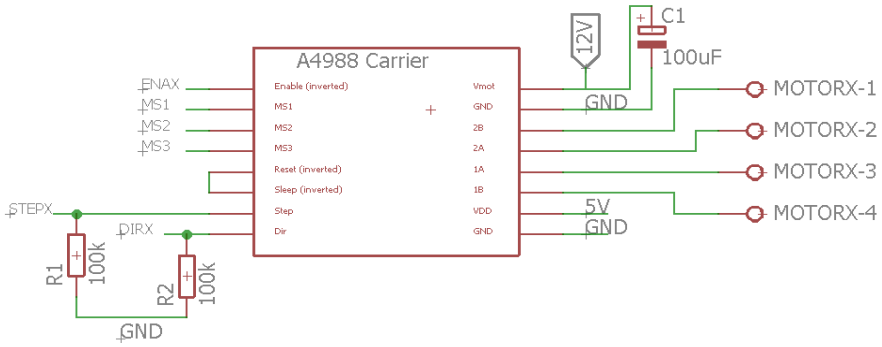


Figura 5.10: Esquemático del excitador del motor de paso del eje X

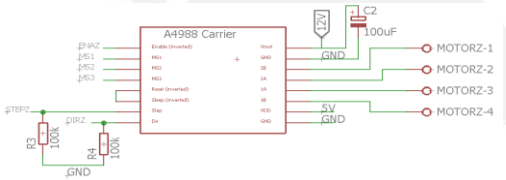


Figura 5.11: Esquemático del excitador del motor de paso del eje Z

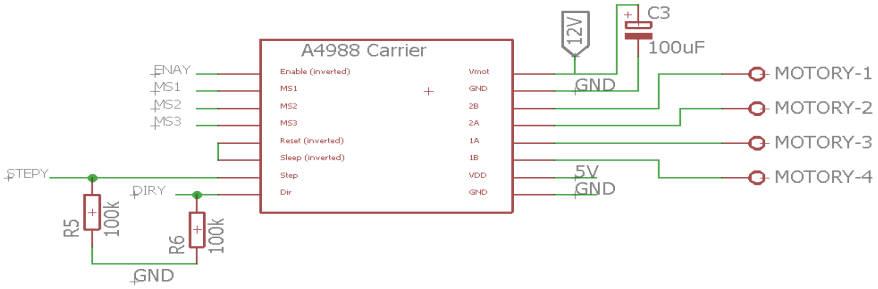


Figura 5.12: Esquemático del excitador del motor de paso del eje Y

El excitador A4988 permite el control del sentido de giro y velocidad del motor a pasos por medio de los pines MS1, MS2, MS3, STEP y DIR según la lógica mostrada en la tabla 5.5.

Tabla 5.5: Control de giro y parada del motor a pasos.

MS 1	MS2	MS3	Pasos
0	0	0	Paso completo
1	0	0	1/2 paso
0	1	0	1/4 de paso
1	1	0	1/8 de paso
1	1	1	1/16 de paso

STEP	DIR	GIRO
Pulso	0	Horario
Pulso	1	Antihorario
-	x	Parado

5.4.1.3 Unidad de procesamiento

La unidad de procesamiento está compuesta por la placa Arduino Nano basada en un microcontrolador ATmega328 y la placa Raspberry 3 que tiene un procesador ARM Cortex-A53 de 64 bits a 1.2 GHz. El primero se encarga de excitar los motores de los ejes X, Y y Z, y el LED de potencia a través de sus controladores. El segundo se encarga de procesar las imágenes adquiridas, tanto para encontrar la imagen mejor enfocada como para hacer el diagnóstico de las muestras; además de controlar la interfaz del usuario que permite iniciar, monitorear, interrumpir y finalizar el proceso de selección. La figura 5.13 muestra las conexiones entre ellos vía USB.

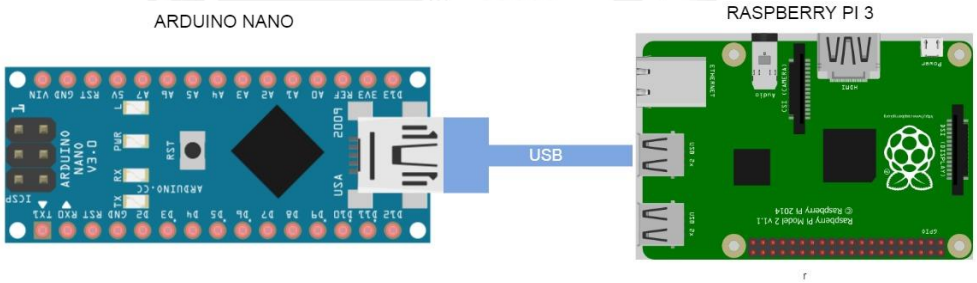


Figura 5.13: Conexión entre la tarjeta Arduino Nano y la Raspberry 3

5.4.1.4 Pantalla táctil

La pantalla táctil es un Waveshare de 7 pulgadas. La plataforma Raspberry se encarga de controlar la pantalla donde el usuario puede iniciar y finalizar el proceso fácilmente. La tabla 5.6 muestra los requerimientos y características de la pantalla. La Raspberry se conecta a través de USB y HDMI a la pantalla Waveshare.

Tabla 5.6. Características de la pantalla táctil

Pantalla Táctil Waveshare	
Característica	Valor
Tamaño	7”
Resolución	1024 x 600 pixeles
Sistema operativo Soportado	Windows 10/8.1/8/7 OS, Raspbian y Ubuntu Mate
Tecnología	Pantalla de cristal Líquido (LCD)
Puerto de datos	USB
Puerto de video	HDMI

En la figura 5.14 se muestran las conexiones del Raspberry con la Pantalla Wavashare.

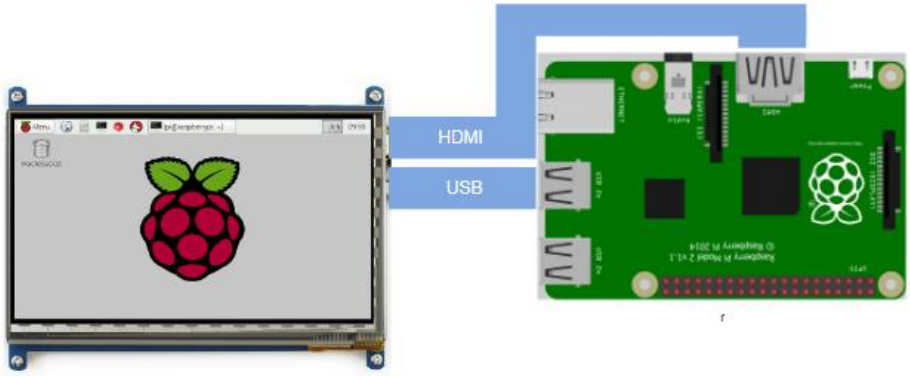


Figura 5.14: Interfaz de usuario y su conexión con la Raspberry

5.4.1.5 Fuente de alimentación

La tabla 5.7 muestra la corriente requerida por cada dispositivo relevante según la tensión que requiere.

Tabla 5.7 Requerimientos de las fuentes de alimentación.

Dispositivos de 5 V	I (mA)	Dispositivos de 12 V control	I (mA)	Dispositivos de 12 V Potencia	I (mA)
Micro-controlador	100	Jetson TK1	400	Motor a pasos	3X250
Excitadores A4988(3)	45	Cámara	500	LED de potencia	900
LED de potencia	35	-	-	-	-
Raspberry pi	2500	-	-	-	-
Pantalla	2000	-	-	-	
Total	4700	Total	900	Total	1650

Por lo tanto, el sistema cuenta con la fuente de alimentación conmutada RD65 de 12V/3A y 5V/6 A (figura 5.15).

Figura 5.15: Fuente switching del sistema.

5.4.2 Software:

5.4.2.1 Software para la plataforma Electrónica Arduino

La placa Arduino Nano basada en un microcontrolador ATmega328 funciona como esclavo del raspberry. El Arduino nano recibe los comandos de encendido del LED de potencia y accionado de los motores. Se encarga

también se sensar la señal de los sensores fin de carrera. El diagrama de flujo se presenta en la Figura 5.16. El código se encuentra en el anexo F.

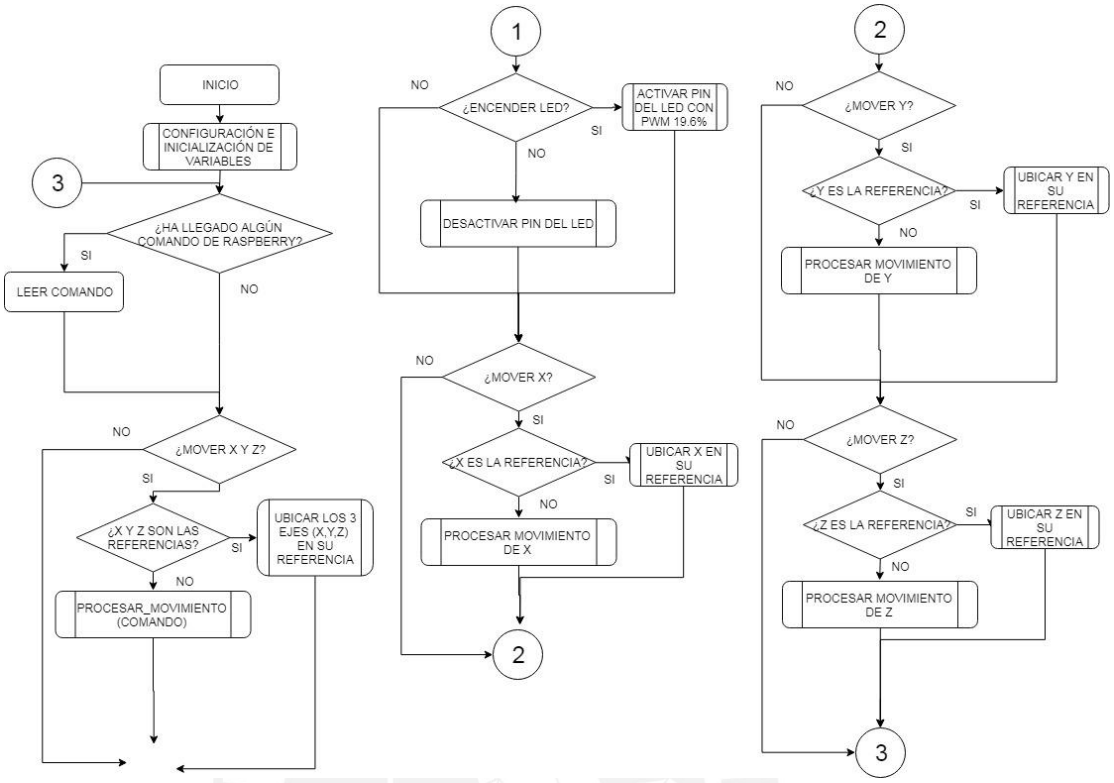


Figura 5.16: Diagrama de flujo general del software del microcontrolador

5.4.2.2 Software para la plataforma Raspberry

El diagrama de flujo principal se muestra en la figura 5.17. En este diagrama se puede apreciar las opciones con la que cuenta el usuario: Inicio del proceso, almacenamiento del diagnóstico y búsqueda de resultados anteriores. En esta figura también se puede ver el proceso PROCESO_MICROSCOPIO, este se ejecuta de acuerdo al diagrama mostrado en la figura 5.18, donde se aprecia todo el ciclo de microscopia. El algoritmo ha sido implementado en Raspberry y en el Anexo I se puede apreciar el código desarrollado.

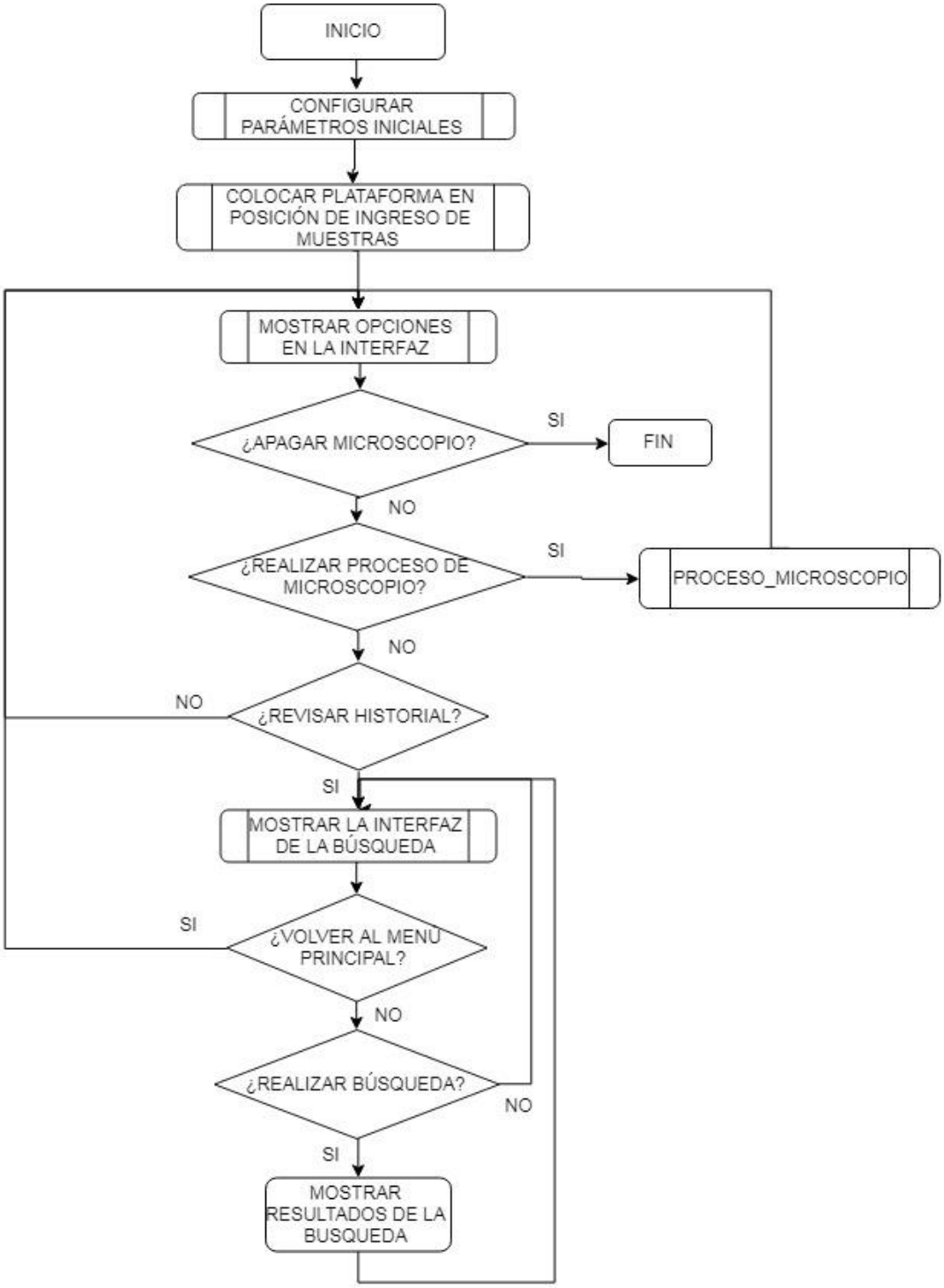


Figura 5.17: Diagrama de flujo del proceso general

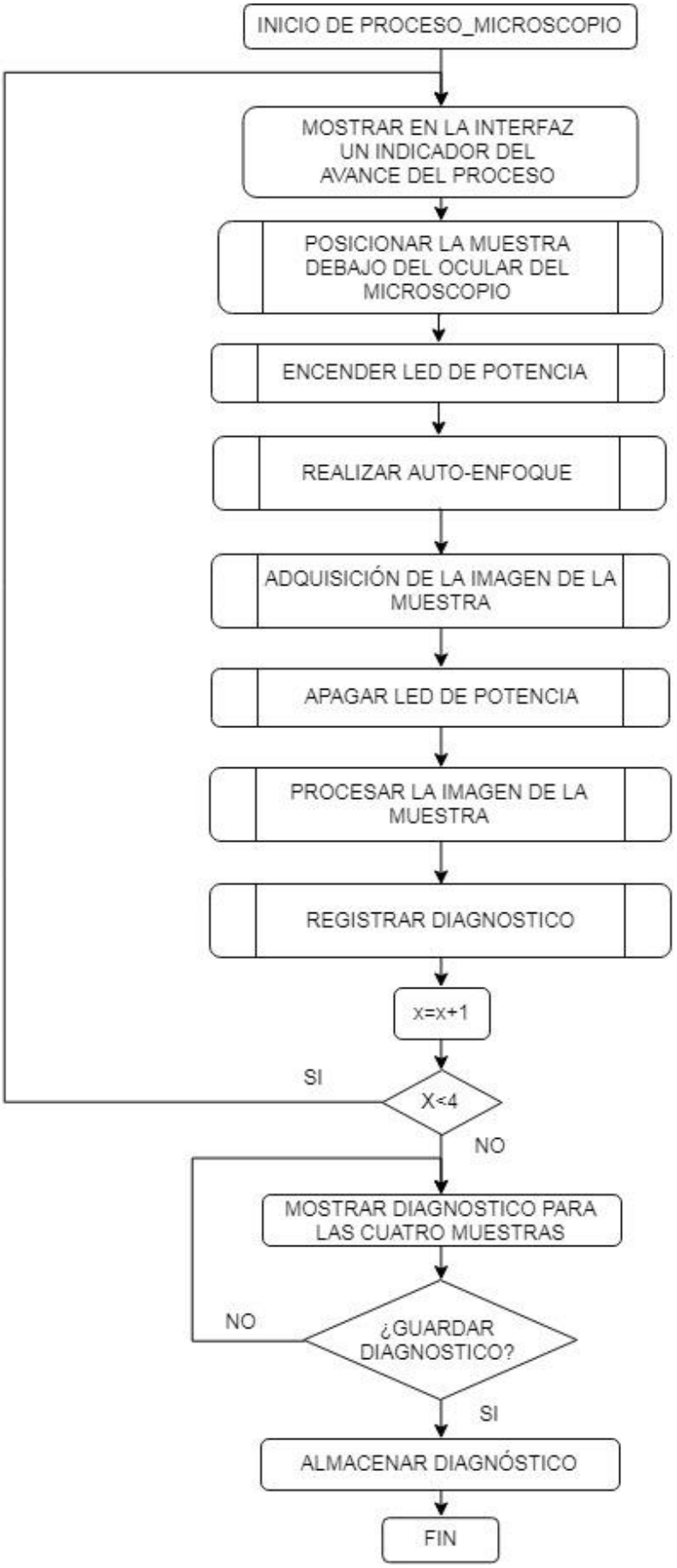


Figura 5.18: Diagrama de flujo del proceso de microscopía

5.5 Interfaz de usuario.

La interfaz permite al usuario iniciar el proceso de forma rápida y sencilla, adicionalmente se pueden guardar los diagnósticos. Entonces, terminado el proceso hay la posibilidad de almacenar el diagnóstico, para ello se deberá ingresar previamente el código de la muestra, como se muestra en la figura 5.19. El software ha sido implementado con mapeo objeto-relacional (ORM), la librería de python utilizada es *PonyORM*. Se ha empleado una base de datos local de tipo SQLite. Las búsquedas de los registros se pueden realizar con información de cualquiera de los campos (Código, Diagnostico, Campos, Bacilos y Fecha). En la figura 5.20 se muestra la evolución del proceso. En la figura 5.21 se muestra el resultado de la búsqueda de todos los campos que contengan el valor 13. El programa implementado se se encuentra en el Anexo I.

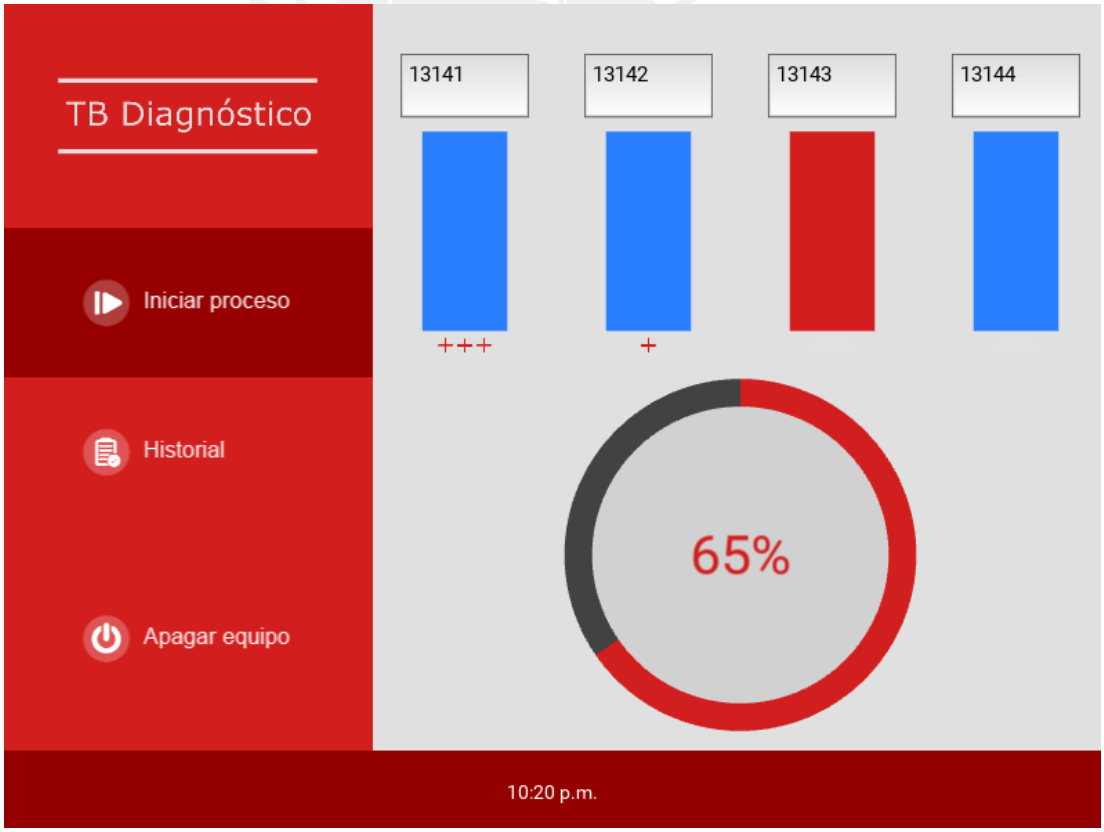


Figura 5.19: Interfaz de usuario: Inicio del proceso.

Figura 5.20: Interfaz de usuario: visualización del teclado táctil.

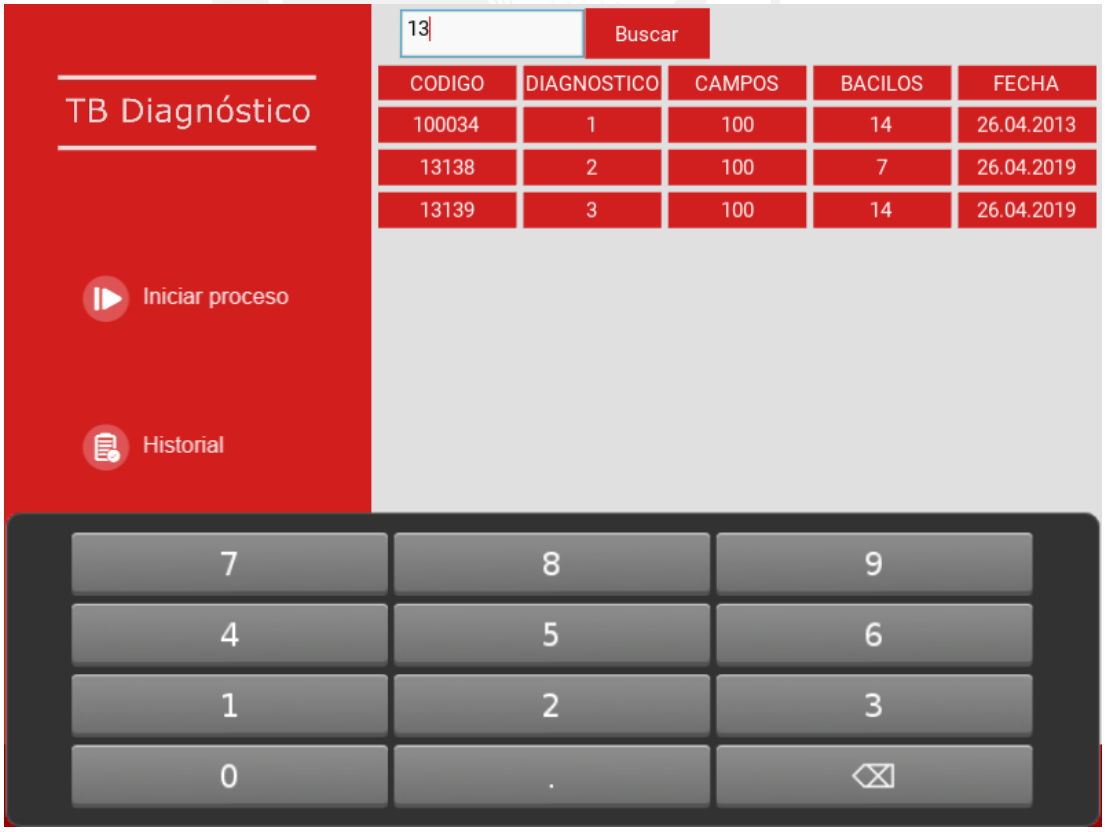


Figura 5.21: Interfaz de usuario: búsqueda de registros.

5.6 Software de autoenfoque

El microscopio comienza a escanear las muestras en la posición de inicio, que corresponde a la esquina superior izquierda del portaobjetos más a la izquierda y a una distancia de 2 mm en la dirección z. La figura 5.22 muestra un diagrama de la vista superior de la platina.

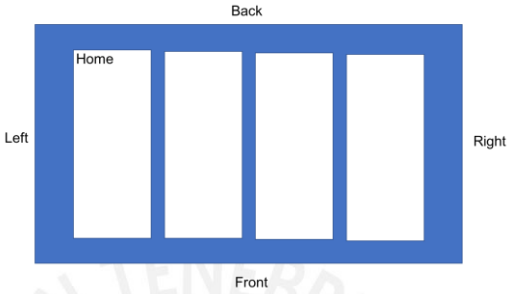


Figura 5.22: Vista superior de la plataforma que contiene hasta 4 muestras.

La placa se mueve hacia arriba mientras que la cámara digital está conectada al microscopio capturando imágenes. La transformada de Fourier en cada micro-paso (0.3um) se calcula sobre la marcha, mientras que el espectro es analizado.

Una imagen enfocada contiene bordes bien definidos los que corresponden a las frecuencias altas, por lo tanto, se espera un cambio en los espectros a medida que la imagen sea enfoca. En la figura 5.23 se muestra una comparación de los espectros de las imágenes enfocadas y desenfocadas. Para el diseño del algoritmo se han usado imágenes distorsionadas mediante filtros blur. Las imágenes correspondientes se muestran en la Figura 5.24.

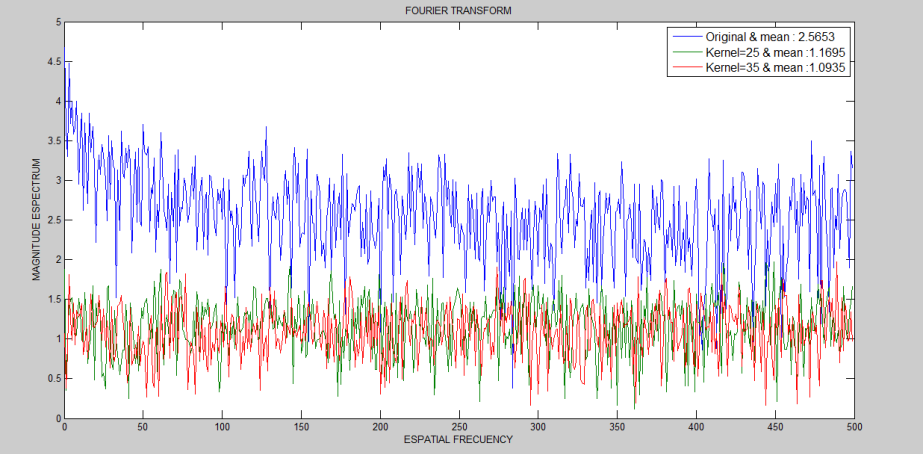


Figura 5.23: Comparación del espectro para una misma imagen enfocada (Azul) y no enfocadas (rojo y verde). Se nota el cambio de las amplitudes en la porción de alta frecuencia del espectro.

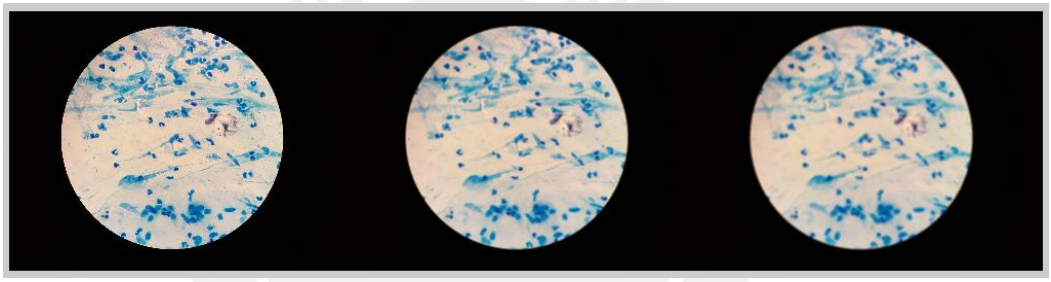


Figura 5.24: Imágenes enfocadas y desenfocadas correspondientes al espectro mostrado en la figura 5.23.

Al momento de la implementación se pudo corroborar, que no era suficiente promediar las componentes de altas frecuencias correspondientes solo a un plano de la superficie gausseana formada por el espectro espacial. Por lo que considerando que el sistema comenzará siempre desde una posición desenfocada conocida, se utiliza la primera imagen desenfocada como filtro pasa altos. Es decir se atenúan las componentes de baja frecuencia de la imagen analizada, restándole a esta en frecuencia la primera imagen desenfocada. Finalmente, se calcula el promedio de todas las filas y columnas del espectro espacial. Los resultados tomando como referencia las imágenes de [Shah M. I., 2017] se presenta en capítulo 6.

El código diseñado e implementado en python se encuentra en el anexo D.

5.7 Implementación de un algoritmo de detección y diagnóstico (conteo) de patrones BAAR.

En este apartado de describe el proceso de conteo de bacilos a partir de una imagen tomada desde un microscopio. Esta imagen es convertida al espacio de color CIELAB. Una vez rotado el espacio a-b 110° se realiza una umbralización con el fin de clasificar los pixeles del color de los bacilos. Al resultado de la umbralización se le realiza una esqueletización de tal forma que se elimina el fondo con sangre u otros microbios que no corresponden a los bacilos; se usó la librería *scikit-image*. Los esqueletos que tengan un punto de inicio y un punto final unidos por una delgada línea serán los correspondientes a los bacilos. Adicionalmente, se realiza un filtro para eliminar las líneas que contengan agujeros y finalmente se realiza un filtro por tamaños. Este algoritmo ha sido diseñado en MATLAB por los miembros del equipo (Cataño, M y Helguera, M), es parte de la presente tesis la implementación en python de este algoritmo, para lo cual se han realizado modificaciones en los algoritmos individuales pues varias funciones de MATLAB no se encuentran listas en Python. Es así que la cantidad de operaciones morfológicas son las mismas pero implementadas a través de diferentes algoritmos. A continuación se describe brevemente los algoritmos utilizados e implementados en python.

La operación de eliminar agujeros se ha realizado a través de la identificación de contornos y jerarquía de los objetos en las imágenes inicialmente etiquetadas. Es decir, se recorre cada contorno y si un contorno padre tiene algún contorno hijo, quiere decir que posee un agujero. En consecuencia, este contorno y los hijos se eliminan. La figura 5.25, muestra esta operación.

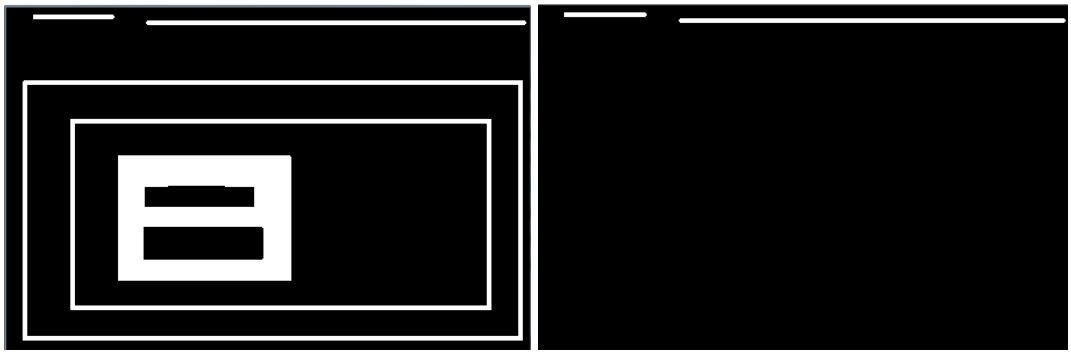


Figura 5.25: Operación de eliminación de agujeros.[16]

El algoritmo realiza tres veces etiquetados con conectividad 8. El primer etiquetado se realiza una vez hecha la segmentación por color. El segundo etiquetado se realiza una vez que se han eliminado los objetos que no tienen dos puntos extremos. El tercer etiquetado se realiza cuando se eliminan los objetos que tienen agujeros. La figura 5.26 a, b y c muestran el resultado luego de los etiquetados.

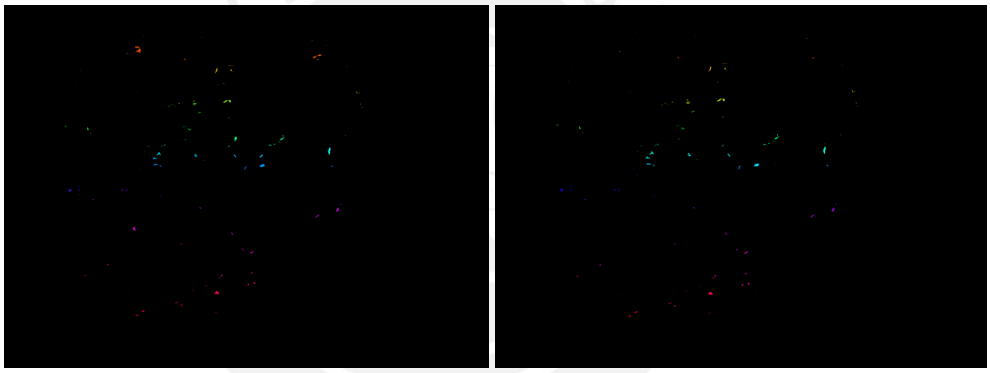


Figura 5.26 a) **Figura 5.26 b)**

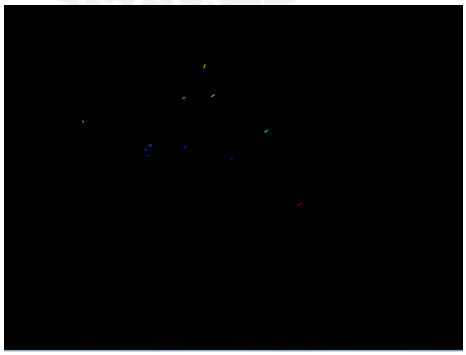


Figura 5.26 c)

Figura 5.26: Etiquetado de objetos. a) Posterior al filtro de color, b) Posterior al filtro de puntos extremos, c) Posterior a la eliminación de agujeros

[16] Fuente: Contours Hierarchy — OpenCV - Python Tutorials 1 documentation.

La figura 5.27 presenta una imagen de una muestra de esputo obtenida a través de un microscopio y el resultado después de la aplicación de las operaciones morfológicas.

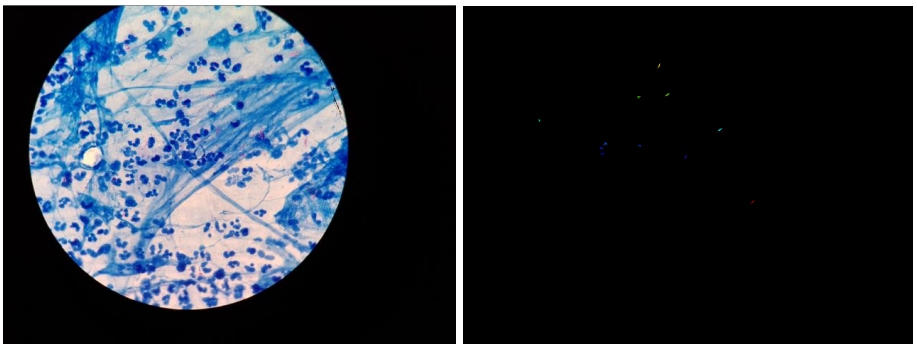


Figura 5.27: Imagen original e imagen final

A continuación en la figura 5.28 se muestra el diagrama de flujo del conteo de bacilos. El código implementado en Python se encuentra en el anexo E

Figura 5.28: Diagrama de flujo de detección y conteo de bacilos

CAPÍTULO 6: RESULTADOS Y DISCUSIÓN

6.1 Resultados.

6.1.1 Resultados del módulo de tinción

Se tiñeron 206 muestras con el equipo, de las cuales las primeras 156 muestras se usaron para calibrarlo (más del 95% de estas muestras eran negativas). Las últimas 50 muestras eran todas positivas, 30 ellas se usaron para calibrar el equipo y realizar mediciones. Las últimas 20 (5 secuencias) dieron como resultado 11 positivas vistas manualmente desde un microscopio no automático. En la figura 6.1 se presentan los resultados de la muestra 13139, donde se puede apreciar la coloración de los bacilos.

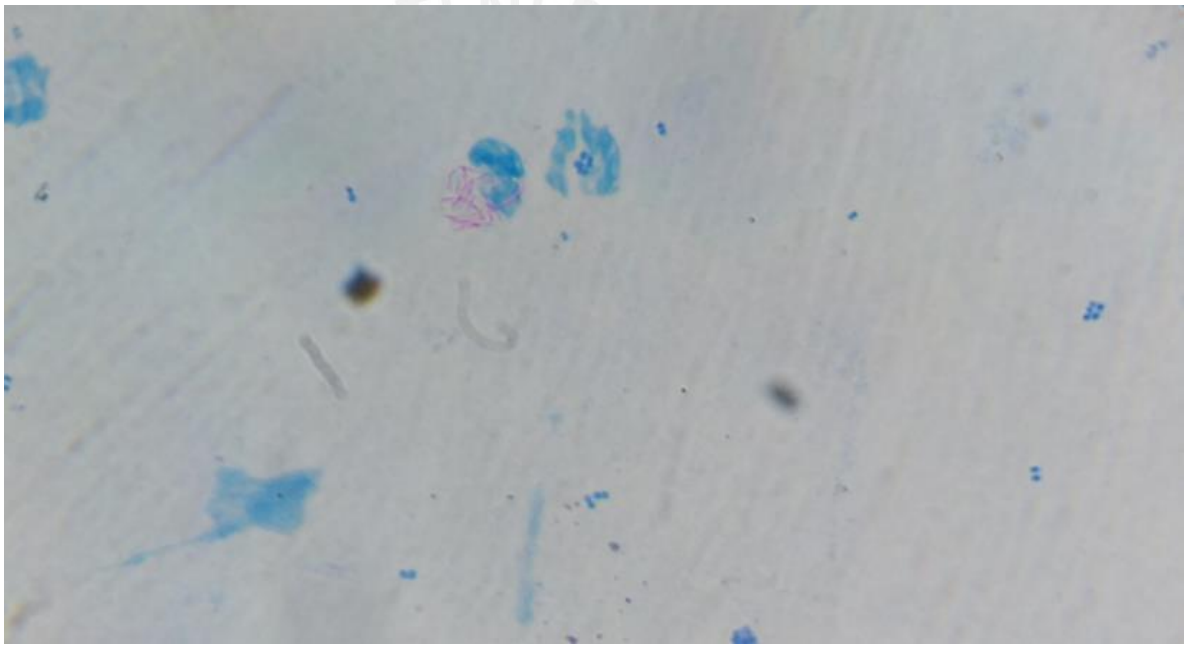


Figura 6.1: Fracción de un campo de una muestra coloreada por el módulo de tinción

En las siguientes líneas se analiza y discute las condiciones y factores que influyeron durante el proceso de tinción automático.

6.1.1.1 Factores determinantes

6.1.1.1.1 Inclinación de las muestras

Las muestras son sujetas por pinzas, dichas pinzas rompen la tensión superficial al calentarse. El calentamiento de las pinzas provoca que la fucsina fluya y caiga de la muestras. Una inclinación adecuada permite que la mayor cantidad de fucsina se concentre lejos de la pinzas durante el calentamiento. Es así que, la inclinación que favorece el recubrimiento con fucsina durante el calentamiento, es perjudicial para el almacenamiento de todos los reactivos, ya que estos se empozan en la parte delantera de los recipientes. Este almacenamiento impide que el actuador pueda bombearlos, pues la bomba se ubica del lado opuesto, ocasionando que queden reactivos en los recipientes.

En la figura 6.11 se puede apreciar que las muestras están recubiertas adecuadamente y tienen una inclinación hacia adelante a fin de evitar la conducción del calor de las pinzas hacia la muestra.

6.1.1.1.2 Tuberías de dispensado de fucsina:

Las tuberías neumáticas usadas para el dispensando de fucsina se hinchan, obstruyen e incluso desprenden partículas. La figura 6.2 muestra la tubería después de más de 15 secuencias.

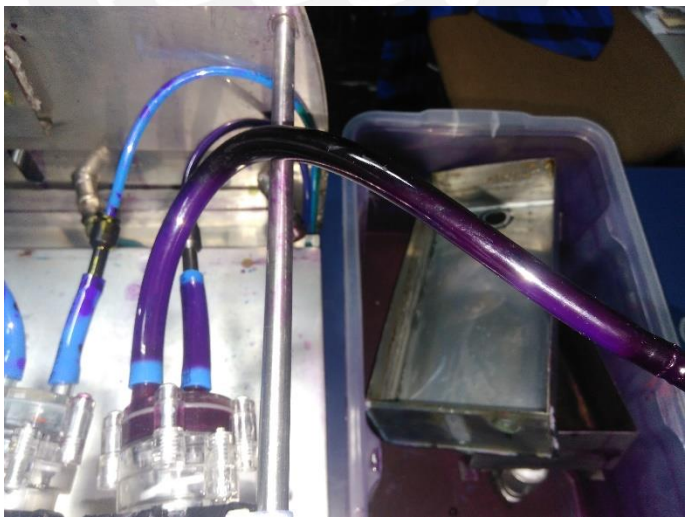


Figura 6.2: Tuberia de dispensado de fucsina

Es importante mencionar que el dispensador (no la tubería) del canal 1 quedó obstruido después de 40 secuencias.

A continuación en la figura 6.3 se puede observar que la coloración de la fucsina de la muestra de la izquierda, es diferente a la coloración normal de la muestra de la derecha. Esta coloración evidencia la contaminación de la fucsina.

Figura 6.3: Muestra con fucsina proveniente del equipo (izquierda) y fucsina no contaminada (derecha)

6.1.1.1.3 Temperatura del calentamiento

La temperatura de calentamiento puede ocasionar que las muestra queden secas y/o de coloración obscura. Al cambiar la inclinación del equipo, las lámparas halógenas se acercan más a las muestras lo que hizo necesario reducir la temperatura. En la posición horizontal la temperatura de referencia, por recomendación del tecnólogo médico, se fijó en 100°C, mientras que cuando se inclinaba hacia adelante (por desajuste de la estructura del equipo que acerca las muestras a la campana), la temperatura adecuada de referencia se fijó en 93°C.

En la figura 6.4 (izquierda) se puede apreciar muestras que tienen un exceso de temperatura, mientras que en la figura 6.4 (derecha) la temperatura es adecuada.

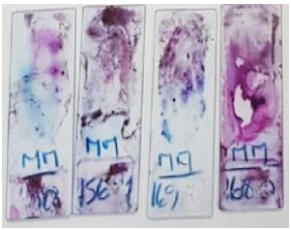
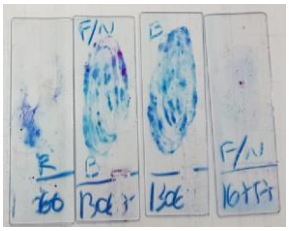


Figura 6.4: Muestras teñidas con exceso de calentamiento (Izquierda) y Muestras teñidas con calentamiento adecuado (Derecha).



Con el control de temperatura activo, se muestran la evolución de las temperaturas en el sensor de referencia y en cada una de las 4 muestras. Para la medición de la temperatura de cada muestra se utilizó las termocuplas de 4 equipos FLUKE 179 con sensor de temperatura. Se ha realizado una validación del control para diversas temperaturas. En las figuras 6.5 y 6.6 se muestran los resultados para temperatura de referencias de 60° y 70°. Dada la indisponibilidad de fucsina y a que las pruebas se realizaron en el laboratorio SIBA de la PUCP, se usó agua en lugar de fucsina. El resultado podría variar levemente para el caso de fucsina.

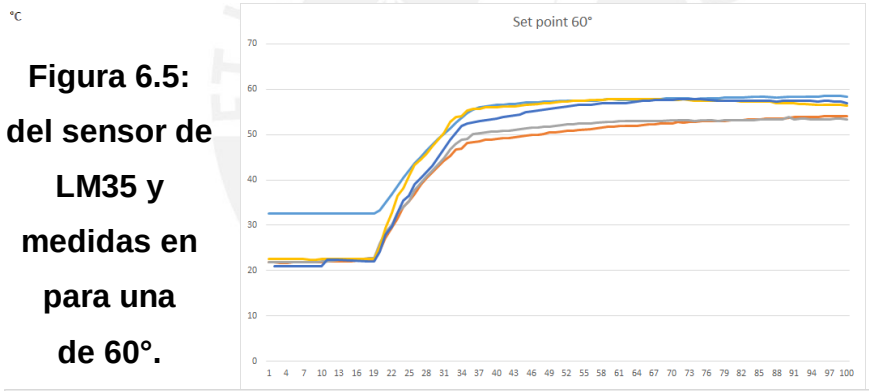


Figura 6.5: Respuesta del sensor de temperatura LM35 y temperaturas medidas en las muestras para una referencias de 60°.

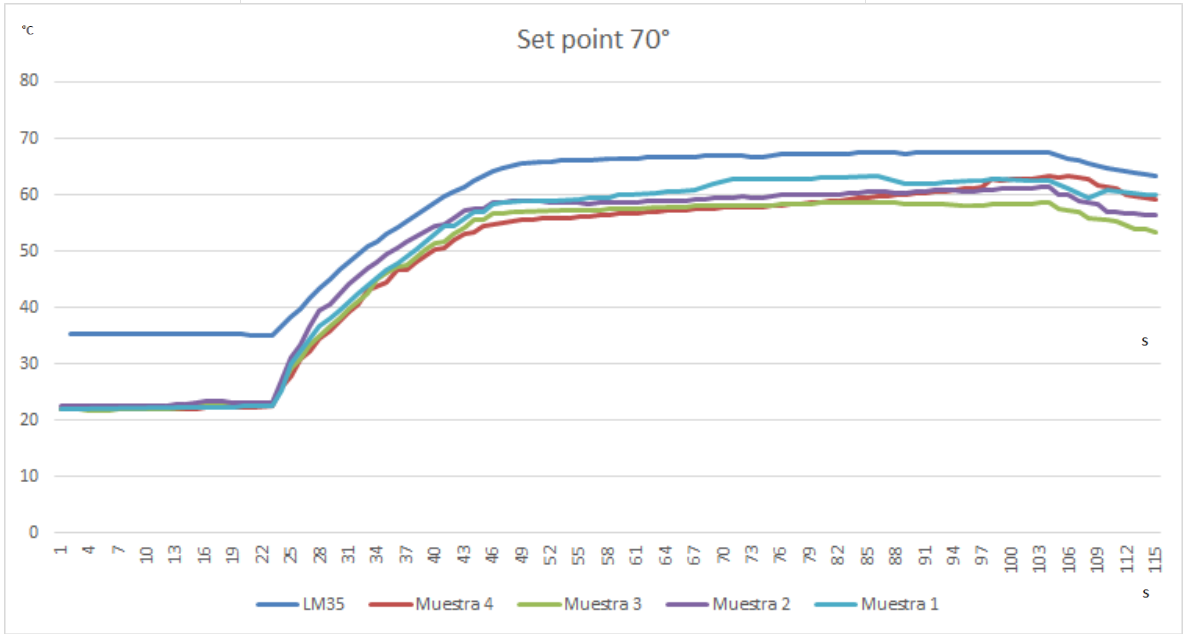


Figura 6.6: Respuesta del sensor de temperatura LM35 y temperaturas medidas en las muestras para una referencias de 70°.

A continuación se presentan las gráficas que muestran la respuesta en el tiempo de temperatura. La figura 6.7 muestra los valores del sensor de referencia durante el calentamiento. El resultado del control de temperatura es para la secuencia de muestras 13063 (93°C) y para la secuencia 16921 (100°C), medidas a condiciones ambientales de 21° y 21.6° respectivamente, unidas en una sola gráfica. Para la referencia de 93°C la posición del calentamiento está dada por el ajuste 4 de la Tabla 6.1, mientras que para la referencia 100°C está dada para el ajuste 3. El tiempo de calentamiento es de setenta y cinco segundos. Para el caso de la secuencia 13063 hubo un precalentamiento de calibración del equipo, que tiene una referencia de 60°, este precalentamiento se da cada vez que se reinicia el equipo. El precalentamiento se puede observar en la figura 6.8.

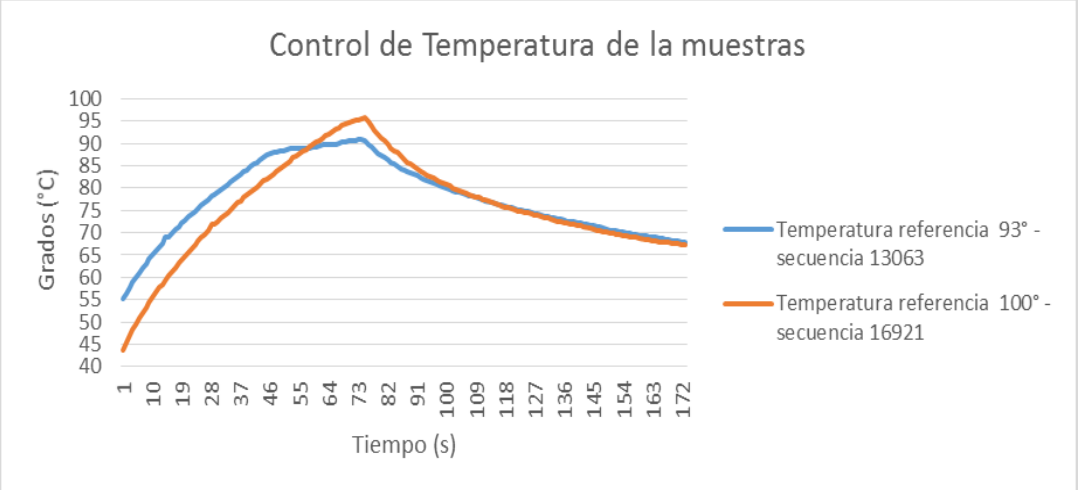


Figura 6.7: Temperatura del sensor de referencia durante el calentamiento

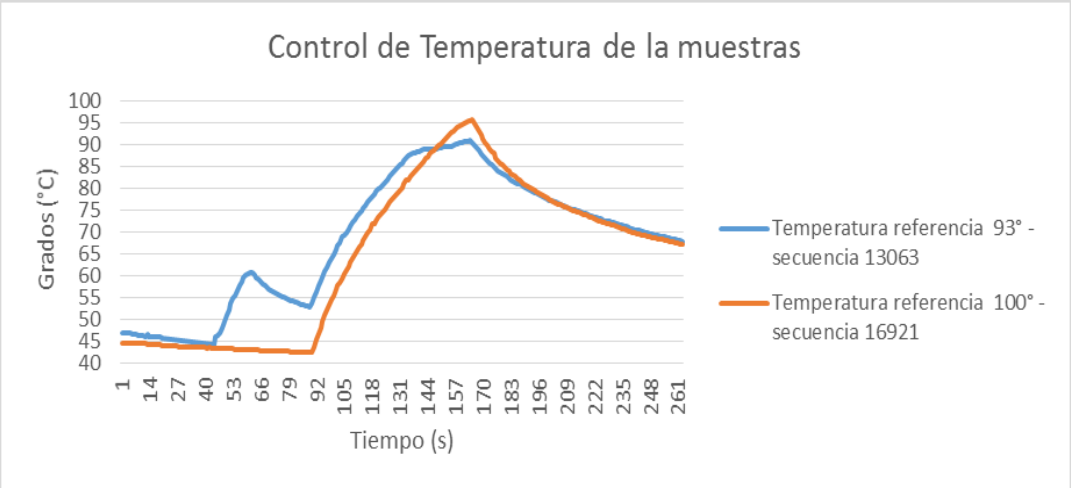


Figura 6.8: Temperatura del sensor de referencia durante el precalentamiento

Como se puede apreciar en las figuras 6.7 y 6.8 el tiempo de establecimiento de temperatura es lento. En consecuencia, dado el corto tiempo de calentamiento no se puede apreciar con detalle el comportamiento en régimen estacionario. Entonces, para validar el control ON/OFF, se realizaron 4 secuencias de pruebas. A continuación en las figuras 6.9 y 6.10 se muestran 2 de los resultados de estas pruebas del control de temperatura. Para estas pruebas se utilizaron el Multímetro Fluke 179 con termocupla para medir la temperatura en el portaobjetos recubierto de fucsina y el Multímetro Fluke 175 para medir la temperatura del sensor LM35. Se puede observar que la temperatura de diferencia es de 10 °C y que el control se realiza de forma apropiada.

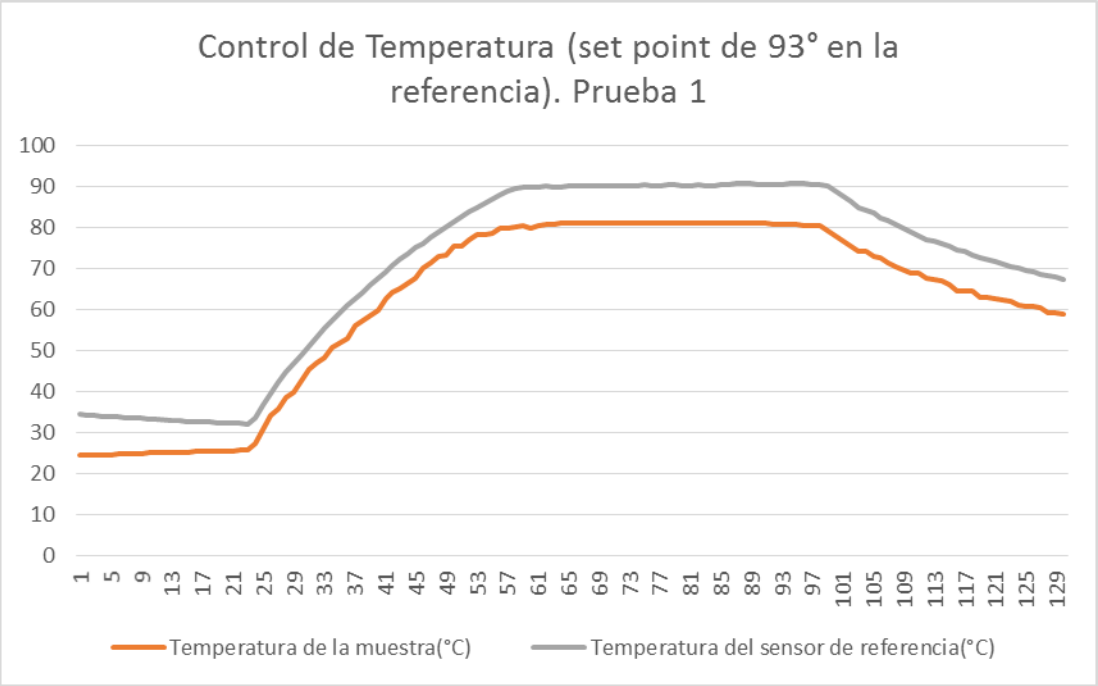


Figura 6.9: Temperatura del sensor de referencia vs temperatura en la fucsina (prueba 1).

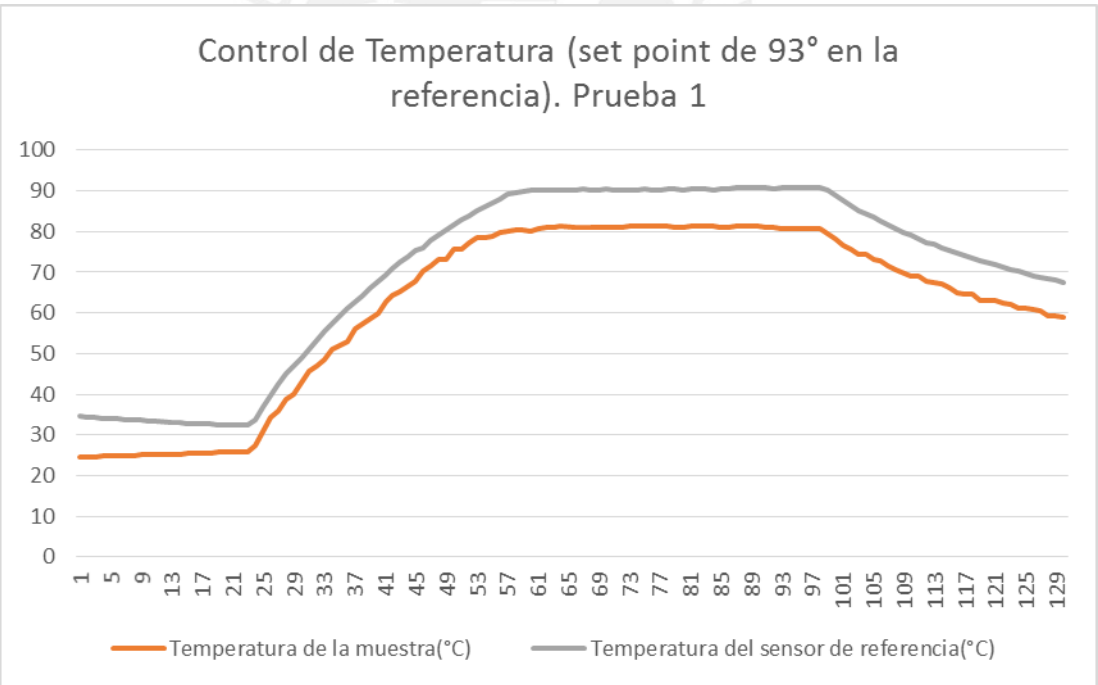


Figura 6.10: Temperatura del sensor de referencia vs temperatura en la fucsina (prueba 2).

6.1.1.1.4 Posiciones de las muestras

Se ha realizado diversos ajustes respecto a la posición del sensor de fin de carrera, en las tablas 6.1 y 6.2 se muestran los de mayor relevancia durante la calibración.

Cambios de dispensado entre el ajuste 1 y 2: El dispensado era excesivo. Se pudo observar el desperdicio de fucsina por lo que se redujo el barrido del dispensado de 3cm a 2.6cm. Además, se agrega un pequeño desplazamiento de 1mm a la posición inicial de dispensado para alejarlo de las pinzas.

Cambios de posición de calentamiento entre el ajuste 3 y 4: se ajusta dos milímetros para que el foco de luz halógeno este alineado con las muestras.

Tabla 6.1: Posiciones del cabezal en milímetros respecto a la ubicación del sensor de fin de carrera.

Ajuste	Posición de calentamiento (mm)	Posición final de dispensado de fucsina (mm)	Posición inicial de dispensado de fucsina (mm)	Posición Inicial para el ingreso de muestras(mm)
1	6	110	140	214
2	6	114	140	214
3	6	113	139	214
4	4	113	139	214

En la tabla 6.2 se presenta la cantidad dispensada en ml de fucsina, para el ajuste 1 y 4. No se coloca el ajuste 5 debido a que en el calentamiento se pierde fucsina, ocasionando que las muestras no queden totalmente recubiertas.

Tabla 6.2: Variación de ml de fucsina respecto al intervalo de dispensado.

Ajuste	Intervalo de dispensado (pasos)	Intervalo de dispensado (mm)	Muestra 1 (ml)	Muestra 2 (ml)	Muestra 3 (ml)	Muestra 4 (ml)	Total (ml)

1	1500	30	3.6	5	3.8	4.6	17
4	1250	25	3	3.55	3	4.5	14.1
5	950	19	2.1	2.75	2.25	3.1	10.2

En la figura 6.11 se muestra el recubrimiento de los portaobjetos luego del calentamiento con el ajuste 4.

Figura 6.11: Recubrimiento de fucsina de las muestras posterior al calentamiento.

6.1.1.1.5 Oxidación de los recipientes

Los recipientes de aluminio presentan capas de óxido tras una semana de almacenaje de fucsina, a pesar de que parece no afectar la fucsina pues el óxido no se dispersa en todo el fluido, existe la posibilidad de que tenga algún efecto perjudicial al ser una oxidación generalizada. Sin el cambio del recipiente (se realizó una limpieza) se pudieron obtener resultados buenos en la coloración. En la figura 6.2 se muestra el recipiente oxidado, también se pudo apreciar oxidación en el recipiente de alcohol. En la figura 6.10 se puede apreciar oxidación del recipiente del alcohol ácido.

Figura 6.12: Oxidación del recipiente de fucsina

6.1.2 Resultados del módulo de microscopia

6.1.2.1 Plataforma móvil

El módulo de microscopio realiza bien el desplazamiento del porta muestras; los desplazamientos de los ejes X e Y permiten el desplazamiento necesario de 2mm para trasladarse entre cada campo de la muestra, mientras que el desplazamiento del eje Z que permite ajustar el enfoque tiene un desplazamiento mínimo de 1/16 de un paso equivalente a 0.0003125mm.

Debido a que para implementar el microscopio del módulo, se adquirió uno comercial y se cortó la parte correspondiente a la óptica, esto redujo la robustez de este módulo y, por lo tanto, las imágenes observadas requieren un algoritmo de mayor complejidad para evitar esta pequeña vibración.

6.1.2.2 Software de autoenfoque

A continuación en las Figuras 6.13 y 6.14 se muestran los resultados de aplicar el algoritmo descrito en el Capítulo 4 para el grupo de datos 1 y 3 de la base de datos de (Shah M. I., 2017). En esta base de datos cada grupo de imágenes corresponde solo a un campo, se define que la imagen mejor enfocada es la décima.

Imagen	Enfoc
1	0.0181
2	0.0855
3	0.1836
4	0.3005
5	0.4096
6	0.5786
7	0.7397
8	0.8586
9	0.8984
10	0.9264
11	1
12	0.9269
13	0.8749
14	0.8308
15	0.6845
16	0.5587
17	0.423
18	0.1739
19	0.0548
20	0

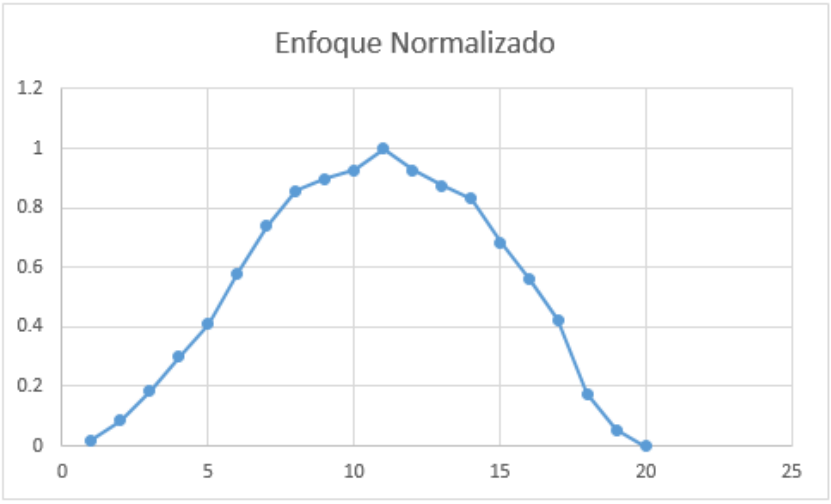


Figura 6.13: Resultados de enfoque normalizado para el primer grupo de imágenes

Imagen	Enfoc
1	0.1054
2	0.2535
3	0
4	0.4676
5	0.0157
6	0.131
7	0.2979
8	0.8091
9	0.9125
10	1
11	0.7546
12	0.9246
13	0.9437
14	0.9573
15	0.72
16	0.5697
17	0.3099
18	0.2101
19	0.4632
20	0.3732

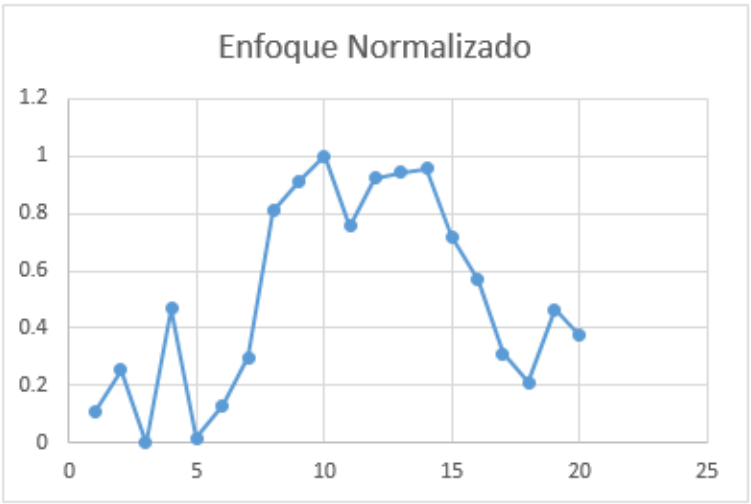


Figura 6.14: Resultados de enfoque normalizado para el segundo grupo de imágenes.

Según se puede apreciar, los resultados pueden llegar a un enfoque promedio de toda la imagen. Sin embargo, dadas las características de las imágenes de microscopio, específicamente el contacto entre la lente y la gota de aceite, provoca que la profundidad de campo sea no uniforme .Es decir, pueden haber objetos que no están enfocados en una imagen, pero si en otra imagen. Esta característica se puede apreciar en la figura 6.15.

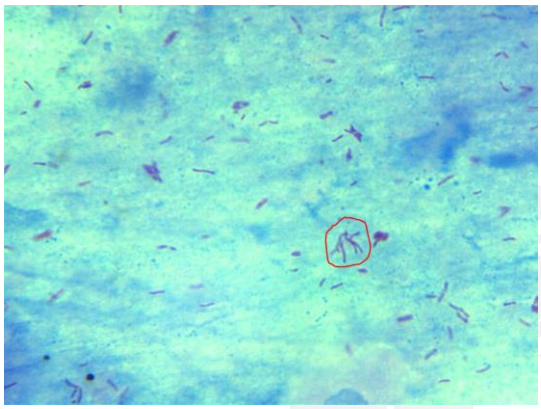


Figura 6.15 a)

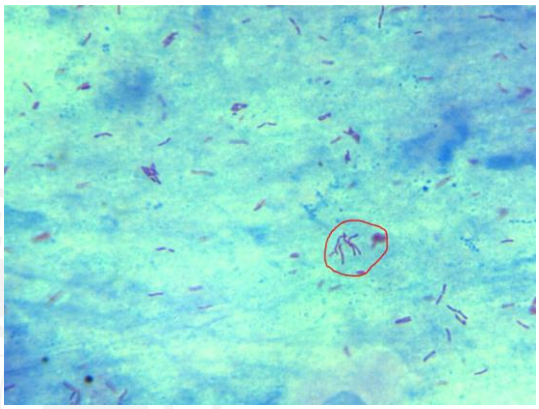


Figura 6.15 b)

Figura 6.15: a) Imagen 10 de la primera secuencia de imágenes, Figura 6.15 b) Imagen 11 de la primera secuencia de imágenes.

6.1.2.3 Software de conteo de bacilos

El software de conteo de bacilos ha sido probado con 3 grupos de muestras positivas tomadas por tecnólogos del Hospital Dos De Mayo, cada grupo contiene 10 sub-grupos de muestras y cada sub-grupo cuenta con 30 imágenes. Se han etiquetado estos 3 juegos de muestras como positivo 1, positivo 2 y positivo 3, donde el tercer grupo corresponde a las muestras que poseen mayor carga bacteriana.

Para el caso de positivo 1 el mínimo valor de bacilos contados es 8 y el máximo 126. En este caso el valor de conteo de 126 corresponde a una muestra que posee una mala tinción, obteniéndose una imagen con bastantes fragmentos del color de los bacilos. En la figura 6.16 se puede

apreciar un campo en donde se detectó 26 bacilos. Sin embargo, hay solo dos bacilos.

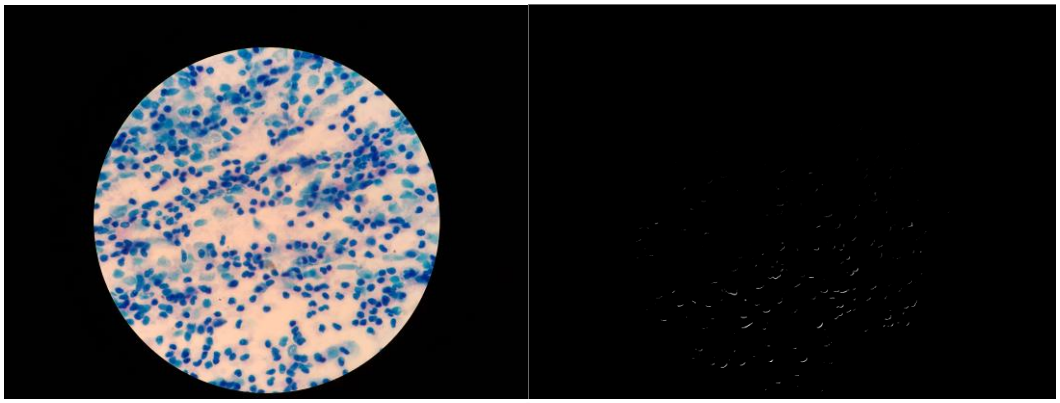


Figura 6.16: Detección incorrecta de bacilos.

En el caso positivo 2 se tiene un conteo mínimo de 6 bacilos y un máximo de 71 bacilos por muestra. Finalmente, en el caso de positivo 3 se tienen un mínimo de 24 y un máximo de 219 bacilos por muestra.

CONCLUSIONES

- **Conclusiones del módulo de tinción:**

Se tiene un equipo que puede teñir 4 muestras en simultaneo de forma automática. Respecto al dispensado de reactivos se concluye que se puede controlar el dispensado de los reactivos variando el tiempo de activación de las bombas y el pequeño desplazamiento de la muestra (barrido). Sin embargo, se ha medido que la cantidad dispensada no es uniforme en todos los dispensadores. En cuanto al control de temperatura, se comprueba que se puede controlar la temperatura en la muestra a partir de un sensor de temperatura de referencia.

La mecánica es muy importante por lo que una sujeción de las muestras sin ángulo de inclinación es crucial para el buen funcionamiento del prototipo. Finalmente, es también importante el material de las estructuras, se ha comprobado que los recipientes metálicos y tuberías neumáticas no son apropiados, ya que se oxidan e hinchan respectivamente. Es también importante la distribución del calor en las muestras, por lo que las pinzas metálicas no son apropiadas ya que estas absorben calor y recalientan parte de la muestra.

- **Conclusiones del módulo de microscopía:**

El equipo realiza los desplazamientos deseados en todos sus ejes, teniendo una mayor sensibilidad en el desplazamiento del eje Z, y por lo tanto funciona correctamente; pero este módulo debe mejorarse. Principalmente se necesita mejorar 2 partes del sistema. En primer lugar el equipo no debe tener vibraciones en las muestras causadas por el accionamiento de los motores o incluso por pequeños golpes en la base que soporta el módulo. Se debe utilizar el microscopio con su base y plataforma de desplazamiento original. Finalmente, es necesario cambiar el mecanismo de acoplamiento de la cámara con el microscopio, pues el acople necesita un continuo ajuste y fijación al ocular del microscopio.

Se implementó una Interfaz de usuario capaz de almacenar los resultados del análisis de microscopía. Además, es posible realizar la búsqueda de los resultados utilizando cualquiera de los campo de búsqueda.

Se implementó un algoritmo de segmentación y conteo de bacilos. Se concluye que se puede usar este algoritmo como un complemento para algoritmos de mayor complejidad, pues la discriminación por color abarca segmentos de la imagen que no son bacilos, pero que tienen una forma muy similar. En conclusión, el algoritmo no es eficiente, actualmente el grupo de investigación está trabajando con otras alternativas.

Se diseñó e implementó un algoritmo de auto-enfoque capaz de encontrar el mejor enfoque promedio para toda una secuencia de imágenes de microscopia.

RECOMENDACIONES Y TRABAJOS FUTUROS

Para trabajos futuros se plantea evaluar el uso de un sensor de temperatura que no requiera contacto. Para ello, se tendrá que evaluar la posibilidad del cambio de la fuente de calor. Además de considerar sensores adicionales de temperatura y humedad tanto del ambiente como del equipo. Esta información podría ser almacenada en una base de datos local o remota de forma automática.

Considerando el sensor de referencia LM35, a fin de poder uniformizar las temperaturas iniciales de cada secuencia se propone evaluar la posibilidad de incluir un precalentamiento de las muestras después de ingresadas las muestras pero antes del dispensado de fucsina.

Finalmente es posible realizar estudios más detallados, es decir realizar un estudio por tipo de muestra, esto permitiría determinar la eficacia del equipo para determinadas muestras: Orina, heces, cefalorraquídeos, pleurales, etc. Para este objetivo, se deberá contar con una base de datos con imágenes agrupadas según el tipo de muestras.

El algoritmo de detección y conteo de bacilos elaborado en la presente tesis podría usarse en combinación con un algoritmo de Deep Learning a fin de obtener mejores resultados.

Para el caso del algoritmo de autoenfoque, se podría obtener un nivel de enfoque para determinados segmentos de la imagen. Es decir, podría obtenerse información del enfoque de bacilos pertenecientes a diferentes áreas y diferentes imágenes.

BIBLIOGRAFÍA

Organization, W. H. (2018). *Global tuberculosis report 2018. Geneva: World Health Organization; 2018. Licence: CC BY-NC-SA 3.0*, de http://www.who.int/tb/publications/global_report/en/

MINSA - I. N. S. *Red de Laboratorios en Salud Pública (RLSP)*. Recuperado el 13 de 11 de 2018, de http://www.portal.ins.gob.pe/es/cnsp/cnsp-c4/red-de-laboratorios-en-salud-publica-rlsp/informacion-general-rlsp

Ministerio de Salud, I. N. S. (2014). *Procedimientos para el control de calidad externo de baciloscopía para el diagnóstico.* Recuperado el 12 de 05 de 2018, de http://www.bvs.ins.gob.pe/insprint/SALUD_PUBLICA/MONO/Control_de_calidad_externo_de_baciloscopia.pdf

Ministerio de Salud de Salud, D. G. (Febrero, 2016). *Análisis de la situación epidemiológica de la tuberculosis en el 2015.* Lima: Ministerio de Salud de Salud. Recuperado el 03 de 09 de 2018, de http://bvs.minsa.gob.pe/local/MINSA/3446.pdf

Raúl Alonso Timana Ruiz, A. S. (2014). *COSTO DE TUBERCULOSIS EN LOS ESTABLECIMIENTOS DE SALUD DEL PERÚ".* Recuperado el 03 de 09 de 2017, de https://www.hfgproject.org/costo-de-tuberculosis-en-los-establecimientos-de-salud-del-peru/

González-Martin. (2017). *Microbiología de la tuberculosis.* Recuperado el 18 de 09 de 2017, de http://www.elsevier.es/es-revista-seminarios-fundacion-espanola-reumatologia-274-articulo-microbiologia-tuberculosis-S1577356614000025

COMINA, G. M. (2011). Development of an automated MODS plate reader to detect early growth of Mycobacterium tuberculosis. *Journal of Microscopy, 242*(10.1111/j.1365-), 325-330.

Ministerio de Salud, I.N.S. (2018). AVANCES EN EL DIAGNÓSTICO Y LA INVESTIGACIÓN DE LA TUBERCULOSIS EN EL PERÚ. *http://www.tuberculosis.minsa.gob.pe* Recuperado el 28 de 04 de 2018,http://www.tuberculosis.minsa.gob.pe/portaldpctb/recursos/20180605214958.pdf

ORGANISMO ANDINO DE SALUD – CONVENIO HIPÓLITO UNANUE, 2018, MANUAL PARA EL DIAGNÓSTICO BACTERIOLÓGICO DE LA TUBERCULOSIS. PARTE 1: MANUAL DE ACTUALIZACIÓN DE LA BACILOSCOPÍA/ Programa “Fortalecimiento de la Red de Laboratorios de Tuberculosis en la Región de las Américas” -- Lima: ORAS - CONHU; 2018.

VALLEJO V, J. C. (2015). Ensayo Xpert MTB/RIF en el diagnóstico de tuberculosis. Revista chilena de enfermedades respiratorias, 127-131. Obtenido de https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0717-73482015000200010

N. Karen R Steingart, ,. P. (s.f.). *Xpert® Mtb/Rif assay for pulmonary tuberculosis and rifampicin resistance in adults.* Obtenido de https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4470349/

Home - Pan American Health Organization". (2017). *Paho.org.* Recuperado el 18 de 09 de 2017, de http://www1.paho.org/hq/dmdocuments/2011/Preguntas_frecuentes_Xper_MTBRIF_final.pdf

I. de Almeida, L. d. (2017). Evaluation of the Mean Cost and Activity Based Cost in the Diagnosis of Pulmonary Tuberculosis in the Laboratory Routine of a High-Complexity Hospital in Brazil. *Frontiers in Microbiology, 8*.

Saba, Y. P. (2013). DISEÑO DE UN SISTEMA AUTOMÁTICO DE PREPARACIÓN DE MUESTRAS DE ESPUTO PARA EL DIAGNÓSTICO DE TBC. PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ.

Tbevidence.org. (2014). *TUBERCULOSIS Diagnostics Technology and Market.* Recuperado el 20 de 09 de 2017, de http://tbevidence.org/wpcontent/uploads/2014/10/UNITAID_TB_Diagnostics_Landscape_3rdedition_

Unitaid. (2017). TUBERCULOSIS Diagnostics Technology Landscape. *Unitaid, 5*, 31-42.

Shah M. I., et al., “Ziehl–Neelsen Sputum smear Microscopy image DataBase (ZNSM-iDB),” 2017, http://14.139.240.55/znsm (27 June 2017). [PMC free article] [PubMed]

Rosebrock, A. (2019). Recognizing digits with OpenCV and Python - PyImageSearch. Retrieved from https://www.pyimagesearch.com/2017/02/13/recognizing-digits-with-opencv-and-python/

Medina, B. (2018). DISEÑO DE UN SISTEMA AUTOMÁTICO DE TINCIÓN DE CUATRO MUESTRAS DE ESPUTO EN SIMULTÁNEO PARA EL DIAGNÓSTICO DE TUBERCULOSIS.

Contours Hierarchy — OpenCV - Python Tutorials 1 documentation. Retrieved from https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contours_hierarchy/py_contours_hierarchy.html

ANEXOS

A. Diagrama de circuito impreso (Módulo de tinción)

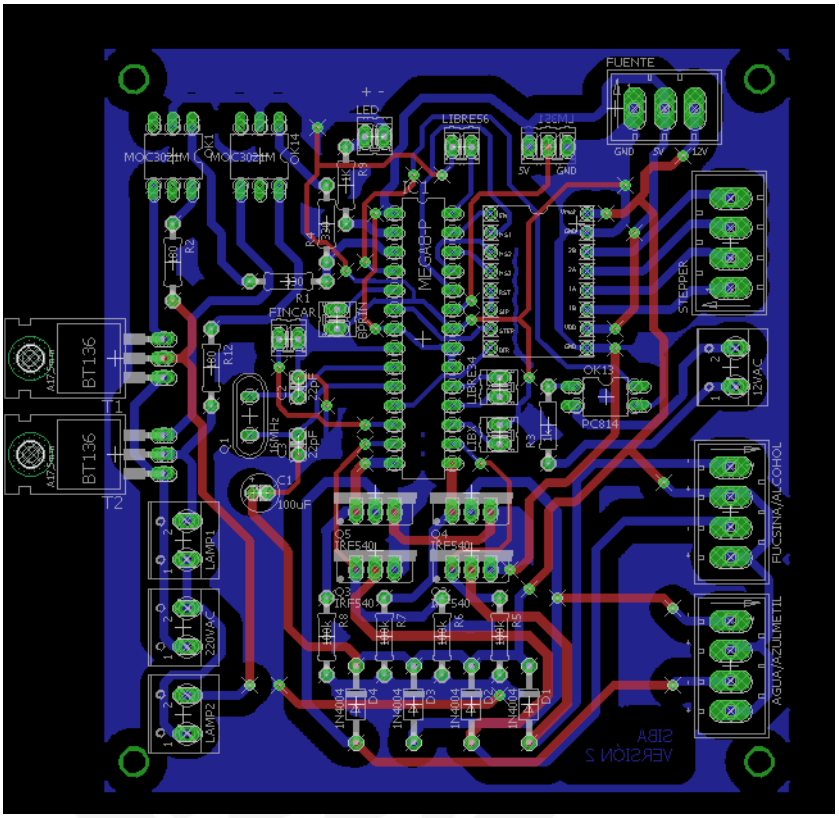


Figura 0.1: Diagrama esquemático.

B. Programa de procesamiento de imagines para la obtención de valores de temperatura

2.

```
from imutils.perspective import four_point_transform
from imutils import contours
import imutils
import cv2
import xlsxwriter

from scipy.misc import toimage

import numpy as np
import matplotlib.pyplot as plt

# define the dictionary of digit segments so we can identify
# each digit on the thermostat
DIGITS_LOOKUP = {
    (1, 1, 1, 0, 1, 1, 1): 0,
```

```
    (0, 0, 1, 0, 0, 1, 0): 1,   # (0, 0, 1, 0, 0, 1, 0)
    (1, 0, 1, 1, 1, 0, 1): 2,   # (1, 0, 1, 1, 1, 1, 0): 2,
    (1, 0, 1, 1, 0, 1, 1): 3,
    (0, 1, 1, 1, 0, 1, 0): 4,
    (1, 1, 0, 1, 0, 1, 1): 5,
    (1, 1, 0, 1, 1, 1, 1): 6,
    (1, 0, 1, 0, 0, 1, 0): 7,
    (1, 1, 1, 1, 1, 1, 1): 8,
    (1, 1, 1, 1, 0, 1, 1): 9
}

workbook =
xlsxwriter.Workbook(r'C:\Users\User\Desktop\Electronica\Joaquin\Tesi
s\Pruebas29052019\Visualizacion Validacion
Hospital\PruebaTemperaturaValidacion2.xlsx')
worksheet = workbook.add_worksheet()
row = 0
col = 0

for a in range(0,236):#633
    try:
        # load the example image
        #string_nombre = "scene00001"
        #nombre_int = int(string_nombre.split("scene")[1])
        #string_new = "scene" + str(nombre_int + 24).zfill(5) +
".jpg"
        string_new = "framefinal" + str(a) + ".jpg"
        string_old =
"C:\\Users\\User\\Desktop\\Electronica\\Joaquin\\Tesis\\Pruebas29052
019\\Visualizacion Validacion Hospital\\"

        image = cv2.imread(string_old+string_new)  #
r"C:\Users\User\Downloads\recognize-digits\recognize-
digits\frame1002.jpg")#Ejemplo25

        # pre-process the image by resizing it, converting it to
        # graycale, blurring it, and computing an edge map
        y1=273
        y2=344
        x1=519
        x2=684
        image = imutils.resize(image, height=500)
        image = image[y1:y2, x1:x2]
        cv2.imshow("I", image)
        #cv2.waitKey(0)

        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        blurred = cv2.GaussianBlur(gray, (5, 5), 0)  # 5 5

        blurred = cv2.bitwise_not(gray)
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (2,
2))
        blurred= cv2.dilate(blurred, kernel, iterations=1)
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (1,
1))
        blurred = cv2.erode(blurred, kernel, iterations=1)

        gray = cv2.bitwise_not(blurred)

        cv2.imshow("gray", gray)
        #cv2.waitKey(0)
```

```
        edged = cv2.Canny(blurred, 50, 100, 255)

        cv2.imshow("Edged", edged)
        #cv2.waitKey(0) #test final

        # find contours in the edge map, then sort them by their
        # size in descending order
        cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
                                cv2.CHAIN_APPROX_SIMPLE)
        cnts = cnts[0] if imutils.is_cv2() else cnts[1]
        cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
        print(cnts)
        displayCnt = None

        # loop over the contours
        for c in cnts:
            # approximate the contour
            peri = cv2.arcLength(c, True)
            approx = cv2.approxPolyDP(c, 0.02 * peri, True)

            # if the contour has four vertices, then we have found
            # the thermostat display
            if len(approx) == 4:
                displayCnt = approx
                break

        print(displayCnt)

        #####Evitar canny
        displayCnt = np.array([(0, 0), (x2-x1-1, 0), (0, y2-y1-1),
(x2-x1-1, y2-y1-1)], dtype="float32")  # (1049, 446)

        #######
        print(displayCnt)

        # extract the thermostat display, apply a perspective
transform
        # to it
        warped = four_point_transform(gray, displayCnt.reshape(4,
2))
        output = four_point_transform(image, displayCnt.reshape(4,
2))

        cv2.imshow("Waldo", warped)
        #cv2.waitKey(0)# test final

        # threshold the warped image, then apply a series of
morphological
        # operations to cleanup the thresholded image
        thresh = cv2.threshold(warped, 0, 255,  # 255
                               9)[1]

        print(cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)

        #####Usando threshold adaptativo
        # plt.hist(warped.ravel(),256,[0,256]); plt.show()

        # thresh = cv2.threshold(warped, 0, 255, cv2.THRESH_OTSU)[1]
        # thresh = cv2.threshold(warped, 0, 255,#255
        #  cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
```

```
        ##################################

        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (1,
5))  # (1,5)

        cv2.imshow("Thresh", thresh)
        #cv2.waitKey(0) #test final

        #thresh = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
#problemas con el kernel

        # find contours in the thresholded image, then initialize
the
        # digit contours lists
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
                                cv2.CHAIN_APPROX_SIMPLE)

        cnts = cnts[0] if imutils.is_cv2() else cnts[1]
        print("holaMundo")
        print(cnts)
        digitCnts = []

        cv2.imshow("Thresh", thresh)
        #cv2.waitKey(0) #test final

        # loop over the digit area candidates
        for c in cnts:
            # compute the bounding box of the contour
            (x, y, w, h) = cv2.boundingRect(c)
            print(x, y, w, h)
            # if the contour is sufficiently large, it must be a
digit
            if w >= 3 and (h >= 15 and h <= 60):  #
                digitCnts.append(c)

        # sort the contours from left-to-right, then initialize the
        # actual digits themselves

        print(digitCnts)

        digitCnts = contours.sort_contours(digitCnts,
                                           method="left-to-
right")[0]
        digits = []

        # loop over each of the digits
        for c in digitCnts:
            # extract the digit ROI
            (x, y, w, h) = cv2.boundingRect(c)
            if w<=10:
                w=15
                #x=x-10
            roi = thresh[y:y + h, x:x + w]
            # compute the width and height of each of the 7 segments
            # we are going to examine
            (roiH, roiW) = roi.shape
            (dW, dH) = (int(roiW * 0.25), int(roiH * 0.15))
            dHC = int(roiH * 0.05)

            # define the set of 7 segments
```

```
            segments = [
                ((0, 0), (w, dH)),  # top
                ((0, 0), (dW, h // 2)),  # top-left
                ((w - dW, 0), (w, h // 2)),  # top-right
                ((0, (h // 2) - dHC), (w, (h // 2) + dHC)),# center
                ((0, h // 2), (dW, h)),  # bottom-left
                ((w - dW, h // 2), (w, h)),  # bottom-right
                ((0, h - dH), (w, h))  # bottom
            ]

            on = [0] * len(segments)

            # loop over the segments
            for (i, ((xA, yA), (xB, yB))) in enumerate(segments):
                # extract the segment ROI, count the total number of
                # thresholded pixels in the segment, and compute
                # the area of the segment
                segROI = roi[yA:yB, xA:xB]
                total = cv2.countNonZero(segROI)
                area = (xB - xA) * (yB - yA)

                # if the total number of non-zero pixels is greater
than
                # 43% of the area, mark the segment as "on"
                if total / float(area) > 0.43:  # 0.5
                    on[i] = 1

            print(on)
            cv2.rectangle(output, (x, y), (x + w, y + h), (0, 255,
0), 1)
            cv2.imshow("Output", output)
            #cv2.waitKey(0) #test final
            # lookup the digit and draw it on the image
            digit = DIGITS_LOOKUP[tuple(on)]
            print("hola")
            print(digit)
            digits.append(digit)
            cv2.rectangle(output, (x, y), (x + w, y + h), (0, 255,
0), 1)
            cv2.putText(output, str(digit), (x - 4, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255, 0),
1)

            # display the digits
        print(a)
        print(digits)
        print(u"{}{}{}.{} \u00b0C".format(*digits))
        worksheet.write(row + a, col, u"{}{}{}.{}".format(*digits))
        cv2.imshow("Input", image)
        cv2.imshow("Output", output)
        #cv2.waitKey(0) #test final
    except:
        worksheet.write(row+a, col, "0")
        pass

workbook.close()
```

C. Programa para Arduino(Módulo de Tinción)

```
//atmega328p-pu
//Pines E/S Mini-bombas
#define fucsina 7 //7
#define agua 8
#define azul_metileno 9 //9
#define alcohol_acido 10 //10
int tbombeo = 50;
//Pines Control de Temperatura
#define cruce_por_cero 2
#define disparo 1 //3
#define LM35 A0

//pulsador
#define p_arranque 3 //12
//#define p_parada 9

//motor a pasos
#define pasos 4
#define direccion 5
#define fin_carrera 6
#define ms1 A1
#define ms2 A2
#define ms3 A3

const int ledPin =  0;// the number of the LED pin
```

```
// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;        // will store last time LED was updated

// constants won't change:
const long interval1 = 1*1000;           // interval at which to blink (milliseconds)
const long interval2 = 2*1000;
////////////////////////////7777


//temperatura
float temperatura_deseada = 70.0;//50
int angulo_maximo = 6500;
int angulo_minimo = 1000;
int error_minimo = 1;//0;
int error_maximo = 5;
int flag=0;
float tiempo_exposicion=0;
float m = (angulo_maximo - angulo_minimo) / (error_minimo - error_maximo);
float b = angulo_minimo - m * error_maximo;
float esc, temperatura_actual, error_temperatura;
```

```
int medicion;
int angulo_disparo = 1000;
int giro;

int primera=1;

//posiciones

int posicion_insercion = 10700; //9900  + 800
//insercion 13000 a 13500;

//dispensado 7800 a 11500;
int posicion_fucsina = 3900;

int posicion_calentamiento = 2300; //450;
//calentamiento 1400 a 4200;

//2300 + 2400/2
int posicion_reactivo = 800; //450; a la zona de reactivo

//3500+800= 4300
int posicion_enjuague1 = 4300; //para desplazarse a la zona de dispensacion
int posicion_enjuague2 = 2000;

int posicion_microscopio =9000; //9000
```

```
int t = 500;
int SP_posicion, PV_posicion, error_posicion, num_pasos,dir;

int reactivo = 11;
char dato = '0';
long tiempo_anterior;

void setup() {

  pinMode(disparo, OUTPUT);
  digitalWrite(disparo, LOW);//Prueba 15/01/2018

//  Serial.begin(9600);
//  pinMode(giro1, OUTPUT);
//  pinMode(giro2, OUTPUT);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);

  pinMode(fucsina, OUTPUT);
  pinMode(agua, OUTPUT);
  pinMode(azul_metileno, OUTPUT);
  pinMode(alcohol_acido, OUTPUT);

  pinMode(pasos, OUTPUT);
  digitalWrite(pasos, LOW);//Prueba 15/01/2018
```

```
  pinMode(direccion, OUTPUT);
  pinMode(ms1, OUTPUT);
  pinMode(ms2, OUTPUT);
  pinMode(ms3, OUTPUT);
  //8vo paso
  digitalWrite(ms1, HIGH);//HIGH
  digitalWrite(ms2, HIGH);//HIGH
  digitalWrite(ms3, LOW);

  pinMode(p_arranque, INPUT);
  digitalWrite(p_arranque, HIGH);

  pinMode(fin_carrera, INPUT);
  digitalWrite(fin_carrera, HIGH);

  pinMode(cruce_por_cero, INPUT);
  digitalWrite(cruce_por_cero, HIGH);

  //analogReference(EXTERNAL);
  PV_posicion = 0.00;
}

void insercion_de_muestras(){
  //SP_posicion = posicion_insercion;
  num_pasos = SP_posicion;
  digitalWrite(direccion, dir);
```

```
  for(int i = 0; i <= num_pasos; i++){
      digitalWrite(pasos, HIGH);
      delayMicroseconds(t);
      digitalWrite(pasos, LOW);
      delayMicroseconds(t);
  }
}

void referencia(){
  digitalWrite(direccion, LOW);
  while(digitalRead(fin_carrera) == LOW){
      digitalWrite(pasos, HIGH);
      delayMicroseconds(t);
      digitalWrite(pasos, LOW);
      delayMicroseconds(t);
  }
}

void secado() {
  tiempo_anterior = millis();
  while(millis() - tiempo_anterior <= tiempo_exposicion){//60000
    medicion = analogRead(LM35);
    temperatura_actual = 5.0*medicion*100.0/1023;//1024.0;
    //temperatura_actual = esc/10.0;
//    Serial.print("Temperatura: ");
//    Serial.println(temperatura_actual);
```

```
//    delay(250);
    error_temperatura = temperatura_deseada -
temperatura_actual;
//control on/off
//    angulo_disparo = 4000;


  if (primera==1){
       if (temperatura_deseada < temperatura_actual){
         flag=0;
       }
       else {
         flag=1;
         }
       }
   else{

//control proporcional


      if (error_temperatura >= error_maximo){
        angulo_disparo = angulo_minimo;
        flag=1;
      }
      else if (error_temperatura < error_maximo &&
error_temperatura > error_minimo) {
        angulo_disparo = m * error_temperatura + b;
        flag=0;
      }
```

```
      else{
        angulo_disparo = angulo_maximo;
        flag=0;

      }
     }
   }
  detachInterrupt(0);
}

void dispensado_de_reactivo(){
  //delay(400);
  digitalWrite(reactivo, HIGH);
  //delay(100);//1500
  //SP_posicion = 2000;
  digitalWrite(direccion, dir);;//digitalWrite(direccion, LOW);
  desplazar_a_posicion();
  //delay(300);//1500
  digitalWrite(reactivo, LOW);
}

void dispensado_de_reactivo2(){///////AGUA
  delay(500);
  digitalWrite(reactivo, HIGH);
  delay(3000);///500
  SP_posicion = 2000;
```

```
  digitalWrite(direccion, LOW);
  desplazar_a_posicion();
  digitalWrite(reactivo, LOW);
}

//void dispensado_fucsina(){
//  reactivo = fucsina;
//  dispensado_de_reactivo();
//}

void enjuague(){
  delay(1000);
  digitalWrite(reactivo, HIGH);
  delay(25000);//8000);//antes estaba 2500
  digitalWrite(reactivo, LOW);
  delay(500);//agregado
}

//void dispensado_alcohol(){
//  reactivo = alcohol_acido;
//  dispensado_de_reactivo();
//}
//
//void dispensado_azul_metileno(){
//  reactivo = azul_metileno;
//  dispensado_de_reactivo();
```

```
//}


void desplazar_a_posicion(){
  digitalWrite(direccion, dir);
  num_pasos = SP_posicion;
  for(int i = 0; i <= num_pasos; i++){
      digitalWrite(pasos, HIGH);
      delayMicroseconds(t);
      digitalWrite(pasos, LOW);
      delayMicroseconds(t);
  }
//  delay(1000);
}


void microscopio(){
  SP_posicion = posicion_microscopio;
  num_pasos = SP_posicion;
  digitalWrite(direccion, HIGH);
  for(int i = 0; i <= num_pasos; i++){
      digitalWrite(pasos, HIGH);
      delayMicroseconds(t);
      digitalWrite(pasos, LOW);
      delayMicroseconds(t);
  }
//  Serial.println("Analisis microscopico");
}
```

```
void loop() {
//  attachInterrupt(0, Shotter, FALLING);
//  secado();
//  while(1){}
  referencia();
  delay(1000);

  if(primera==1){
    temperatura_deseada = 60.0;
    attachInterrupt(0, Shotter, FALLING);
    tiempo_exposicion=15000;
    secado();
    delay(1000);
    primera=0;
  }

  SP_posicion = posicion_insercion;//10700
  dir=1;
  insercion_de_muestras();

  Serial.println("Presione pulsador de arranque luego de insertar las muestras");
  while(digitalRead(p_arranque)==HIGH){
        unsigned long currentMillis = millis();
    if (currentMillis - previousMillis < interval2) {
```

```
    if(currentMillis - previousMillis >= interval1) {
        digitalWrite(ledPin, LOW);
        //if(cont==0){
          //Imprimir en LCD o otra cosa que demore menos del
tiempo del pin en baja
          //cont=1;
        //}
       }
   }else{
    digitalWrite(ledPin, HIGH);
    previousMillis = currentMillis;
    //cont=0;
    }
  }
 digitalWrite(ledPin, HIGH);
 delay(1000);

 SP_posicion = 3700+50;//4100
 dir=0;  //digitalWrite(direccion, LOW);
 desplazar_a_posicion();

 reactivo = fucsina;
 SP_posicion = 1300;//2000;
 dir=0;
 dispensado_de_reactivo();
 delay(3000);//2000
```

```
SP_posicion = 4700-50+700+100;//3400;//4200//4700+500
dir=0;
desplazar_a_posicion();
temperatura_deseada=93;//105
tiempo_exposicion=75000;//120000
attachInterrupt(0, Shotter, FALLING);
secado();

referencia();
delay(1000);
SP_posicion = posicion_insercion;//10700
dir=1;
insercion_de_muestras();
delay(300000);

SP_posicion = 3700+4700+1100+900;//3700+4700+2000
dir=0;  //digitalWrite(direccion, LOW);
desplazar_a_posicion();

SP_posicion = 2400;//1800
dir=1;//digitalWrite(direccion, HIGH);
desplazar_a_posicion();
reactivo = agua;
enjuague();
delay(3000);
```

```
  digitalWrite(ms1, LOW);//LOW 1 PASO
  digitalWrite(ms2, LOW);
  digitalWrite(ms3, LOW);

  SP_posicion = 400;//200
  for(int i=0; i<3; i++) {
    dir=1;
    desplazar_a_posicion();
    delay(75);
    dir=0;
    desplazar_a_posicion();
    delay(75);
  }
  digitalWrite(ms1, HIGH);//HIGH
  digitalWrite(ms2, HIGH);//HIGH
  digitalWrite(ms3, LOW);
  delay(2000);

 ///SETEO AGUA1
  referencia();
  SP_posicion = 2400+400;
  dir=1;//digitalWrite(direccion, HIGH);
  desplazar_a_posicion();
 ////////////
```

```
SP_posicion =3350; //2550;//2950
dir=1;//digitalWrite(direccion, HIGH);
desplazar_a_posicion();

reactivo = alcohol_acido;
SP_posicion =950;//2000;
dir=0;
dispensado_de_reactivo();
delay(120000);//00);

SP_posicion = 1350+1050;//950;
dir=0;//;digitalWrite(direccion, LOW);
desplazar_a_posicion();
reactivo = agua;
enjuague();
delay(3000);

digitalWrite(ms1, LOW);//LOW
digitalWrite(ms2, LOW);//
digitalWrite(ms3, LOW);
SP_posicion = 400;//200
for(int i=0; i<3; i++) {
  dir=1;
  desplazar_a_posicion();
  delay(75);
```

```
    dir=0;//digitalWrite(direccion, LOW);
    desplazar_a_posicion();
    delay(75);
  }
  digitalWrite(ms1, HIGH);
  digitalWrite(ms2, HIGH);
  digitalWrite(ms3, LOW);
  delay(2000);

 ///SETEO AGUA2
  referencia();
  SP_posicion = 2400+400;
  dir=1;//digitalWrite(direccion, HIGH);
  desplazar_a_posicion();
 ////////////


  SP_posicion = 2200;//1900;ajustado para prueba20/07/2017
  dir=1;
  desplazar_a_posicion();

  SP_posicion = 950;//2000;
  dir=0;
  reactivo =azul_metileno;
  dispensado_de_reactivo();
  delay(60000);//180000);
```

```
//  SP_posicion = 0;//100;//1900;ajustado para prueba20/07/2017
//  digitalWrite(direccion, HIGH);
  SP_posicion = 200+1050;//ajustado para prueba 11/08/2017
  dir=0;
  desplazar_a_posicion();
  reactivo = agua;
  enjuague();
  delay(3000);

  digitalWrite(ms1, LOW);//LOW
  digitalWrite(ms2, LOW);
  digitalWrite(ms3, LOW);
  SP_posicion = 400;//200
  for(int i=0; i<3; i++) {
    dir=1;
    desplazar_a_posicion();
    delay(75);
    dir=0;
    desplazar_a_posicion();
    delay(75);
  }
  digitalWrite(ms1, HIGH);//HIGH
  digitalWrite(ms2, HIGH);//HIGH
  digitalWrite(ms3, LOW);
  delay(2000);
```

```
//secado
referencia();

SP_posicion = 300;//4100
dir=1;  //digitalWrite(direccion, LOW);
desplazar_a_posicion();

temperatura_deseada = 70.0;
attachInterrupt(0, Shotter, FALLING);
tiempo_exposicion=60000;
secado();

digitalWrite(ms1, LOW);//LOW
digitalWrite(ms2, LOW);
digitalWrite(ms3, LOW);
SP_posicion = 400;//200
for(int i=0; i<3; i++) {
  dir=1;
  desplazar_a_posicion();
  delay(75);
  dir=0;
  desplazar_a_posicion();
  delay(75);
}
digitalWrite(ms1, HIGH);//HIGH
```

```
  digitalWrite(ms2, HIGH);//HIGH
  digitalWrite(ms3, LOW);
  delay(2000);

  referencia();

  SP_posicion = 300;//4100
  dir=1;  //digitalWrite(direccion, LOW);
  desplazar_a_posicion();

  temperatura_deseada = 70.0;
  attachInterrupt(0, Shotter, FALLING);
  tiempo_exposicion=60000;
  secado();

///SETEO AGUA3
  //referencia();
  //SECAR
  //SP_posicion = 600;
  //digitalWrite(direccion, HIGH);
  //desplazar_a_posicion();
  //attachInterrupt(0, Shotter, FALLING);//cuando deshabilitas la
interrupcion???
  //secado();
  //delay(500);
  //SECAR
```

```
//  microscopio();//Comentado Prueba 15/01/2018
//  while(digitalRead(p_arranque)==HIGH){} //Comentado Prueba
15/01/2018



//  delay(3000);
//  insercion_de_muestras();
}

//interrupcion de cruce por cero
void Shotter() {
  //delayMicroseconds(angulo_disparo);
  if(flag==1){
  delayMicroseconds(1000);
  digitalWrite(disparo, HIGH);
  delayMicroseconds(10);
  digitalWrite(disparo, LOW);
  }else{
    }


}
```

D. Programa de autoenfoque diseñado en Matlab.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread(r'C:\Users\User\Desktop\Autofocus_Microscope-
1_set1\100.jpg')
img2 = cv2.imread(r'C:\Users\User\Desktop\Autofocus_Microscope-
```

```
1_set1\01.jpg')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

#filterHP=gray-gray2

rows, cols = gray.shape
fil,col=int(rows/2),int(cols/2)
print(col)
##########3
##Difuminado

# blur = cv2.GaussianBlur(img,(35,35),100)
#https://www.aprenderpython.net/suavizando-imagenes-con-opencv/
#
# f = np.fft.fft2(blur)
# imT2= 20*np.log(np.abs(np.fft.fftshift(f)))
# imf2 =imT2
# imf4=imf2[fil][0:499]

############
#plt.subplot(121),plt.imshow(imf2, cmap = 'gray')
#plt.title('Input Image'), plt.xticks([]), plt.yticks([])

# while(True):
#
#     cv2.namedWindow("Original", cv2.WINDOW_NORMAL);
#     cv2.resizeWindow('jpg', rows, cols)
#     cv2.imshow('Original',img)
#     cv2.namedWindow("Difuminado", cv2.WINDOW_NORMAL);
#     cv2.imshow('Difuminado',blur)
#     if cv2.waitKey(1) & 0xFF == ord('q'):
#         break
#########################3

# Espectro de magnitud para una fila con python
# f = np.fft.fft2(gray)
# imT2= np.fft.fftshift(f)
# imf2 =20*np.log(np.abs(imT2))
# imf3=imf2[fil][0:499]
# Espectro de magnitud con python

dft = cv2.dft(np.float32(gray),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum =
20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

f = np.fft.fft2(gray)
imT2= 20*np.log(np.abs(np.fft.fftshift(f)))

#plt.subplot(121),plt.imshow(imf2, cmap = 'gray')
#plt.title('Input Image'), plt.xticks([]), plt.yticks([])
f_HP = np.fft.fft2(gray2)
imT2_HP= 20*np.log(np.abs(np.fft.fftshift(f_HP)))

resta=imT2-imT2_HP
#plt.plot(resta)
#plt.show()
row=np.mean(resta, axis=1)
print('row')
```

```
print(np.mean(row))
######################
```

E. Programa de detección y conteo implementado en Python.

```
# Implemetación en python del programa diseñado en MATLAB por Miguel
Cataño y Maria Helguera: Prog_final
# Implementación Joaquin GV.
# Realizar validación y comparación

import numpy as np
import cv2
from skimage.morphology import skeletonize as skl
from imutils.contours import sort_contours
from bwmorph_thin import bwmorph_thin
import math
import imutils
import xlsxwriter
import xlrd


def skeletonize(img):
    """ OpenCV function to return a skeletonized version of img, a
Mat object"""

    #  hat tip to http://felix.abecassis.me/2011/09/opencv-
morphological-skeleton/

    img = img.copy() # don't clobber original
    skel = img.copy()

    skel[:,:] = 0
    kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3,3))

    while True:
        eroded = cv2.morphologyEx(img, cv2.MORPH_ERODE, kernel)
        temp = cv2.morphologyEx(eroded, cv2.MORPH_DILATE, kernel)
        temp  = cv2.subtract(img, temp)
        skel = cv2.bitwise_or(skel, temp)
        img[:,:] = eroded[:,:]
        if cv2.countNonZero(img) == 0:
            break

    return skel

def skeleton_endpoints(skel):
    # Function source: https://stackoverflow.com/questions/26537313/
    # how-can-i-find-endpoints-of-binary-skeleton-image-in-opencv
    # make out input nice, possibly necessary
    skel = skel.copy()
    skel[skel != 0] = 1
    skel = np.uint8(skel)

    # apply the convolution
    kernel = np.uint8([[1,  1, 1],
                       [1, 10, 1],
                       [1,  1, 1]])
    src_depth = -1
    filtered = cv2.filter2D(skel, src_depth,kernel)

    # now look through to find the value of 11
```

```
    # this returns a mask of the endpoints, but if you just want the
    # coordinates, you could simply return np.where(filtered==11)
    out = np.zeros_like(skel)
    out[np.where(filtered == 11)] = 1
    rows, cols = np.where(filtered == 11)
    coords = list(zip(cols, rows))
    return coords

in_wb = xlrd.open_workbook(r'C:\Users\User\Downloads\30tomas-
20180903T185836Z-001\30tomas\muestras_f.xlsx')
in_sheet = in_wb.sheet_by_name("muestra02")

out_wb = xlsxwriter.Workbook(r'C:\Users\User\Downloads\30tomas-
20180903T185836Z-001\30tomas\muestras_f_test.xlsx')

for col in range(0,3):
    ws1 = out_wb.add_worksheet("pos"+str(col))
    print("pos"+str(col))
    for fil in range(0,10):#10
        print("fila"+str(fil))
        for a in range(1,31):#31
            try:
                sh1=int(in_sheet.cell(2+fil, 1+col).value)
                string_old = "C:\\Users\\User\\Downloads\\30tomas-
20180903T185836Z-001\\30tomas\\"
                string_new = "pos"+str(col+1)+"\\"+str(sh1) + "\\"+
str(sh1)+"-" + str(a) + ".jpg"
                string_cambio=string_old+string_new
                ##print(string_cambio)
                frame = cv2.imread(string_cambio)
                #frame = imutils.resize(frame, height=480)#Agregado
para cambiar el tamaño de la imagen
                frame_prima = imutils.resize(frame, height=480)
                ##cv2.imshow('L', frame_prima)

                im2 = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)
                imax = im2[:, :, 1]  # a
                ima = (im2[:, :, 1] / 255) * 220 - 110  # a
                # imay = im2[:, :, 2]  # a
                imb = (im2[:, :, 2] / 255) * 220 - 110  # b
                iml = im2[:, :, 0]  # L

                im2 = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)
                imax = im2[:, :, 1]  # a
                ima = (im2[:, :, 1] / 255) * 220 - 110  # a
                # imay = im2[:, :, 2]  # a
                imb = (im2[:, :, 2] / 255) * 220 - 110  # b
                iml = im2[:, :, 0]  # L

                an = np.deg2rad(110)  # Se rota an grados el plano a
- b
                ima2 = ima * np.cos(an) - imb * np.sin(an)
                # ima2 = ((ima * np.cos(an) - imb *
np.sin(an))/220)*255 +110
                # ima3 = ima2.astype(np.uint8)
                imb2 = imb * np.cos(an) + ima * np.sin(an)
                # imb2 = ((imb * np.cos(an) + ima * np.sin(an)) /
220) * 255 + 110
                # imb3 = imb2.astype(np.uint8)

                imb2p = imb2 >= 0  # No se toma en cuenta los b
```

```
resultantes luego del giro
                des1 = 20  # Desviación Std de sigmoide
                des2 = 20  # Desviación Std de la gaussiana
                k1 = np.exp(-ima2 ** 2 / des2 ** 2) * (1 - np.exp(-
imb2 ** 2 / des1 ** 2))
                # k2 = exp(-imb2. ^ 2 / des1 ^ 2) - 1;
                k1 = k1 * imb2p  # Resalta los bacilos % k2 = k2. *
~imb2p;
                # k1 = k1*255
                # k1 = k1.astype(np.uint8)
                imy2 = (k1 > 0.5)  # *255
                # imy2 = imy2.astype(np.uint8)

                imy2p = (k1 > 0.5) * 255
                imy2p = imy2p.astype(np.uint8)

                bw2 = skl(imy2)  # skeletonize(imy2)
                bw2 = bw2.astype(np.uint8) * 255
                bw3 = skeleton_endpoints(bw2)
                bw3p = np.zeros_like(bw2)
                [cv2.circle(bw3p, ep, 0, 1, 0) for ep in bw3]  #
[cv2.circle(bw3p, ep, 0, 255, 1) for ep in bw3]

                [num, L] = cv2.connectedComponents(imy2p, 8)  #
Etiquetado de segmentos
                L1 = L
                # a=1-bw3p
                # L=1-L*a  # Ubicacion de los puntos extremos
                L = L * bw3p
                # cv2.imshow('a', a*255)

                # label_hue = np.uint8(179 * L/ np.max(L))
                # blank_ch = 255 * np.ones_like(label_hue)
                # labeled_img = cv2.merge([label_hue, blank_ch,
blank_ch])
                # # cvt to BGR for display
                # labeled_img = cv2.cvtColor(labeled_img,
cv2.COLOR_HSV2BGR)
                # # set bg label to black
                # labeled_img[label_hue == 0] = 0
                # cv2.imshow('L', labeled_img)


                for k in range(1, num + 1):  # Se retiran a aquellas
figuras que tienen mas de 2 puntos extremos
                    s1 = sum(sum(L == k))
                    if s1 > 3 or s1 == 0:
                        L1[L1 == k] = 0

                imy3 = (L1 > 0) * 255  # Umbralizado
                imy3 = imy3.astype(np.uint8)
                # cv2.imshow('y3', imy3)
                [num, L2] = cv2.connectedComponents(imy3, 8)  #
Etiquetado de lo segmentos con dos puntos extremos

                label_hue = np.uint8(179 * L2 / np.max(L2))
                blank_ch = 255 * np.ones_like(label_hue)
                labeled_img = cv2.merge([label_hue, blank_ch,
blank_ch])
                # cvt to BGR for display
                labeled_img = cv2.cvtColor(labeled_img,
```

```
cv2.COLOR_HSV2BGR)
                # set bg label to black
                labeled_img[label_hue == 0] = 0
                # cv2.imshow('L2', labeled_img)

                # L3 = bwmorph(imy3, 'shrink', Inf)

                L3 = imy3.copy()
                LX, contours, hierarchy = cv2.findContours(imy3,
cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

                # Prueba de eliminar agujeros
                # im =
cv2.imread(r'C:\Users\User\Desktop\tree_hierarchy.png')
                # imCopy = im.copy()
                # imgray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
                # ret, thresh = cv2.threshold(imgray, 127, 255, 0)
                # image, contours, hierarchy =
cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
                #
                # for i in range(0, len(contours)): # iterate
through each contour.
                #     if (hierarchy[0,i,3] == -1) &
(hierarchy[0,i,2] > 0) : #if hierarchy(0,i,2)
                #         cv2.drawContours(imCopy, contours, i, (0,
0, 0),-1)
                # Eliminando agujeros prueba

                for i in range(0, len(contours)):  # iterar sobre
cada contorno, para eliminar agujeros.
                # Verificar para todos lo contornos padres (-1) y si
tienen algun contorno hijo rellenarlo(hacerlo background)
                    if (hierarchy[0, i, 3] == -1) & (hierarchy[0, i,
2] > 0):  # if hierarchy(0,i,2)
                        cv2.drawContours(L3, contours, i, (0, 0, 0),
-1)

                L3prima = imutils.resize(L3, height=480)
                ##cv2.imshow('L3', L3prima)
                [num, L] = cv2.connectedComponents(L3, 8)  #
Etiquetando los que no tienen agujeros

                nf = num

                L3 = L3 / 255
                L3 = L3.astype(np.uint8)
                bw2 = skl(L3)
                bw2_prima= imutils.resize(bw2/1, height=480)
                ##cv2.imshow('bw2', bw2_prima)

                L2 = L * bw2/1
                L2_prima= imutils.resize(L/255, height=480)
                ##cv2.imshow('l2prima', L2_prima)

                if num > 1:
                    for k in range(1, num + 1):
                        #print(sum(sum(L2 == k)))
                        if sum(sum(L2 == k)) > 0:
                            if sum(sum(L == k)) / sum(sum(L2 == k))
> 10 or sum(sum(L == k)) < 150:#100:
                                L[L == k] = 0
```

```
                                nf = nf - 1
                        else:
                            if sum(sum(L == k)) < 150:#100:
                                L[L == k] = 0
                                nf = nf - 1

                    label_hue = np.uint8(179 * L / np.max(L))
                    blank_ch = 255 * np.ones_like(label_hue)
                    labeled_img = cv2.merge([label_hue, blank_ch,
blank_ch])
                    # cvt to BGR for display
                    labeled_img = cv2.cvtColor(labeled_img,
cv2.COLOR_HSV2BGR)
                    # set bg label to black
                    labeled_img[label_hue == 0] = 0

                    # cv2.imshow('L', L1)

                    # Display the resulting frame
                    # cv2.imshow('originalA',imax)
                    # cv2.imshow('originalB', imay)
                    # cv2.imshow('ima2',ima3)
                    # cv2.imshow('imb3', imb3)

                    labeled_img_prima= imutils.resize(labeled_img,
height=480)
                    ##cv2.imshow('skeleton', labeled_img_prima)
                    print(nf)

                    ws1.write(a-1, fil, nf)

                    ##cv2.waitKey(0)
                    ##cv2.destroyAllWindows()
                except:
                    ws1.write(a - 1, fil, "no hay")
                    pass
out_wb.close()
```

F. Programa para arduino (Modulo de microscopio).

```
/*
 * Clock Microstepping demo
 *
 * Moves the stepper motor like the seconds hand of a watch.
 *
 * Copyright (C)2015 Laurentiu Badea
 *
 * This file may be redistributed under the terms of the MIT license.
```

```
 * A copy of this license has been included with this distribution in the file LICENSE.
 */
#include <Arduino.h>|
#include "A4988.h"
#include "MultiDriver.h"
#include "SyncDriver.h"


// Motor steps per revolution. Most steppers are 200 steps or 1.8 degrees/step
#define MOTOR_STEPS 200

// All the wires needed for full functionality
#define DIR A0
#define STEP A1
#define DIR2 7
#define STEP2 8
#define DIR3 10
#define STEP3 11

// microstep control for A4988
#define MS12 5
#define MS22 4
#define MS32 3

// Finales de carrera
```

```
#define fin_carrera1 A3
#define fin_carrera2 A4
#define fin_carrera3 A5

//Led Luz
int ledPin = 9;      // LED connected to digital pin 9

// Creación de los objetos de motor a pasos
A4988 stepper1(MOTOR_STEPS, DIR, STEP, MS12, MS22, MS32);
A4988 stepper2(MOTOR_STEPS, DIR2, STEP2, MS12, MS22, MS32);
A4988 stepper3(MOTOR_STEPS, DIR3, STEP3, MS12, MS22, MS32);

SyncDriver controller(stepper1, stepper2, stepper3);

void setup() {
  /*
   * Set target motor RPM.
   */

  stepper1.setRPM(300);
  stepper1.setMicrostep(16); // make sure we are in full speed mode

  stepper2.setRPM(300);
```

```
stepper2.setMicrostep(16);

stepper3.setRPM(300);
stepper3.setMicrostep(16);

pinMode(fin_carrera1, INPUT);
digitalWrite(fin_carrera1, HIGH);

pinMode(fin_carrera2, INPUT);
digitalWrite(fin_carrera2, HIGH);

pinMode(fin_carrera3, INPUT);
digitalWrite(fin_carrera3, HIGH);

pinMode(ledPin, OUTPUT);   // sets the pin as output
//analogWrite(ledPin, 100);
//setPwmFrequency(6, 1);
//setPwmFrequency(6, 1);

pinMode(MS12, OUTPUT);
pinMode(MS22, OUTPUT);
pinMode(MS32, OUTPUT);

Serial.begin(9600);
while (!Serial) {
```

```
    ; // wait for serial port to connect. Needed for native USB port
only
  }
}

int cont=0;
char funcion="0";

void loop() {
  String response = "";

  setPwmFrequency(ledPin, 8);//Divisor=8,64
  //analogWrite(ledPin, 100);

  //referenciaXYZ();
  //delay(1000);

  if (Serial.available()>0) {
    response = Serial.readStringUntil('\n');
    funcion=response[0];
      }
    //Serial.println(funcion);

    if(funcion=='W'){
      if(response=="Wref"){
          referenciaXYZ();
```

```
        Serial.println("okXYZcero\n");
        }
    else{
        parseMoveXYZ(response);
        Serial.println("okXYZmove\n");
        }
    funcion="0";
  }

 if(funcion=='L'){
    if(response=="Lon\n"){
        //setPwmFrequency(6, 1);
        analogWrite(ledPin, 50);//100
        Serial.println("okLedon\n");
        }
    else{
        analogWrite(ledPin, 0);
        Serial.println("okLedof\n");
        }
    funcion="0";
  }

if(funcion=='X'){
  if(response=="Xref"){
      referenciaX();
      Serial.println("okXref\n");
```

```
    }
  else{
    parseMoveX(response);
    Serial.println("okXmove\n");
    }
  funcion="0";
 }

if(funcion=='Y'){
  if(response=="Yref"){
    referenciaY();
    Serial.println("okYref\n");
    }
  else{
    parseMoveY(response);
    Serial.println("okYmove\n");
    }
  funcion="0";
 }

if(funcion=='Z'){
  if(response=="Zref"){
    referenciaZ();
    Serial.println("okZref\n");
    }
  else{
```

```
        parseMoveZ(response);
        Serial.println("okZmove\n");
        }
    funcion="0";
   }

}

void referenciaXYZ(){
//  stepper1.setMicrostep(16);
//  stepper2.setMicrostep(16);
//  stepper3.setMicrostep(16);

  while(digitalRead(fin_carrera1) == LOW &&
digitalRead(fin_carrera2) == LOW && digitalRead(fin_carrera3)==
LOW){
    controller.rotate(-1,-1,1);
    }

  while(digitalRead(fin_carrera1) == HIGH ||
digitalRead(fin_carrera2) == HIGH || digitalRead(fin_carrera3) ==
HIGH){
    if(digitalRead(fin_carrera1) == HIGH &&
digitalRead(fin_carrera2) == LOW && digitalRead(fin_carrera3) ==
LOW){
          controller.rotate(0,-1,1);}
```

```
   else if(digitalRead(fin_carrera1) == LOW &&
digitalRead(fin_carrera2) == HIGH && digitalRead(fin_carrera3)
== LOW){
        controller.rotate(-1,0,1);}
   else if(digitalRead(fin_carrera1) == LOW &&
digitalRead(fin_carrera2) == LOW && digitalRead(fin_carrera3) ==
HIGH){
        controller.rotate(-1,-1,0);}
   else if(digitalRead(fin_carrera1) == LOW &&
digitalRead(fin_carrera2) == HIGH && digitalRead(fin_carrera3)
== HIGH){
        controller.rotate(-1,0,0);}
   else if(digitalRead(fin_carrera1) == HIGH &&
digitalRead(fin_carrera2) == LOW && digitalRead(fin_carrera3) ==
HIGH){
        controller.rotate(0,-1,0);}
   else if(digitalRead(fin_carrera1) == HIGH &&
digitalRead(fin_carrera2) == HIGH && digitalRead(fin_carrera3)
== LOW){
        controller.rotate(0,0,1);}
   else if(digitalRead(fin_carrera1) == HIGH &&
digitalRead(fin_carrera2) == HIGH && digitalRead(fin_carrera3)
== HIGH){
        controller.rotate(0,0,0);
        break;}

  }

}
```

```
//////////////////7

void referenciaXYZ1(){
//  stepper1.setMicrostep(16);
//  stepper2.setMicrostep(16);
//  stepper3.setMicrostep(16);

  while(digitalRead(fin_carrera1) == LOW &&
digitalRead(fin_carrera2) == LOW && digitalRead(fin_carrera3)==
LOW){
    controller.rotate(-1,-1,-1);
    }

  while(digitalRead(fin_carrera1) == HIGH ||
digitalRead(fin_carrera2) == HIGH || digitalRead(fin_carrera3) ==
HIGH){
    if(digitalRead(fin_carrera1) == HIGH &&
digitalRead(fin_carrera2) == LOW && digitalRead(fin_carrera3) ==
LOW){
        controller.rotate(0,-1,-1);}
    else if(digitalRead(fin_carrera1) == LOW &&
digitalRead(fin_carrera2) == HIGH && digitalRead(fin_carrera3)
== LOW){
        controller.rotate(-1,0,-1);}
    else if(digitalRead(fin_carrera1) == LOW &&
digitalRead(fin_carrera2) == LOW && digitalRead(fin_carrera3) ==
HIGH){
        controller.rotate(-1,-1,0);}
```

```
   else if(digitalRead(fin_carrera1) == LOW &&
digitalRead(fin_carrera2) == HIGH && digitalRead(fin_carrera3)
== HIGH){
        controller.rotate(-1,0,0);}
   else if(digitalRead(fin_carrera1) == HIGH &&
digitalRead(fin_carrera2) == LOW && digitalRead(fin_carrera3) ==
HIGH){
        controller.rotate(0,-1,0);}
   else if(digitalRead(fin_carrera1) == HIGH &&
digitalRead(fin_carrera2) == HIGH && digitalRead(fin_carrera3)
== LOW){
        controller.rotate(0,0,-1);}
   else if(digitalRead(fin_carrera1) == HIGH &&
digitalRead(fin_carrera2) == HIGH && digitalRead(fin_carrera3)
== HIGH){
        controller.rotate(0,0,0);
        break;}

  }

}

////////////////Mover eje X/////////////////////////////////////////////

void referenciaX(){
  stepper1.setMicrostep(16);
  while(digitalRead(fin_carrera1) == LOW){
```

```
    stepper1.rotate(-1);
  }
}

void parseMoveX(String response){
      //Serial.print("[");
      String movement="";
      String movestep="";
      int i=0;
      int movepaso=0;
      char c = response[i];
      char movez=response[i];

      while (response[i]!=','){
        c = response[i];
        movement+=c;
        //Serial.print(c);
        i=i+1;
      }

    //Takes on more char to identify the number of movement
    c = response[i-1];
    //Serial.println(c);
      movez=response[i+1];
       //c is the number of movement
      if(c=='1'){
```

```
  stepper1.setMicrostep(1);
  }
 else if(c=='2'){
  stepper1.setMicrostep(2);
  }
 else if(c=='4'){
  stepper1.setMicrostep(4);
  }
 else if (c=='8'){
  stepper1.setMicrostep(8);
  }
 else{
  stepper1.setMicrostep(16);
  }
 //Serial.println("----------------------------------------");

//Serial.print("]");
 while (response.length()!=i){ //Take first 5 chars from readings
  c = response[i+2];
  movestep+=c;
  //Serial.print(c);
  i=i+1;
 }
movepaso=movestep.toInt();

if (movez=='L'){
```

```
      stepper1.rotate(movepaso);
      delay(1000/64);
     }
     else if (movez=='R'){
      stepper1.rotate(-movepaso);
      delay(1000/64);
     }

}

////////////////Mover eje Y/////////////////////////////////////////////
void referenciaY(){

  stepper2.rotate(-1);
  stepper2.setMicrostep(16);
  while(digitalRead(fin_carrera2) == LOW){
    stepper2.rotate(-1);
  }
}

void parseMoveY(String response){
      //Serial.print("[");
      String movement="";
      String movestep="";
      int i=0;
      int movepaso=0;
```

```
  char c = response[i];
  char movez=response[i];

  //stepper2.setRPM(120);

  while (response[i]!=','){
    c = response[i];
    movement+=c;
    //Serial.print(c);
    i=i+1;
 }

//Takes on more char to identify the number of movement
c = response[i-1];
//Serial.println(c);
  movez=response[i+1];
   //c is the number of movement
  if(c=='1'){
    stepper2.setMicrostep(1);
    }
  else if(c=='2'){
    stepper2.setMicrostep(2);
    }
  else if(c=='4'){
    stepper2.setMicrostep(4);
    }
```

```
 else if (c=='8'){
   stepper2.setMicrostep(8);
   }
 else{
   stepper2.setMicrostep(16);
   }
 //Serial.println("----------------------------------------");

//Serial.print("]");
 while (response.length()!=i){ //Take first 5 chars from readings
   c = response[i+2];
   movestep+=c;
   //Serial.print(c);
   i=i+1;
 }
movepaso=movestep.toInt();

if (movez=='L'){
 stepper2.rotate(movepaso);
 delay(1000/64);
}
else if (movez=='R'){
 stepper2.rotate(-movepaso);
 delay(1000/64);
}
```

```
}



////////////////Mover eje Z/////////////////////////////////////////////



void referenciaZ(){
  stepper3.setMicrostep(16);
  while(digitalRead(fin_carrera3) == LOW){
    stepper3.rotate(1);//-1
  }
}

void parseMoveZ(String response){
      //Serial.print("[");
      String movement="";
      String movestep="";
      int i=0;
      int movepaso=0;
      char c = response[i];
      char movez=response[i];

      while (response[i]!=','){
        c = response[i];
        movement+=c;
        //Serial.print(c);
```

```
   i=i+1;
 }

//Takes on more char to identify the number of movement
c = response[i-1];
//Serial.println(c);
  movez=response[i+1];
   //c is the number of movement
  if(c=='1'){
    stepper3.setMicrostep(1);
    }
  else if(c=='2'){
    stepper3.setMicrostep(2);
    }
  else if(c=='4'){
    stepper3.setMicrostep(4);
    }
  else if (c=='8'){
    stepper3.setMicrostep(8);
    }
  else{
    stepper3.setMicrostep(16);
    }
  //Serial.println("----------------------------------------");

 //Serial.print("]");
```

```
      while (response.length()!=i){ //Take first 5 chars from readings
        c = response[i+2];
        movestep+=c;
        //Serial.print(c);
        i=i+1;
      }
     movepaso=movestep.toInt();

     if (movez=='L'){
      stepper3.rotate(movepaso);
      delay(1000/64);
     }
     else if (movez=='R'){
      stepper3.rotate(-movepaso);
      delay(1000/64);
     }

}

//////Set PWM frecuencia////////////////////

void setPwmFrequency(int pin, int divisor) {
  byte mode;
  if(pin == 5 || pin == 6 || pin == 9 || pin == 10) {
    switch(divisor) {
      case 1: mode = 0x01; break;
```

```
      case 8: mode = 0x02; break;
      case 64: mode = 0x03; break;
      case 256: mode = 0x04; break;
      case 1024: mode = 0x05; break;
      default: return;
    }
    if(pin == 5 || pin == 6) {
      TCCR0B = TCCR0B & 0b11111000 | mode;
    } else {
      TCCR1B = TCCR1B & 0b11111000 | mode;
    }
  } else if(pin == 3 || pin == 11) {
    switch(divisor) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 32: mode = 0x03; break;
      case 64: mode = 0x04; break;
      case 128: mode = 0x05; break;
      case 256: mode = 0x06; break;
      case 1024: mode = 0x07; break;
      default: return;
    }
    TCCR2B = TCCR2B & 0b11111000 | mode;
  }
}
```

```
//////////////////////////MOVE XYZ/////////////////////////////////////////////

void parseMoveXYZ(String response){
   int i=0;
   String movestepX="";
   String movestepY="";
   String movestepZ="";
   int movepasoX=0;
   int movepasoY=0;
   int movepasoZ=0;
   char c = response[i];

   //Serial.print(response);
   i=i+1;

   while (response[i]!=','){
     c = response[i];
     movestepX+=c;
     //Serial.print(c);
     i=i+1;
   }

   i=i+1;

   while (response[i]!=','){ //Take first 5 chars from readings
     c = response[i];
```

```
      movestepY+=c;
      //Serial.print(c);
      i=i+1;
    }

    i=i+1;

    while (response.length()!=i){ //Take first 5 chars from readings
      c = response[i];
      movestepZ+=c;
      //Serial.print(c);
      i=i+1;
    }

    movepasoX=movestepX.toInt();
    movepasoY=movestepY.toInt();
    movepasoZ=movestepZ.toInt();
    int movepas= movepasoX + movepasoY + movepasoZ;
    Serial.print(movepas);

    controller.rotate(movepasoX,movepasoY,movepasoZ);
    delay(100/64);
}


//void time1(){
```

```
//
//  if (currentMillis - previousMillis >= interval) {
//    previousMillis = currentMillis;
//  }
//
//}
```

G. Diagrama de circuito impreso (Módulo de Microscopio)

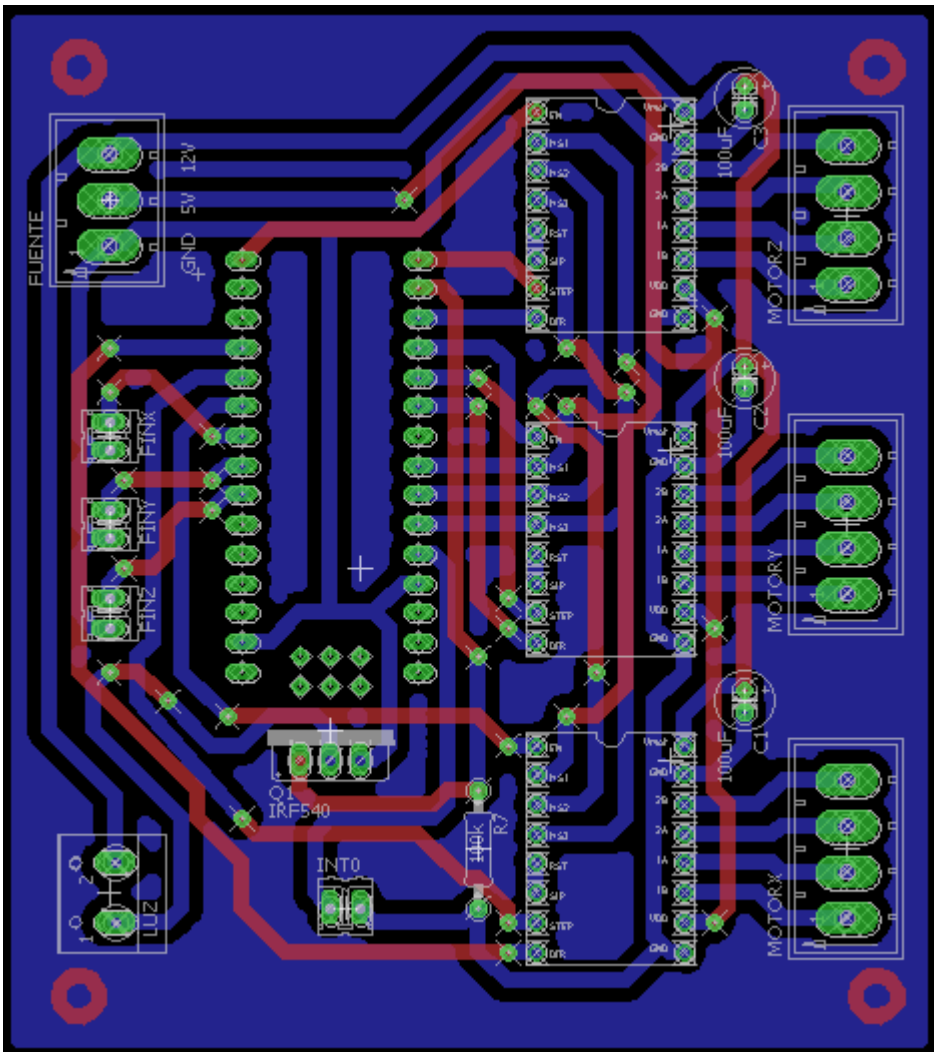


Figura 0.2: Esquemático impreso del módulo de tinción.

H. Control de temperatura para diversas posiciones del sensor de referencia del módulo de tinción.

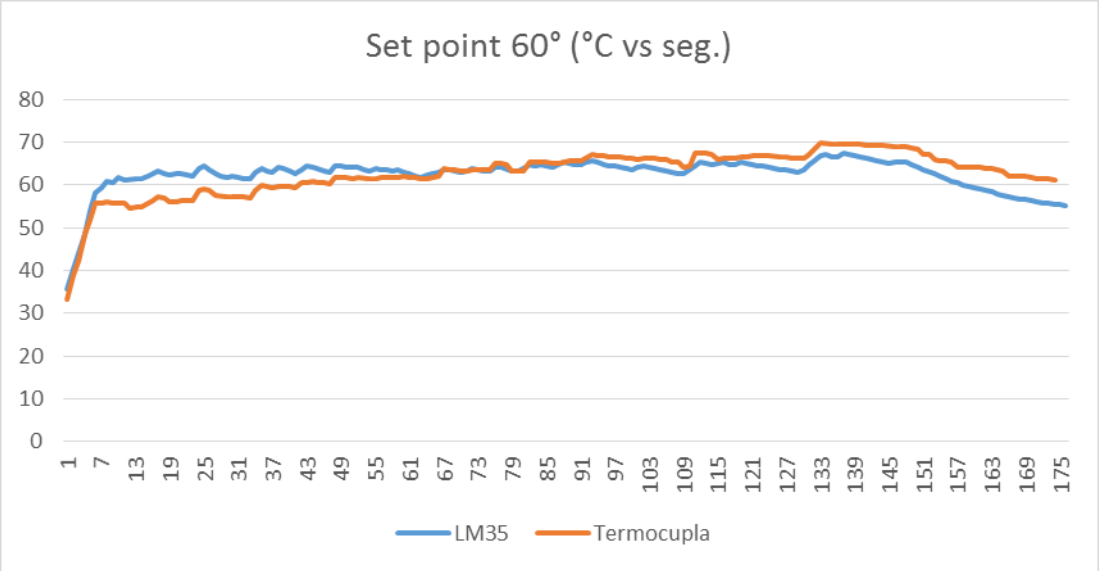


Figura 0.3: Respuesta del sensor LM35 y termocupla con una referencia de 60°C

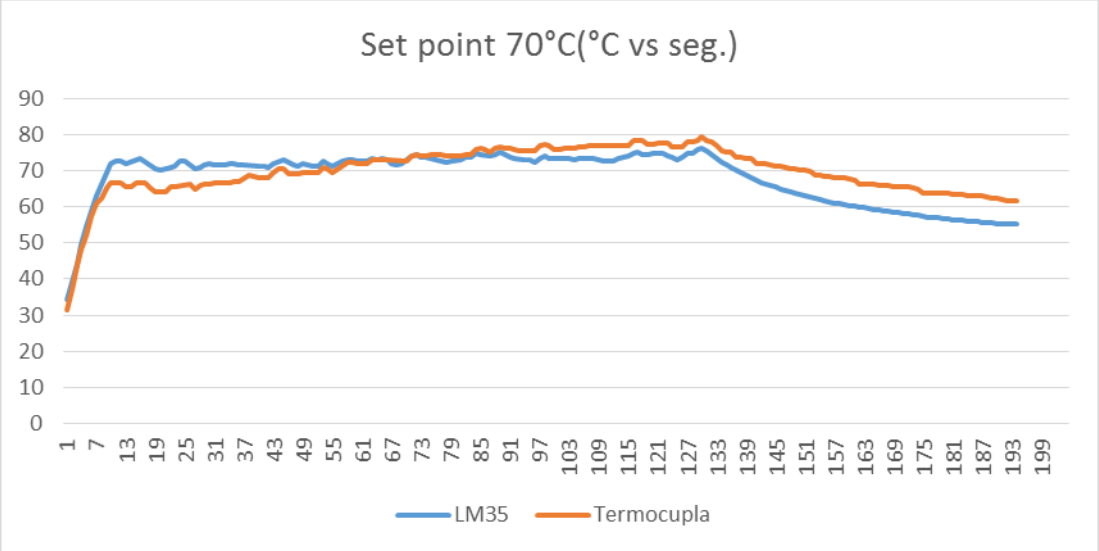


Figura 0.4: Respuesta del sensor LM35 y termocupla con una referencia de 70°C

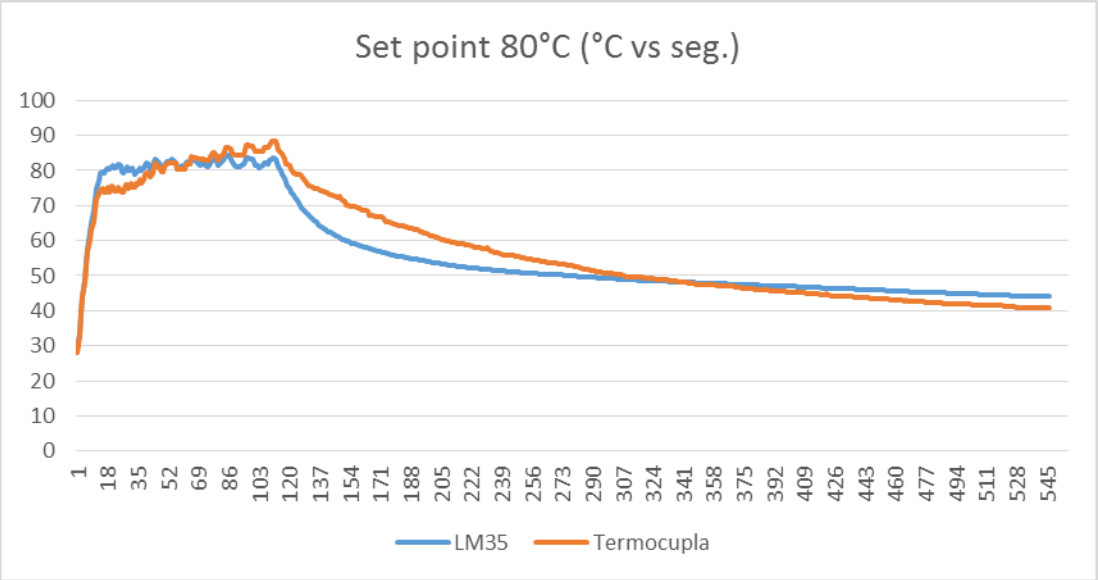


Figura 0.5: Respuesta del sensor LM35 y termocupla con una referencia de 80°C

I. Programa de la interfaz gráfica y del proceso de diagnóstico

- Diseño de la interfaz:

```
#:kivy 1.7.2
#:import kivy kivy

<SaveDialog@Popup>:
    #title='Test popup'
    size_hint:(0.4,0.4)
    auto_dismiss: False

    GridLayout:
        rows:2
        BoxLayout:
            size_hint:(1,0.7)
            Label:
                text: '¿Desea guardar el diagnóstico?'
                #on_release: root.dismiss()
        BoxLayout:
            size_hint:(1,0.3)
            Button:
                text: 'GUARDAR'
                on_release: root.save()
            Button:
                text: 'CERRAR'
                on_release: root.cancel()

<RV>:
    #orientation: 'vertical'
    GridLayout:
        rows:2
        BoxLayout:
            orientation: 'horizontal'
            size_hint:(1,0.9)
```

```
            BoxLayout:
                orientation: 'vertical'
                size_hint:(.335,1)

                canvas.before:
                    Color:
                        rgb: 0.82, 0.12, 0.12 # your color here
                    Rectangle:
                        pos: self.pos
                        size: self.size

                BoxLayout:
                    size_hint:(1,0.3)

                    Button:
                        id: logo
                        background_normal: 'assets/normal.png'
                        background_down: 'assets/pressed.png'
                        on_release: root.logo_clicked()

                        Image:
                            source: 'assets/logo0.png'
                            #size:indow.size

size:self.parent.parent.size[0]*0.7,self.parent.parent.size[1]*0.7
                            center_x: self.parent.center_x
                            center_y: self.parent.center_y

                            #width:100
                            halign: "center"
                            allow_stretch: True

                BoxLayout:
                    size_hint:(1,0.2)
                    Button:
                        id: start
                        background_normal: 'assets/normal.png'
                        background_down: 'assets/pressed.png'
                        on_release: root.back_screen()

                        Image:
                            id:alpha
                            source: 'assets/menu_start.png'
                            opacity: 1.2

size:153,34#self.parent.parent.size[0]*0.5709,self.parent.parent.siz
e[1]*0.3778
                            center_x: self.parent.center_x
                            center_y: self.parent.center_y
                            allow_stretch: True

                BoxLayout:
                    size_hint:(1,0.2)

                    Button:
                        id: disinfection
                        background_normal: 'assets/normal.png'
                        background_down: 'assets/pressed.png'
                        on_release: root.disinfection_clicked()

                        Image:
```

```
                    source: 'assets/menu_disinfection.png'
                    opacity: 1.2
                    center_y: self.parent.center_y
                    center_x: self.parent.center_x
                    size: 153, 34
                    allow_stretch: True

            BoxLayout:
                size_hint:(1,0.2)
                Button:
                    id: history
                    background_normal: 'assets/normal.png'
                    background_down: 'assets/pressed.png'
                    on_release: root.history_clicked()

                    Image:
                        source: 'assets/menu_history.png'
                        opacity: 1.2
                        center_y: self.parent.center_y
                        center_x: self.parent.center_x
                        size: 153, 34
                        allow_stretch: True

            BoxLayout:
                size_hint:(1,0.3)
                Button:
                    id: turnoff
                    background_normal: 'assets/normal.png'
                    background_down: 'assets/pressed.png'
                    on_release: root.turnoff_clicked()

                    Image:
                        source: 'assets/menu_turnoff.png'
                        opacity: 1.2
                        center_y: self.parent.center_y
                        center_x: self.parent.center_x
                        size: 153, 34
                        allow_stretch: True


        BoxLayout:
            orientation: 'vertical'
            size_hint:(0.665,1)

            BoxLayout:
                orientation: "vertical"

                canvas.before:
                    Color:
                        rgb: 0.88, 0.88, 0.88  # your color here
                    Rectangle:
                        pos: self.pos
                        size: self.size

                BoxLayout:
                    size_hint: 0.5,0.1
                    padding: [20,5,20,5]
                    TextInput:
                        size_hint_x:0.3
                        id: txtinput0
                    Button:
```

```
                            size_hint_x:0.2
                            background_normal:''
                            background_color: 0.82, 0.12, 0.12,1
                            text: "Buscar"
                            on_release: root.get_users()

                    GridLayout:
                        size_hint: 1, None
                        size_hint_y: None
                        height: 25
                        cols: 5
                        spacing: 5
                        padding: [5,0,5,0]

                        MyLabel:
                            text: "CODIGO"
                            halign: 'center'

                        MyLabel:
                            text: "DIAGNOSTICO"
                            halign: 'center'

                        MyLabel:
                            text: "CAMPOS"
                            halign: 'center'

                        MyLabel:
                            text: "BACILOS"
                            halign: 'center'

                        MyLabel:
                            text: "FECHA"
                            halign: 'center'

                    BoxLayout:
                        #padding: [5,0,5,0]
                        RecycleView:

                            viewclass: 'MyLabel'
                            data: [{'text': str(x)} for x in
root.data_items]
                            effect_cls: "ScrollEffect"

                            RecycleGridLayout:
                                size_hint: 1, None
                                size_hint_y: None
                                cols: 5
                                default_size: None, dp(26)

                                default_size_hint: 1,None
                                size_hint_y: None
                                height: self.minimum_height
                                orientation: 'vertical'
                                multiselect: True
                                touch_multiselect: True
                                spacing: 5
                                padding: [5,5,5,5]

        #Box:'08:30pm'

<MainScreen>:
```

```
    #orientation: 'vertical'
    GridLayout:
        rows:2
        BoxLayout:
            orientation: 'horizontal'
            size_hint:(1,0.9)

            BoxLayout:
                orientation: 'vertical'
                size_hint:(.335,1)

                canvas.before:
                    Color:
                        rgb: 0.82, 0.12, 0.12 # your color here
                    Rectangle:
                        pos: self.pos
                        size: self.size

                BoxLayout:
                    size_hint:(1,0.3)

                    Button:
                        id: logo
                        background_normal: 'assets/normal.png'
                        background_down: 'assets/pressed.png'
                        on_release: root.logo_clicked()

                        Image:
                            source: 'assets/logo0.png'
                            #size:indow.size

size:self.parent.parent.size[0]*0.7,self.parent.parent.size[1]*0.7
                            center_x: self.parent.center_x
                            center_y: self.parent.center_y

                            #width:100
                            halign: "center"
                            allow_stretch: True

                BoxLayout:
                    size_hint:(1,0.2)
                    Button:
                        id: start
                        background_normal: 'assets/normal.png'
                        background_down: 'assets/pressed.png'
                        on_release: root.start_clicked()

                        Image:
                            id:alpha
                            source: 'assets/menu_start.png'
                            opacity: 1.2

size:153,34#self.parent.parent.size[0]*0.5709,self.parent.parent.siz
e[1]*0.3778
                            center_x: self.parent.center_x
                            center_y: self.parent.center_y
                            allow_stretch: True


                BoxLayout:
```

```
            size_hint:(1,0.2)
            Button:
                id: history
                background_normal: 'assets/normal.png'
                background_down: 'assets/pressed.png'
                on_release: root.history_clicked()

                Image:
                    source: 'assets/menu_history.png'
                    opacity: 1.2
                    center_y: self.parent.center_y
                    center_x: self.parent.center_x
                    size: 153, 34
                    allow_stretch: True

        BoxLayout:
            size_hint:(1,0.3)
            Button:
                id: turnoff
                background_normal: 'assets/normal.png'
                background_down: 'assets/pressed.png'
                on_release: root.turnoff_clicked()

                Image:
                    source: 'assets/menu_turnoff.png'
                    opacity: 1.2
                    center_y: self.parent.center_y
                    center_x: self.parent.center_x
                    size: 153, 34
                    allow_stretch: True


    BoxLayout:
        orientation: 'vertical'
        size_hint:(0.665,1)
        canvas.before:
            Color:
                rgb: 0.88, 0.88, 0.88 # your color here
            Rectangle:
                pos: self.pos
                size: self.size

        BoxLayout:
            horientation: 'horizontal'
            size_hint:(1,0.05)
            canvas.before:
                Color:
                    rgb: 0.88, 0.88, 0.88 # your color here
                Rectangle:
                    pos: self.pos
                    size: self.size

        BoxLayout:
            size_hint_y: 0.3
            GridLayout:
                cols: 4
                BoxLayout: ## here is one Box
                    orientation: "vertical"
                    padding: [20,0,20,0]
                    BoxLayout:
                        size_hint_y: 0.3
```

```
                                TextInput:
                                    id: t1
                                    text:
                            BoxLayout:
                                padding: [0,10,0,0]
                                Image:
                                    id: m1
                                    nocache: 1
                                    source: 'assets/muestra_0.png'
                                    y: self.parent.y#+self.size[1]/2
                                    x: self.parent.x#+self.size[0]/2
                                    halign: "center"
                                    #size: 60, 160
                                    allow_stretch: True
                            BoxLayout:
                                size_hint_y: 0.1
                                padding: [0,5,0,0]
                                Image:
                                    id: c1#cruces
                                    nocache: 1
                                    source:
'assets/start_process00.png'
                                    y: self.parent.y#+self.size[1]/2
                                    x: self.parent.x#+self.size[0]/2
                                    halign: "center"
                                    #size: 60, 160
                                    allow_stretch: True

                        BoxLayout: ## here is another Box
                            orientation: "vertical"
                            padding: [20,0,20,0]
                            BoxLayout:
                                size_hint_y: 0.3
                                TextInput:
                                    id: t2
                            BoxLayout:
                                padding: [0,10,0,0]
                                Image:
                                    id: m2
                                    nocache: 1
                                    source: 'assets/muestra_0.png'
                                    y: self.parent.y#+self.size[1]/2
                                    x: self.parent.x#+self.size[0]/2
                                    halign: "center"
                                    #size: 60, 160
                                    allow_stretch: True
                            BoxLayout:
                                size_hint_y: 0.1
                                padding: [0,5,0,0]
                                Image:
                                    id: c2#cruces
                                    nocache: 1
                                    source:
'assets/start_process00.png'
                                    y: self.parent.y#+self.size[1]/2
                                    x: self.parent.x#+self.size[0]/2
                                    halign: "center"
                                    #size: 60, 160
                                    allow_stretch: True
```

```
                        BoxLayout: ## here is one Box
                            orientation: "vertical"
                            padding: [20,0,20,0]
                            BoxLayout:
                                size_hint_y: 0.3
                                TextInput:
                                    id: t3
                            BoxLayout:
                                padding: [0,10,0,0]
                                Image:
                                    id: m3
                                    nocache: 1
                                    source: 'assets/muestra_0.png'
                                    y: self.parent.y#+self.size[1]/2
                                    x: self.parent.x#+self.size[0]/2
                                    halign: "center"
                                    #size: 60, 160
                                    allow_stretch: True
                            BoxLayout:
                                padding: [0,5,0,0]
                                size_hint_y: 0.1
                                Image:
                                    id: c3#cruces
                                    nocache: 1
                                    source:
'assets/start_process00.png'
                                    y: self.parent.y#+self.size[1]/2
                                    x: self.parent.x#+self.size[0]/2
                                    halign: "center"
                                    #size: 60, 160
                                    allow_stretch: True


                        BoxLayout: ## here is another Box
                            orientation: "vertical"
                            padding: [20,0,20,0]
                            BoxLayout:
                                size_hint_y: 0.3
                                TextInput:
                                    id: t4
                            BoxLayout:
                                padding: [0,10,0,0]
                                Image:
                                    id: m4
                                    nocache: 1
                                    source: 'assets/muestra_0.png'
                                    y: self.parent.y#+self.size[1]/2
                                    x: self.parent.x#+self.size[0]/2
                                    halign: "center"
                                    #size: 60, 160
                                    allow_stretch: True
                            BoxLayout:
                                size_hint_y: 0.1
                                padding: [0,5,0,0]
                                Image:
                                    id: c4#cruces
                                    nocache: 1
                                    source:
'assets/start_process00.png'
                                    y: self.parent.y#+self.size[1]/2
                                    x: self.parent.x#+self.size[0]/2
```

```
                                    halign: "center"
                                    #size: 60, 160
                                    allow_stretch: True


                BoxLayout:
                    size_hint_y: 0.4
                    id:p2
                    padding: [0,5,0,0]
                    #Label:#Tienes que verficar con label
                        #id: p2
                        #size_hint:(1,0.4)
                    y: self.parent.y#+self.size[1]/2
                    x: self.parent.x#+self.size[0]/2

        #Box:'08:30pm'
        #BoxLayout:

<MyLabel@Label>
    canvas.before:
        Color:
            rgb:0.82, 0.12, 0.12,1
        Rectangle:
            pos: self.pos
            size: self.size

<KeyboardScreen>:
    displayLabel: displayLabel
    kbContainer: kbContainer
    BoxLayout:
        orientation: 'vertical'

        canvas.before:
            Color:
                rgb: 0.59, 0, 0 # your color here
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            size_hint_y: 0.15
            text: "INGRESE LOS CODIGOS DE MUESTRAS"

        BoxLayout:
            size_hint_y: 0.25
            GridLayout:
                cols: 4
                BoxLayout: ## here is one Box
                    orientation: "vertical"
                    padding: [20,0,20,0]
                    BoxLayout:
                        size_hint_y: 0.3
                        TextInput:
                    BoxLayout:
                        padding: [0,10,0,0]
                        Image:
                            id: p1
                            nocache: 1
                            source: 'assets/muestra_0.png'
                            y: self.parent.y#+self.size[1]/2
                            x: self.parent.x#+self.size[0]/2
                            halign: "center"
```

```
                    #size: 60, 160
                    allow_stretch: True

            BoxLayout: ## here is another Box
                orientation: "vertical"
                padding: [20,0,20,0]
                BoxLayout:
                    size_hint_y: 0.3
                    TextInput:
                BoxLayout:
                    padding: [0,10,0,0]
                    Image:
                        id: px2#p2
                        nocache: 1
                        source: 'assets/muestra_0.png'
                        y: self.parent.y#+self.size[1]/2
                        x: self.parent.x#+self.size[0]/2
                        halign: "center"
                        #size: 60, 160
                        allow_stretch: True

            BoxLayout: ## here is one Box
                orientation: "vertical"
                padding: [20,0,20,0]
                BoxLayout:
                    size_hint_y: 0.3
                    TextInput:
                BoxLayout:
                    padding: [0,10,0,0]
                    Image:
                        id: p3
                        nocache: 1
                        source: 'assets/muestra_0.png'
                        y: self.parent.y#+self.size[1]/2
                        x: self.parent.x#+self.size[0]/2
                        halign: "center"
                        #size: 60, 160
                        allow_stretch: True

            BoxLayout: ## here is another Box
                orientation: "vertical"
                padding: [20,0,20,0]
                BoxLayout:
                    size_hint_y: 0.3
                    TextInput:
                BoxLayout:
                    padding: [0,10,0,0]
                    Image:
                        id: p4
                        nocache: 1
                        source: 'assets/muestra_0.png'
                        y: self.parent.y#+self.size[1]/2
                        x: self.parent.x#+self.size[0]/2
                        halign: "center"
                        #size: 60, 160
                        allow_stretch: True

    BoxLayout:
        id: kbContainer
        size_hint_y: 0.2
        orientation: "horizontal"
```

```
            padding: [100,0,100,0]

        Label:
            id: displayLabel
            size_hint_y: 0.05
            markup: True
            text: "[b]Key pressed[/b] - None"
            halign: "center"
        Button:
            background_normal:''
            background_color: 0.82, 0.12, 0.12,1 # your color here
            text: "Volver"
            size_hint: (0.15,0.05)
            on_release: root.back_screen()
            halign: "center"
        Widget:
            # Just a space taker to allow for the popup keyboard
            size_hint_y: 0.5


<VRFloatLayout>:
    AnchorLayout:
        anchor_x: 'center'
        anchor_y: 'top'
        ScreenManager:
            id: manager
            size_hint: 1, .9
    AnchorLayout:
        anchor_x: 'center'
        anchor_y: 'bottom'
        BoxLayout:
            orientation: 'horizontal'
            size_hint:(1,0.1)
            Label:
                id: my_time
                text: '08:30pm'
                font_size: 14
                canvas.before:
                    Color:
                        rgb: 0.58, 0, 0 # your color here
                    Rectangle:
                        pos: self.pos
                        size: self.size
```

- **Programa de la interfaz:**

```
# coding:utf-8
from kivy.app import App
from kivy.uix.image import Image
from kivy.clock import Clock
from kivy.graphics.texture import Texture
import cv2
import numpy as np

from kivy.uix.widget import Widget
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.floatlayout import FloatLayout
from kivy.uix.image import Image
```

```
from kivy.uix.button import Button

import serial
import time
import serial.tools.list_ports

class KivyCamera(Image):
    def __init__(self, capture, fps, **kwargs):
        super(KivyCamera, self).__init__(**kwargs)
        self.capture = capture
        Clock.schedule_interval(self.update, 1.0 / fps)

    def update(self, dt):
        ret, frame = self.capture.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        rows, cols = gray.shape
        fil, col = int(rows / 2), int(cols / 2)

        f = np.fft.fft2(gray)
        imT2 = np.fft.fftshift(f)#imT2 = np.log(np.fft.fftshift(f))
        imf2 = 20*np.log(np.abs(imT2))#imf2 = np.abs(imT2)
        imf3 = imf2[fil][0:499]
        promedio = np.mean(imf3)
        print(promedio)

        if ret:
            # convert it to texture
            buf1 = cv2.flip(frame, 0)
            buf = buf1.tostring()
            image_texture = Texture.create(
                size=(frame.shape[1], frame.shape[0]),
colorfmt='bgr')
            image_texture.blit_buffer(buf, colorfmt='bgr',
bufferfmt='ubyte')
            # display image from the texture
            self.texture = image_texture

class CamApp(App):
    def build(self):
        self.my_camera = Image()

        self.arduinoPort = serial.Serial("COM4", 250000, timeout=1)
        time.sleep(1.8)
        self.Lon()
        self.capture = cv2.VideoCapture(1)
        self.my_camera = KivyCamera(capture=self.capture, fps=30)

        layoutVideo = FloatLayout()
        layoutBotoes = BoxLayout(orientation='horizontal')

        buttonUp = Button(text='SUBE', size_hint=(0.5, 0.1))
        buttonUp.bind(on_press=self.sube)
        buttonDown = Button(text='BAJA', size_hint=(0.5, 0.1))
        buttonDown.bind(on_press=self.baja)

        layoutBotoes.add_widget(buttonUp)  # acrescentamos o botao
cinza numa linha horizontal
        layoutBotoes.add_widget(buttonDown)  # acrescentamos o botao
cinza numa linha horizontal

        layoutVideo.add_widget(self.my_camera)
```

```
        layoutVideo.add_widget(layoutBotoes)

        return layoutVideo

    def sube(self, button):
        print("SubeZ")
        flagCharacter = "Z 16,L+100\n"
        self.enviar_serial(flagCharacter.encode())

    def baja(self, button):
        print("Bajaz")
        flagCharacter = "Z 16,L-100\n"
        self.enviar_serial(flagCharacter.encode())

    def on_stop(self):
        #without this, app will not exit even if the window is
closed
        self.capture.release()
        # Cerrando puerto serial
        flagCharacter = "Lof\n"
        self.enviar_serial(flagCharacter.encode())
        self.arduinoPort.close()

    def Lon(self):
        flagCharacter = "Lon\n"
        self.enviar_serial(flagCharacter.encode())

    def enviar_serial(self,flagCharacter):
        #time.sleep(1)

        #arduinoPort = serial.Serial("COM12", 9600, timeout=1)

        #time.sleep(1.8)
        self.arduinoPort.write(flagCharacter)

        while 1:
            getSerialValue = self.arduinoPort.readline().decode()
            if "ok" in getSerialValue:
                break
        #print('\n Valor retornado de Arduino: %s' %
(getSerialValue))
        #time.sleep(1.8)

if __name__ == '__main__':
    CamApp().run()
```

- **Programa que recorre las muestras:**

```
from sys import platform
import serial
import time
import serial.tools.list_ports

def enviar_serial(flagCharacter):##en la otra version se asume ya
codificado
    arduinoPort.write(flagCharacter.encode())

    while 1:
        getSerialValue = arduinoPort.readline().decode()
        if "ok" in getSerialValue:
            break
```

```
    print('\n Valor retornado de Arduino: %s' % (getSerialValue))
    time.sleep(0.000001)

def Lon(self):
    flagCharacter = "Lon\n"
    self.enviar_serial(flagCharacter.encode())

def mover_campo(pasos_por_campo_X,pasos_por_campo_Y):
    for a in range(0, pasos_por_campo_X):

        flagCharacter = "X 16,L-1\n"  ##Eje Z 11.4cm ----2150-----
>188.6grados/cm, Xmax=4300, Ymax=4500, Zmax=-4900(obs:limit switch
position)
        # 'k'
        enviar_serial(flagCharacter)

    for a in range(0, pasos_por_campo_Y):

        flagCharacter = "Y 16,L-1\n"  ##Eje Z 11.4cm ----2150-----
>188.6grados/cm, Xmax=4300, Ymax=4500, Zmax=-4900(obs:limit switch
position)
        # 'k'
        enviar_serial(flagCharacter)

def referencia(ref_X,ref_Y,ref_Z):

    if ref_X==1 and ref_Y ==1 and ref_Z==1:
        flagCharacter = "Wref\n"
        enviar_serial(flagCharacter)

    if ref_X==1 and ref_Y ==0 and ref_Z==0:
        flagCharacter = "Xref\n"
        enviar_serial(flagCharacter)

    if ref_X==0 and ref_Y ==1 and ref_Z==0:
        flagCharacter = "Yref\n"
        enviar_serial(flagCharacter)

    if ref_X==0 and ref_Y ==0 and ref_Z==1:
        flagCharacter = "Zref\n"
        enviar_serial(flagCharacter)

    if ref_X==1 and ref_Y ==1 and ref_Z==0:
        flagCharacter = "Xref\n"
        enviar_serial(flagCharacter)
        flagCharacter = "Yref\n"
        enviar_serial(flagCharacter)

    if ref_X==0 and ref_Y ==1 and ref_Z==1:
        flagCharacter = "Yref\n"
        enviar_serial(flagCharacter)
        flagCharacter = "Zref\n"
        enviar_serial(flagCharacter)

    if ref_X==1 and ref_Y ==0 and ref_Z==1:
        flagCharacter = "Xref\n"
        enviar_serial(flagCharacter)
        flagCharacter = "Zref\n"
        enviar_serial(flagCharacter)

ports = list(serial.tools.list_ports.comports())
```

```
for port_no, description, address in ports:
    if 'Arduino' in description:
        print(port_no)
    if 'CH340' in description:
        print(port_no)

arduinoPort = serial.Serial("COM2", 9600, timeout=1)
time.sleep(1.8)

#Encender el Led
Lon()

#Posicionar en la referencia
ref_X=1;
ref_Y=1;
ref_Z=1;
referencia(ref_X,ref_Y,ref_Z)

#Posicionar en el ingreso de muestras
flagCharacter = "W3800,4500,-4000\n" #modificar
enviar_serial(flagCharacter)

#Solicitar el ingreso de los identificadores de las muestras en la
pantalla

#Recorrer las muestras
pos_1="W3800,4500,-4000\n"
pos_2="W3800,4500,-4000\n"
pos_3="W3800,4500,-4000\n"
pos_4="W3800,4500,-4000\n"
posiciones_muestras=[pos_1,pos_2,pos_3,pos_4]

for i in range(0, 4):
    # Posicionar en el extremo de la primera muestra
    referencia(1,1,1)

    #Ubicar en la posicion de la i muestra
    flagCharacter = posiciones_muestras[i]
    enviar_serial(flagCharacter)

    campos_horizontal = 10
    campos_vertical = 10
    pasos_por_campo_X = 100;
    pasos_por_campo_Y = 0;

    for a in range(0, campos_vertical):
        for b in range(0, campos_horizontal):
            mover_campo(pasos_por_campo_X, pasos_por_campo_Y)
            # Funcion de Autoenfoque
            # Funcion de conteno y diagnóstico
        pasos_por_campo_X = 100;
        pasos_por_campo_Y = 100;


# Cerrando puerto serial
arduinoPort.close()
```