INSTITUTO TECNOLÓGICO SUPERIOR DE TEZIUTLÁN

Tesis

"Sistema de control para un dispositivo de infusión de soluciones por vía intravenosa en centros de salud"

PRESENTA:

LUIS ÁNGEL HERNÁNDEZ REYES

CON NÚMERO DE CONTROL

19TE0827

PARA OBTENER EL TÍTULO DE:

INGENIERO MECATRONICO

CLAVE DEL PROGRAMA ACADÉMICO

IMCT-2010-229

DIRECTOR (A) DE TESIS:

M.I.M. LUIS MANUEL GARCÍA MARTÍNEZ

"La Juventud de hoy, Tecnología del Mañana"

TEZIUTLÁN, PUEBLA, FEBRERO 2024

PRELIMINARES

AGRADECIMIENTOS

A mi padre, a mi madre y a mi hermana; por darme todo su apoyo incondicional, y por ser mis mejores maestros desde mi infancia, por darme todo lo necesario para lograr mis metas y siempre animarme a ser mejor.

A mis asesores, por compartir su conocimiento conmigo, por ayudarme con todas mis dudas y por tenerme mucha paciencia durante estos meses, no podría haber pedido una mejor guía en este paso tan importante.

A mis mejores amigos dentro y fuera de la carrera, sin su apoyo en las tardes de estudio, ni su compañía en las tardes de distracción habría sido posible llegar hasta este punto, especialmente gracias a mi amigo Sebastián Pineda, su compañía durante la pandemia fue la única razón por la que no enloquecí.

RESUMEN

El proyecto busca encontrar una solución que cumpla con lo necesario para el control de bombas de infusión a través de una interfaz amigable, robusta y segura. Esto con el fin de contribuir a una administración precisa y eficiente de fluidos en entornos médicos.

Por ello, se trabajó utilizando Tkinter en Python con el fin de desarrollar una interfaz gráfica de usuario (GUI) para el control de bombas de infusión. Esta GUI proporciona a los usuarios una plataforma eficaz para ajustar la velocidad de infusión, configurar parámetros y supervisar con precisión la administración de fluidos. Utilizando también sensores capaces de mostrar diferentes valores y generar un historial accesible en cualquier momento, recopilando información sobre la temperatura corporal del paciente, asi como su nivel de oxígeno en la sangre, dicho historial se presenta mediante gráficos de líneas que representan su variación con respecto al tiempo.

La interfaz muestra la información vital en todo momento. Incluye otras ventanas con funciones para iniciar el proceso de infusión, siempre que se verifique la autoridad de quien desea modificar los parámetros. Se incorporaron funciones de seguridad, como autenticación de los usuarios, para que el inicio y paro de emergencia solo se activen por el personal autorizado cuando sea necesario.

La implementación en Python brinda flexibilidad y facilita el mantenimiento, permitiendo futuras actualizaciones y mejoras. Utiliza colores amigables con el usuario y muestra la información requerida cuando es solicitada.

INTRODUCCIÓN

Con el fin de optimizar y asegurar la administración precisa de medicamentos y nutrientes a pacientes, se buscan maneras de mejorar el proceso de infusión por vía intravenosa, haciéndolo más seguro y completo. Las bombas de infusión cuentan con funciones generales como el sonar una alarma en caso de error y un paro de emergencia para la situación que lo amerite, es conveniente agregar características que permitan un mejor seguimiento a las necesidades del paciente.

Una bomba de infusión de desplazamiento positivo busca tomar ventaja mecánica por un elemento actuador de bombeo, ya sea con movimiento alternativo o giratorio, para suministrar el medicamento desde la bomba hacia el paciente, se clasifican en dos grandes grupos, rotativas y reciprocantes (Jiménez Mur, 2023). Representan una parte fundamental para los hospitales, y otros centros de atención, un buen desempeño es crucial para sostener la vitalidad del paciente.

Esta iniciativa surge de la necesidad de proporcionar al personal profesional de la salud una herramienta eficiente y segura. Necesidades que se presentan a causa de la pandemia que asoló al mundo en 2020, y que desde entonces nos presenta una "nueva normalidad", en la que los procedimientos médicos se han visto modificados para estar listos ante cualquier contratiempo.

Para beneficiar a los pacientes que requieren cuidados de infusión es necesario mejorar las herramientas que existen actualmente a su disposición, la composición general de una bomba de infusión consta de algunos elementos básicos:

- Una pantalla que muestre los datos correspondientes a la cantidad de volumen suministrada en un periodo de tiempo, regularmente ml/h.
- Un botón de arranque y paro que sirva para iniciar el proceso de infusión, y que en caso de algún problema lo detenga.
- El mecanismo que permita a la bomba realizar el proceso de infusión.

CONTENIDO

PRELIMINARES
AGRADECIMIENTOS I
RESUMEN II
INTRODUCCIÓN III
CONTENIDO IV
INDICE DE FIGURAS VIII
INDICE DE TABLAS IX
CAPITULO I 12
GENERALIDADES DEL PROYECTO 12
1.1 Descripción de la empresa 13
1.1.1 Datos generales de la empresa 13
1.1.2 Misión 14
1.1.3 Visión 14
1.2 Problemas de investigación a resolver 15
1.3 Preguntas de investigación 16
1.4 Objetivos 16
1.4.1 Objetivo general 16
1.4.2 Objetivos específicos 16
1.5 Justificación de la investigación 17
CAPITULO II 18
MARCO TEÓRICO 18
2.1 Bomba de desplazamiento positivo 19
2.1.1 Descripción general 19
2.1.2 Funcionamiento 19
2.1.3 Partes que componen la bomba de infusión 20
2.2 Normas vigentes 21
2.2.1 Norma oficial mexicana NOM-001-SEDE-2012, instalaciones eléctricas (utilización). 21
2.2.2 Norma oficial mexicana NOM-004-SSA3-2012, del expediente clínico. .. 22

2.2.3 Norma oficial mexicana NOM-022-SSA3-2012, que instituye las condiciones para la administración de la terapia de infusión en los estados unidos mexicanos. 23
2.3 Sistemas dinámicos 24
2.3.1 Sistema 24
2.3.2 Sistemas de control 24
2.3.3 Sistemas en lazo abierto 25
2.3.4 Sistemas en lazo cerrado 25
2.4 Diseño de un controlador PID 25
2.5 Sensores 26
2.5.1 Sensor MAX30102 26
2.5.2 Sensor MAX6675K 27
2.6 Sensores en la investigación biomédica y la atención médica 28
2.7 Saturación del oxigeno 29
2.7.1 Monitoreo de oxígeno en la sangre 29
CAPITULO III 31
ANALISIS DE LA SITUACION ACTUAL 31
3.1 Terapia de infusión en México 32
3.1.1 Principios de operación 32
3.1.2 Clasificación de los sistemas de infusión 33
3.2 Comparación con otros sistemas de infusión 35
CAPITULO IV 36
DESARROLLO Y METODOLOGÍA 36
4.1 Alcance y enfoque de la investigación 37
4.2 Hipótesis 37
4.3 Diseño y metodología 37
4.4 Requisitos funcionales 38
4.4.1 Requisito funcional 1 38
4.4.2 Requisito funcional 2 39
4.4.3 Requisito funcional 3 40
4.4.4 Requisito funcional 4 41
4.5 Requisitos no funcionales 42

4.5.1 Requisito no funcional 1 42
4.5.2 Requisito no funcional 2 43
4.5.3 Requisito no funcional 3 44
4.5.4 Requisito no funcional 4 45
4.6 Requisitos comunes de las interfaces 46
4.6.1 Interfaz de usuario 46
4.6.2 Interfaz de hardware 46
4.7 Casos de uso 47
4.8 Ciclo de vida en V 48
4.9 Diseño arquitectónico 49
4.10 Desarrollo de la interfaz 49
4.11 Programa en Python 50
4.11.1 Diseño 51
4.11.2 Sensor de oxigenación 51
4.11.3 Sensor de temperatura 54
4.12 Iteraciones previas 57
CAPITULO V 59
RESULTADOS 59
5.1 Interfaz de usuario 60
5.2 Pestañas principales 60
5.2.1 Monitoreo 61
5.2.2 Configuración 62
5.2.3 Identificación 63
5.3 Historial 65
CAPITULO VI 67
CONCLUSIONES Y RECOMENDACIONES 67
CAPITULO VII 69
COMPETENCIAS 69
CAPITULO VIII 71
FUENTES DE INFORMACIÓN 71
CAPITULO IX 75

ANEXOS 75
Anexo 1. Asignación de Asesor(a), Comisión Revisora, Entrega de Trabajo Profesional y Dictamen 76
Anexo 2. Carta de autorización del autor para la consulta y publicación electrónica del trabajo de investigación 77
Anexo 3. Formato de licencia de uso 78
Anexo 4. Manual de usuario del sistema 79
Anexo 5. Programa de Arduino para el sensor MAX30102 83
Anexo 6. Código de Arduino para el sensor MAX6675 87
Anexo 7. Código de Python. 88

INDICE DE FIGURAS

Imagen 4.1 47
Imagen 4.2 48
Imagen 4.3 52
Imagen 4.4 53
Imagen 4.5 53
Imagen 4.6 55
Imagen 4.7 56
Imagen 4.8 57
Imagen 4.9 58
Imagen 5.1 60
Imagen 5.2 61
Imagen 5.3 62
Imagen 5.4 63
Imagen 5.5 64
Gráfico 5.1 65
Gráfico 5.2 66

INDICE DE TABLAS

Tabla 4.1 38
Tabla 4.2 40
Tabla 4.3 39
Tabla 4.4 41
Tabla 4.5 42
Tabla 4.6 43
Tabla 4.7 44
Tabla 4.8 45
Tabla 4.9 54

CAPITULO I

GENERALIDADES DEL PROYECTO

1.1 Descripción de la empresa

Teziutlán ha sido históricamente un centro de desarrollo económico en la región nororiental de Puebla, basado primero en la industria minera y metalúrgica, luego en la fruticultura y la ganadería, y más recientemente en la producción de prendas de vestir. Por supuesto, la actividad industrial siempre ha propiciado el crecimiento de varias actividades económicas como el comercio, el transporte, los servicios financieros y de forma muy específica, la educación.

Atendiendo las necesidades de la sociedad y a los principios de la Ley de Educación del Estado de Puebla, el Instituto Tecnológico Superior de Teziutlán se consolida como una institución que tiene como objetivo lograr una educación de calidad, moderna, eficiente, orientada al servicio y más cercana a las necesidades de las personas y sus intereses, promoviendo la transparencia de los recursos humanos, materiales y financieros disponibles y el uso efectivo y ejecución oportuna de sus planes de trabajo. (Instituto Tecnológico Superior de Teziutlán, s.f.).

1.1.1 Datos generales de la empresa

Instituto Tecnológico Superior de Teziutlán

Dirección

Fracción l y ll S/N, Aire Libre, Teziutlán, Puebla; C.P. 73960.

Teléfono

+52(231)3114000 / 01 / 02

Correo electrónico

web_@teziutlan.tecnm.mx

Giro de la empresa

Es una institución de educación superior pública con giro terciario.

1.1.2 Misión

Formar profesionales que se constituyan en agentes de cambio y promuevan el desarrollo integral de la sociedad, mediante la implementación de procesos académicos de calidad.

1.1.3 Visión

Llegar a ser la Institución de Educación Superior Tecnológica más reconocida en el Estado de Puebla, que ofrezca un proceso de Enseñanza – Aprendizaje certificado, comprometido con la excelencia académica y la formación integral del alumno, contribuyendo al desarrollo sustentable, económico, político y social del Estado de Puebla (Instituto Tecnológico Superior de Teziutlán, s.f.).

1.2 Problemas de investigación a resolver

Una bomba de infusión tradicional cuenta con funciones de inicio y paro, a raíz de la pandemia de COVID-19 muchas personas presentan secuelas negativas, principalmente en el sistema respiratorio. Con eso en mente, existe la necesidad de monitorear constantemente los niveles de oxigenación del paciente, especialmente durante un proceso tan delicado como lo es la infusión por vía intravenosa, ya que no todos los cuerpos reaccionan del mismo modo a las soluciones administradas.

Estos asuntos pueden ser atendidos elaborando una interfaz que se adapte a las necesidades básicas de una bomba de infusión, y que además incorpore lecturas de temperatura y saturación de oxígeno. El programa se ejecuta mediante código Python, empaquetado para que únicamente se ejecute una aplicación que funcione durante el tiempo que sea requerido, mediante una pantalla de entrada táctil, ya que cuenta con funciones que requieren de la interacción directa del usuario, no mediante teclados o botones de entrada.

Su desarrollo se lleva a cabo utilizando herramientas de programación para computadora, las cuales no necesitan de un software especializado para funcionar, por lo que cualquier computadora de gama baja es suficiente para realizar pruebas de funcionamiento y de desarrollo.

Si bien la pandemia terminó y la sociedad adoptó una nueva normalidad, es evidente que muchos centros de salud no se encuentran preparados para un aumento en el volumen de pacientes, y es bien sabido que en ocasiones las personas no reciben la atención medica que requieren en el momento que se necesita, se busca que la interfaz implementada en la bomba de infusión junto con sus sensores pueda contrarrestar esta tendencia.

1.3 Preguntas de investigación

¿Qué parámetros se establecen para controlar una bomba de infusión?

¿Cómo controlar el mecanismo de una bomba de infusión para obtener el mejor desempeño?

¿Cuál es la mejor manera de llevar a cabo el proceso de control en una bomba de infusión?

1.4 Objetivos

1.4.1 Objetivo general

Controlar el funcionamiento de una bomba de desplazamiento positivo a través de una interfaz gráfica de usuario, para mejorar el desempeño y accesibilidad del dispositivo de infusión, aplicando herramientas tecnológicas de simulación y control.

1.4.2 Objetivos específicos

Analizar el funcionamiento de una bomba de infusión de desplazamiento positivo para obtener los parameros de desempeño del dispositivo y la dinámica del sistema.

Proponer un sistema de control electrónico para mejorar el proceso de infusión que permita suministrar cantidades precisas de cualquier solución.

Implementar en la bomba de infusión una interfaz gráfica de usuario que permita facilitar el control y uso del dispositivo.

1.5 Justificación de la investigación

En una bomba de infusión es necesario establecer una correcta comunicación entre sus tres pilares: mecanismo, sistema de arranque y paro, pantalla. Con eso en mente, se busca el mejor desempeño de la bomba de infusión, con un diseño de interfaz simplificado que deje poco espacio a las confusiones causadas por la introducción errónea de datos de suministro de medicamentos, ya sea por el tiempo o por la unidad de medida del volumen requerido.

La interfaz tiene el propósito de ofrecer a los usuarios la capacidad de ajustar parámetros cruciales, como la velocidad de infusión, y vigilar en tiempo real el avance de la administración de fluidos, así como otros datos capturados con sensores. Se requiere respaldar por algoritmos avanzados, el sistema de control busca garantizar una administración precisa durante el tratamiento, cumpliendo rigurosamente con las prescripciones médicas.

El buen funcionamiento depende de una correcta capacitación al personal médico y de los mismos usuarios, ya que una confusión durante el proceso puede ser causa de un error fatal, los más comunes siendo la velocidad de aplicación y cantidad inadecuada del medicamento. Por ello, la presente investigación busca encontrar la mejor opcion para implementar un sistema de control fácil de entender, y mantener, para poder realizar el proceso de infusión de forma segura y eficaz.

Este proyecto no solo se enfoca en la funcionalidad práctica, sino que también incorpora elementos de seguridad, que busca evitar el uso indebido del aparato, y, en consecuencia, evitar el riesgo y disminuir los errores durante la infusión. La implementación en Python no solo proporcionará flexibilidad, sino que también establece una base sólida para futuras actualizaciones y mejoras en la interfaz y el sistema de control, contribuyendo así al avance continuo de la atención médica.

CAPITULO II

MARCO TEÓRICO

2.1 Bomba de desplazamiento positivo

2.1.1 Descripción general

Las bombas de infusión médicas suministran soluciones, por ejemplo: medicamentos, nutrientes alimentarios, productos químicos, entre otros, a pacientes en diversas condiciones físicas, de forma controlada y continua (University of Pennsylvania, 2015).

Se constituyen a partir de una unidad principal con pantalla y controlador, y un sistema de infusión formado por una jeringa o un conjunto de tubos conectados al paciente. Proporciona medicamentos en cantidades definidas con precisión y a un ritmo definido según sea requerido (Díaz, 2023).

2.1.2 Funcionamiento

Las bombas de infusión funcionan mediante el uso de sistemas mecánicos o electrónicos para regular la administración de medicamentos o líquidos controlados a un paciente. Las principales instrucciones de uso de la bomba de infusión son las siguientes:

Configuración: se establecen parámetros de infusión, entre ellos, la velocidad de infusión, la cantidad de líquido total de infusión y el tiempo de infusión.

Carga de medicamentos: los medicamentos se colocan en el sistema de infusión con bomba, que puede ser administrado por jeringa o un conjunto de tubos que se conectan a un recipiente de medicamento.

Conexión del paciente: el sistema de infusión se conecta al paciente mediante una cánula, catéter o aguja, según el método de administración.

Programación: utilizando la pantalla de la unidad, se configura la bomba para llevar a cabo la infusión necesaria, se ajustan parámetros como la tasa de infusión y la dosis deseada.

Iniciar infusión: una vez configurados los parámetros, la bomba se enciende y comienza a administrar el medicamento o líquido al paciente.

Monitoreo y ajuste: durante todo el proceso, la bomba de infusión monitorea continuamente la velocidad de infusión, el tiempo transcurrido y otros parámetros relevantes. Si se detectan discrepancias o desviaciones de la norma, la bomba hace sonar una alarma para alertar al personal médico que puede realizar los ajustes necesarios.

Infusión completa: cuando se alcanza el volumen total establecido o se completa el tiempo de infusión establecido, la bomba se detiene automáticamente (Díaz, 2023).

2.1.3 Partes que componen la bomba de infusión

Una bomba de infusión consta de varios componentes que funcionan juntos, con el fin de suministrar soluciones líquidas y otros medicamentos de manera controlada. Las partes principales de un mecanismo de infusión se presentan a continuación:

Unidad principal: El elemento principal de la bomba, que contiene las partes vitales para el funcionamiento, como la electrónica, el sistema control y la pantalla. Permite que el usuario configure y controle los parámetros de infusión y el monitoreo durante el proceso.

Sistema de infusión: Es un conjunto de componentes usado para administrar medicamento o una solución desde una bomba a un paciente. Puede constar de una jeringa de un solo uso, un conjunto conectado de tubos, o bien, una combinación de los dos mencionados.

Conector del paciente: Es el medio entre el sistema de infusión y el paciente. Puede ser una cánula, un catéter o una aguja, según la vía en que se administra y las necesidades clínicas.

Pantalla de control: La pantalla permite visualizar información esencial como la tasa de infusión, el volumen requerido y las alertas. Estos controles permiten que el usuario ingrese parámetros de administración y los ajuste según las necesidades.

Fuente de alimentación: Para su funcionamiento, la bomba de infusión necesita una fuente de alimentación, la cual puede ser una batería recargable o una conexión a la corriente.

Además de estos componentes principales, una bomba de infusión puede tener otras características como conectividad inalámbrica, alertas, sensores de diferentes tipos o puertos de datos para comunicarse con sistemas de información hospitalarios (Díaz, 2023).

2.2 Normas vigentes

Los equipos médicos cuentan con una serie de normas que buscan garantizar la calidad de los aparatos y la seguridad del paciente, ya que, al tratarse de dispositivos destinados a preservar la vida humana, los estándares de calidad deben ser los más altos, de lo contrario puede resultar en pérdidas económicas para el sector hospitalario, o bien, pérdidas humanas.

2.2.1 Norma oficial mexicana NOM-001-SEDE-2012, instalaciones eléctricas (utilización).

Esta norma regula, entre muchas otras cosas, las instalaciones en centros hospitalarios y de salud en su artículo 517, en el que destacan las siguientes directrices:

El sistema eléctrico básico de un hospital debe consistir en dos sistemas independientes que puedan proporcionar energía para iluminación y servicios eléctricos vitales para la seguridad humana y el funcionamiento seguro y eficiente del hospital, si el servicio eléctrico de uso cotidiano se interrumpe por cualquier motivo. Estos dos sistemas deben ser el sistema de emergencia y el sistema de equipamiento. Los sistemas de emergencia deben limitarse a los circuitos necesarios para la seguridad humana y la atención de pacientes críticos. Estas están etiquetadas como cadenas derivadas de seguridad humana y cadenas derivadas críticas.

Considere la siguiente lista de equipos médicos, entre otros: desfibriladores, incubadoras, sistemas de infusión rápida, etc. Si se interrumpe el suministro de energía normal o se produce una falla interna en el sistema eléctrico, el circuito del seguro de vida cambia automáticamente a energía de respaldo en 10 segundos o menos (SENER, 2019).

2.2.2 Norma oficial mexicana NOM-004-SSA3-2012, del expediente clínico.

La revisión y actualización de esta norma, tiene como propósito establecer con precisión los criterios científicos, éticos, tecnológicos y administrativos obligatorios en la elaboración, integración, uso, manejo, archivo, conservación, propiedad, titularidad y confidencialidad del expediente clínico, el cual se constituye en una herramienta de uso obligatorio para el personal del área de la salud, de los sectores público, social y privado que integran el Sistema Nacional de Salud (Secretaría de Salud, 2012). Todo expediente clínico, deberá tener los siguientes datos generales:

- Tipo, domicilio y nombre del establecimiento y en su caso, nombre de la institución a la que pertenece;
- En su caso, la razón y denominación social del propietario o concesionario;
- Nombre, edad, sexo, y domicilio del paciente; y
- Los demás que señalen las disposiciones sanitarias.

2.2.3 Norma oficial mexicana NOM-022-SSA3-2012, que instituye las condiciones para la administración de la terapia de infusión en los estados unidos mexicanos.

Históricamente la terapia de infusión intravenosa ha contribuido, de manera importante, en el desarrollo de mejores tratamientos para la atención a la salud. Sin embargo, este procedimiento también ofrece serios riesgos para los pacientes, para el personal y para las instituciones prestadoras de servicios de salud, en virtud de que se ve incrementada la estancia hospitalaria (días camas) y el gasto por las complicaciones adyacentes (SEGOB, 2012).

Según la norma oficial, el equipo de administración de la terapia de infusión deberá cambiarse cada 72 horas si existe sospecha de contaminación o infección sistémica asociada a un catéter central o periférico, se procederá al retiro inmediato. Entre las regulaciones principales de esta norma de manera general se encuentran:

- Características de los Insumos
- Contenedores para las soluciones intravenosas
- Uso de circuitos intravenosos
- Catéteres
- Preparación de la piel
- Fijación del catéter
- Consideraciones sobre la terapia de infusión
- Consideración sobre el paciente
- Selección e integración de material y equipo
- Administración de la solución intravenosa
- Mantenimiento de la terapia de infusión intravenosa
- Cambio del sitio de inserción del catéter venoso periférico

2.3 Sistemas dinámicos

2.3.1 Sistema

Es un conjunto de componentes que trabajan juntos buscando lograr un objetivo específico. Se le llama componente a la unidad de un sistema que es específica de su función. El concepto de sistemas no se limita de ninguna manera a los sistemas físicos, sino que también puede aplicarse a fenómenos dinámicos abstractos como el transporte, el crecimiento de la población, la economía, y la biología (Ogata, 1996).

Del mismo modo, el autor Bolton, (Bolton, 2013) menciona que un sistema puede interpretarse como una caja con una entrada y una salida de contenido irrelevante, cuya importancia radica en la interacción entre la entrada y la salida. El ejemplo que propone dice que un motor se puede considerar como un sistema, en el cual su entrada es la alimentación de energía eléctrica y su salida es un eje rotando.

2.3.2 Sistemas de control

Actualmente existen diversos tipos de control en diferentes sistemas, dentro de los más importantes podemos destacar los siguientes: mecánicos, eléctricos, neumáticos, hidráulicos y térmicos.

Se puede inferir que existen muchos sistemas los cuales están presentes en nuestra vida cotidiana, tan solo el estar vivo es un acto de ello, según (Bolton, 2013) quien hace alusión al cuerpo humano, regula su temperatura de forma constante, a lo que podría referirse como un sistema de control de temperatura.

2.3.3 Sistemas en lazo abierto

Los sistemas cuyas salidas no afectan la operación de control se denominan sistemas de control de red abierta. En otras palabras, en un sistema de control de red abierta, la salida no se compara con la entrada por lo que no ajusta sus parámetros con el paso del tiempo. Un ejemplo práctico es una lavadora, el remojo, lavado y aclarado en lavadora se realizan según el tiempo. El aparato no mide la señal de salida, es decir, que tan limpia sale la ropa. (Ogata, 1996)

2.3.4 Sistemas en lazo cerrado

Los sistemas de control que comparan la salida con la entrada, es decir, se retroalimentan, son llamados frecuentemente sistemas de control de lazo cerrado. En la práctica, los términos control realimentado y control de malla cerrada son intercambiables. En un sistema de control de malla cerrada la señal de error, la cual es diferenciada entre la señal de entrada y la señal realimentada, se alimenta al controlador de modo que se reduzca el error y lleve la salida del sistema a un valor deseado (Ogata, 1996).

2.4 Diseño de un controlador PID

Los controladores ahora se utilizan comúnmente en todo tipo de empresa. Las técnicas de fabricación a gran escala implican que los compensadores no se implementan para una función específica, sino que son dispositivos universales que pueden adaptarse a tareas específicas según las necesidades del usuario.

El Controlador Derivado Integral Proporcional (Controlador PID) es un dispositivo de control universal que requiere que el diseñador especifique valores apropiados para los diversos parámetros que contiene según la situación. Por tanto, se elimina la necesidad de implementar o fabricar un compensador. Como sugiere el nombre, un controlador PID es un caso especial de un compensador de adelanto-retraso, donde el compensador de adelanto-retraso es un compensador proporcional-diferencial y

el compensador de adelanto-retraso es un compensador proporcional-integral. A partir del producto de dos compensadores se obtiene un regulador con dos ceros, un polo en el origen y una ganancia.

Por tanto, el controlador PID cuenta con tres parámetros de libre elección: la posición de los dos puntos cero y el valor de ganancia. En el análisis y diseño empírico de sistemas de control, estos métodos pueden probarse porque, además de la capacidad de manipular señales de entrada, también muestran la relación entre las características de respuesta del sistema a señales de entrada comunes y el orden del sistema. entrada real. Las señales de prueba utilizadas para crear controladores incluyen: paso, rampa, parábola, pulso y otras funciones.

2.5 Sensores

Son herramientas que detectan y responden a algún tipo de información del entorno físico. Muchos tipos de sensores se utilizan diariamente y se clasifican según la cantidad y calidad de la detección.

Algunos ejemplos son sensores de corriente, magnéticos o de radio, de humedad, de velocidad o flujo de fluidos, ópticos, de posición, de presión, de calor o temperatura, ambientales y sensores químicos. (NIBIB, 2022).

2.5.1 Sensor MAX30102

Es un dispositivo que integra un pulsioxímetro y un pulsómetro. Dispone de radiación infrarroja, fotodetectores, circuitos electrónicos de baja frecuencia que logran suprimir la luz ambiental y componentes ópticos. Admite la interfaz de comunicación I2C para transferir fácilmente información a Arduino u otros microcontroladores de pulso y oxígeno. El chip puede apagar el módulo o entrar en modo de suspensión. Coloque el dispositivo en su dedo, pulmón o muñeca para leer su frecuencia cardíaca y/o ritmo cardíaco (Unit Electronics, s.f.).

¿Para qué sirve el sensor de pulso MAX30102?

Es ideal para integrar en proyectos que miden pulso y frecuencia cardíaca porque funciona de la siguiente manera:

- Método de resolución fotográfica: mide el pulso y el nivel de oxígeno en la sangre a través del tejido humano. Cuando los vasos sanguíneos laten, el tejido humano producirá una transmitancia de luz diferente.
- Fuente de luz: luz de una longitud de onda específica emitida por un diodo utilizado para medir el hemo oxigenado (HbO2) y la hemoglobina (Hb) en la sangre arterial.
- Transmitancia convertida en señal eléctrica: los cambios en la cantidad de pulso arterial provocan cambios en la transmitancia de la luz. En este punto, la luz reflejada por el tejido humano es captada por un sensor fotoeléctrico y convertida en una señal eléctrica. (Unit Electronics, s.f.).

2.5.2 Sensor MAX6675K

Un termopar es un sensor hecho de una combinación de dos metales que crea una diferencia de potencial dependiendo de la temperatura, pero esta diferencia de potencial es demasiado pequeña si desea utilizar un microcontrolador.

Aquí es donde entra en juego el MAX6675 IC, ya que el MAX6675 contiene elementos electrónicos necesarios para amplificar, compensar y convertir las señales analógicas en señales digitales que suministran un termopar al sensor del instrumento. (Unit Electronics).

El módulo de interfaz MAX6675 para termopares tipo K acondiciona la señal analógica proporcionada por el sensor tipo termopar para que el microcontrolador pueda leer la señal digitalmente.

Simplemente se debe conectar el módulo para que funcione, ya que cuenta con las especificaciones necesarias para realizar las operaciones previamente mencionadas

(amplificar, compensar y convertir el voltaje) por lo que establecer comunicación a un microcontrolador es relativamente fácil.

Una desventaja que presenta este circuito es que sólo está disponible en un paquete SOIC, por lo que no es tan fácil de usar en una placa de pruebas.

El módulo de interfaz de termopar tipo K es una gran manera de iniciar con la medición de altas temperaturas con cualquier microcontrolador, ya que incluye el MAX6675 y el termopar tipo K en una conveniente placa de conexión para que pueda comenzar a experimentar lo antes posible (Geek Factory, s.f.).

2.6 Sensores en la investigación biomédica y la atención médica

En investigación médica y biomédica, son necesarios los sensores de todo tipo para detectar elementos fisiológicos específicos, y luego transmitir o informar estos datos cuantitativos al personal calificado del área de la salud.

Los termómetros convierten la expansión de un líquido o la curvatura de una tira metálica cuando se calienta en un valor correspondiente a la temperatura corporal.

Las nuevas tecnologías portátiles, como los relojes inteligentes, contienen sensores que monitorean, analizan y transmiten información sobre la frecuencia cardíaca y las horas de sueño. Los investigadores utilizan estos dispositivos para controlar y monitorear el bienestar de las personas, detectando patrones para intentar adelantarse y potencialmente intervenir para prevenir problemas de salud graves como accidentes cerebrovasculares o enfermedades cardíacas.

Un oxímetro de pulso mide los cambios del cuerpo en la absorción de tipos específicos de luz para medir la frecuencia cardíaca y la saturación de oxígeno. Estos sensores se utilizan regularmente en hospitales y clínicas, aunque también es posible adquirir uno para uso doméstico. (NIBIB, 2022).

2.7 Saturación del oxigeno

En nuestro cuerpo, la sangre se encarga de llevar oxígeno saliendo de los pulmones a cualquier parte del cuerpo. Nuestra sangre se compone mayormente por agua. La manera más simple de mover oxígeno por un líquido es disolver el oxígeno en el líquido, es decir, una mezcla homogénea. Como sea, debido a nuestra naturaleza activa, demandamos enormes necesidades metabólicas, por lo que es imposible disolver suficiente oxígeno en el cuerpo para cumplir nuestras necesidades biológicas. Por ello, el cuerpo humano desarrolló un método para mejorar la capacidad de nuestra sangre para transportar oxígeno. Este método inteligente consiste en utilizar una molécula especial, llamada hemoglobina. Cada partícula de hemoglobina consta de cuatro unidades hemo individuales, cada una de las cuales requiere un átomo de hierro. Una molécula de hemo completa es capaz de mover cuatro partículas de oxígeno, una por cada grupo hemo. El hierro se encarga de producir el color rojo en la molécula de hemo, del mismo modo que el color rojo del suelo o las rocas se debe a las propiedades del hierro. El hemo en la sangre no fluye libremente, sino que está empaquetado en las células. El tono rojizo de la hemoglobina da a las células un aspecto rojo, por eso se les llama glóbulos rojos. Los cuales se encuentran suspendidos en una base acuosa de sangre llamada plasma, la cual contiene otros tipos de proteínas, células y electrolitos. Si se eliminaran todos los glóbulos rojos de la sangre, el plasma restante ya no sería rojo, sino sólo un líquido claro o de tonalidad pajiza (CardiacSense, s.f.).

2.7.1 Monitoreo de oxígeno en la sangre

Las personas pueden hacerse análisis para ver su nivel de oxígeno en la sangre. Esta prueba se llama análisis de gases en sangre arterial. No obstante, esto requiere una muestra de sangre real de una arteria con una aguja conectada a una jeringa. Aunque se puede realizar y se utiliza ampliamente en entornos médicos, es inconveniente y se puede realizar ocasionalmente. Una forma más sencilla de

averiguar los niveles de oxígeno en sangre sin tomar una muestra de sangre es utilizar las propiedades de cambio de color de la molécula de hemoglobina.

Es sabido que la molécula de hemoglobina totalmente cargada con las cuatro partículas de oxígeno tendrá un tono rojo brillante, pero una molécula de hemoglobina sin oxígeno tendrá un tono más fuerte y opaco, por lo que simplemente se puede hacer un dispositivo para medir la intensidad de esta luz. Las moléculas de hemo se reflejan en el cuerpo. Si toda la luz es de color rojo brillante, entonces se sabe que el hemo está oxigenado. Cuanto más oscuro es el color, más vacía está la molécula de hemo.

Esto se puede medir e interpretar con un porcentaje simple, el 100% está en una luz roja fuerte, lo que significa que las partículas de hemoglobina están totalmente cargadas de oxígeno; el 0% representa un color rojo obscuro, lo que significa que las moléculas de hemo presentan una ausencia de oxígeno. A este número se le conoce como porcentaje de saturación de oxígeno, a menudo abreviado como SpO2. Este dispositivo se llama monitor de saturación de oxígeno o monitor de SpO2.

Hoy en día, los monitores de saturación de oxígeno como este en su mayoría también pueden medir la frecuencia cardíaca. Los monitores de oximetría de pulso combinados a menudo se denominan oxímetros de pulso u oxímetros de pulso. Estos monitores de oxígeno en sangre funcionan iluminando nuestro cuerpo con una pequeña luz y midiendo la luz reflejada (generalmente desde la punta del dedo). (CardiacSense, s.f.).

CAPITULO III

ANALISIS DE LA SITUACION ACTUAL

3.1 Terapia de infusión en México

Los pacientes a menudo requieren soluciones o medicamentos que se administran en volúmenes muy precisos y a una velocidad de infusión específica (por hora, por día o en un volumen fijo). La precisión y uniformidad que se requiere para este proceso la proporciona un sistema de infusión que controla de manera electrónica o mecánica la velocidad de infusión de la solución o fármaco. Los dispositivos de infusión y sus tubos o "juegos" desechables se utilizan para administrar líquidos o soluciones a pacientes por distintas vías, como puede ser de forma subcutánea, intravenosa, epidural, enteral o parenteral (CENETEC, 2004).

El sistema de infusión consta de al menos dos componentes:

1. Un mecanismo que suministra medicamentos o soluciones.
2. Un mecanismo o dispositivo para controlar la velocidad de infusión.

3.1.1 Principios de operación

En el momento cuando administramos un líquido o solución a un paciente mediante un acceso para hemodiálisis, creamos un sistema formado por una línea o "dispositivo" de infusión y los vasos sanguíneos de la persona atendida. Este sistema satisface la relación P = RF, donde P es la presión requerida para producir un flujo F, superando una resistencia R. La resistencia al flujo depende de:

- El tamaño del catéter o aguja de infusión.
- La distancia de los tubos y forma geométrica del "conjunto" de infusión.
- La resistencia en el lecho vascular o intrafascial.
- Si existe un filtro de aire o de partículas.
- El lugar de infusión.
- Resistencia al flujo del líquido inyectado.
- Longitud de la tubería.

La resistencia al flujo sanguíneo es mayor en los adultos que en los recién nacidos. Por lo tanto, la presión que normalmente se requiere para asegurar una buena infusión en recién nacidos es menor que la requerida en adultos.

En pacientes neonatales, las velocidades de perfusión deben estar entre 0,1 y 99,9 ml/hora (microinfusión). Las mini bombas de infusión suelen ser iguales que las bombas de infusión universales, con la principal diferencia de administrar un menor caudal. Habitualmente, esta herramienta permite programar la cantidad de líquido a administrar y, si no se ha utilizado el fármaco, sonará una alarma cuando se alcance este volumen. Incluso después de alcanzar este volumen, mantienen la velocidad de infusión del fármaco muy baja para evitar la obstrucción de la aguja. A este flujo se le conoce para "mantener la vena abierta" (KVO, "*keep vein open*").

3.1.2 Clasificación de los sistemas de infusión

Es posible categorizar los sistemas de infusión según la forma en que se entrega el líquido al paciente:

Control mediante presión: El sistema de bomba genera presión constantemente y el caudal llega a variar dependiendo de la resistencia.

Controlado por la gravedad: su diseño está orientado para situaciones de bajo riesgo, incluidos tratamientos de flujos alternativos, donde una presión de suministro baja es suficiente para mantener el caudal seleccionado. Un ejemplo típico es un controlador de riego por goteo, que normalmente no tiene sistema de bombeo y se basa en "kits" de soluciones estándar. El caudal requerido se establece en goteo por minuto y se controla mediante una válvula de cierre de línea.

Regulación activa de la presión. También llamados infusores, estos dispositivos crean una presión regular y en ocasiones tienen sensores de caída; los modelos más nuevos incluyen sistemas de modo de flujo que proporcionan una representación visual de la resistencia al flujo.

Controlado por volumen. En estos dispositivos, el volumen (caudal) varía y permanece constante. La presión alcanzada varía con respecto a la resistencia. Se incluyen en este grupo:

Bombas peristálticas: funcionan presionando una bolsa o tubo flexible para crear movimiento de fluido en un recipiente. Hay dos tipos en esta clasificación: bombas peristálticas lineales y bombas rotativas. Una bomba peristáltica lineal tiene una serie de discos en forma de dedos que comprimen el tubo en una onda en movimiento continuo, forzando el líquido desde el depósito hacia el paciente. Una bomba peristáltica rotativa utiliza un rotor para forzar el fluido a través de un rodillo a través de un canal semicircular hacia un tubo.

Bombas de cartucho o de pistón: En estas bombas, el "dispositivo" de infusión o parte de la bomba de cartucho contiene una o más cámaras, y la acción de uno o más pistones mueve todo lo que hay en su interior. Contiene líquido para permitir el movimiento del fluido. El caudal depende de la distancia recorrida y la velocidad de movimiento del pistón.

Bomba de inyección: Se prefieren cuando se requieren pequeñas cantidades y caudales bajos. Estas bombas mueven el émbolo de la inyección a un ritmo controlado y así administrar el medicamento al paciente. La tasa de administración puede ser una administración en bolo continua o gradual durante un período de tiempo. Inserte la jeringa en la bomba e instale el émbolo en el soporte del émbolo, la jeringa se vacía cuando se empuja hacia adelante el soporte del émbolo.

Bomba que controla la analgesia del paciente: En contraste de las bombas de infusión de uso mayor, estas bombas dejan que los pacientes puedan autoadministrarse dosis en bolo. Es especialmente importante que estos aparatos cuenten con algún mecanismo que presente resistencia al libre flujo. Pueden ser de un solo uso o reutilizables. Las bombas desechables funcionan con baterías u otros mecanismos neumáticos o flexibles. Dependiendo de las necesidades del paciente

se puede programar de diferentes formas: bolo a demanda, dosis de carga, infusión continua, infusión continua mediante bolo a demanda y concentración del fármaco.

Bomba de anestesia: Mecanismos de inyección diseñados para sedar o anestesiar. El diseño permite manipular o ajustar el caudal durante la infusión, proporcionan velocidades de infusión más altas que las bombas de jeringa estándar y pueden administrar dosis en bolo, lo que significa que pueden administrar dosis de inducción más rápido y más oportunamente. Entre sus funciones importantes se incluyen la programación de la concentración basada en la masa total del paciente y la calibración automática de la bomba mediante la identificación del medio de infusión específico. (CENETEC, 2004).

3.2 Comparación con otros sistemas de infusión

Existen diversos tipos de sistemas y bombas de infusión disponibles en el mercado, esto provoca que la innovación sea un tema necesario para sobresalir entre los competidores, es por ello por lo que se busca que este sistema cubra las necesidades y cumpla con funciones que otros sistemas no pueden realizar.

La propuesta de este equipo busca no solo administrar un tratamiento de infusión al paciente, también monitorear su temperatura y saturación de oxígeno mediante sensores, mostrar la información en tiempo real en la interfaz y guardar un historial de los tres procesos que realiza al mismo tiempo, dicho historial se pretende sea accesible durante cualquier momento del proceso.

De este modo se presenta una adición al funcionamiento tradicional de una bomba de infusión genérica, lo que busca facilitar el uso y aplicación de la misma para todos los involucrados.

CAPITULO IV

DESARROLLO Y METODOLOGÍA

4.1 Alcance y enfoque de la investigación

La trascendencia de esta investigación radica en presentar una interfaz acoplada a un sistema de control para ser aplicado en una bomba de infusión de desplazamiento positivo, buscando tener un impacto en el sector médico en el área de terapias de infusión con una interfaz fácil de usar y de entender, que monitoree los signos vitales en tiempo real. Todo esto por las condiciones propias del diseño, y que la metodología desarrollada en esta investigación propone un orden de acción y toma de decisión efectiva adaptable a otras exigencias y necesidades.

4.2 Hipótesis

Incluir una interfaz simplificada que muestre datos continuos y un historial de administración en una bomba de infusión, junto con sensores de temperatura y saturación del oxígeno, otorgará un mejor proceso y control de la infusión en pacientes de la región de Teziutlán.

4.3 Diseño y metodología

- Investigar el funcionamiento de una bomba de desplazamiento positivo.
- Elegir el lenguaje de programación adecuado para el diseño de una interfaz que permita asignar los valores con los que debe trabajar el sistema.
- Programar una interfaz fácil de usar y de entender para el control del sistema y registro de información.
- Determinar la función de transferencia del sistema.
- Realizar pruebas para encontrar el estilo de control que mejor se adapte a las necesidades.
- Implementar el modelo de control al mecanismo de la bomba de infusión.
- Comprobar el correcto funcionamiento de la bomba de infusión.

4.4 Requisitos funcionales

4.4.1 Requisito funcional 1

- Disponibilidad de información: Los usuarios podrán consultar en cualquier momento el estado de los signos vitales del paciente.
    a. La información disponible del sistema podrá ser consultada por cualquier usuario en cualquier momento (signos vitales).

Tabla 4.1

Requisito funcional 01

Identificación del requerimiento:	RF01
Nombre del requerimiento:	Disponibilidad de información.
Características:	Los usuarios podrán consultar en cualquier momento el estado de los signos vitales del paciente.
Descripción del requerimiento:	La información disponible del sistema podrá ser consultada por cualquier usuario en cualquier momento (signos vitales).
Requerimiento NO funcional:	RNF01 RNF02 RNF03 RNF04
Prioridad del requerimiento: Alta	

Nota. Descripción del requisito funcional 01.

4.4.2 Requisito funcional 2

- Autentificación de usuarios: Los usuarios deberán identificarse para acceder a las zonas restringidas de la aplicación del sistema.
    a. El sistema podrá ser consultado y modificado por los usuarios validados.

Tabla 4.2

Requisito funcional 02

Identificación del requerimiento:	RF02
Nombre del requerimiento:	Autentificación de usuarios.
Características:	Los usuarios deberán identificarse para acceder a las zonas restringidas de la aplicación del sistema.
Descripción del requerimiento:	El sistema podrá ser consultado y modificado por los usuarios validados.
Requerimiento NO funcional:	RNF01 RNF02 RNF03
Prioridad del requerimiento: Alta	

Nota. Descripción del requisito funcional 02.

4.4.3 Requisito funcional 3

- Gestión de datos: Permite generar un historial gráfico para cada signo vital del paciente.
    a. El usuario puede generar una gráfica con los datos guardados del proceso de infusión llevado a cabo para cada paciente.

Tabla 4.3

Requisito funcional 03

Identificación del requerimiento:	RF03
Nombre del requerimiento:	Gestión de datos.
Características:	Permite generar un historial gráfico para cada signo vital del paciente.
Descripción del requerimiento:	El usuario puede generar una gráfica con los datos guardados del proceso de infusión llevado a cabo para cada paciente.
Requerimiento NO funcional:	RNF01 RNF02 RNF03
Prioridad del requerimiento: Alta	

Nota. Descripción del requisito funcional 03.

4.4.4 Requisito funcional 4

- Integración de componentes: El sistema tendrá una gestión administrativa y médica.
    a. Los datos generados para cada paciente deben ser almacenados y estar disponibles para el personal administrativo, con el fin de generar un historial médico en caso de ser necesario.

Tabla 4.4

Requisito funcional 04

Identificación del requerimiento:	RF04
Nombre del requerimiento:	Integración de componentes.
Características:	El sistema tendrá una gestión administrativa y médica.
Descripción del requerimiento:	Los datos generados para cada paciente deben ser almacenados y estar disponibles para el personal administrativo, con el fin de generar un historial médico en caso de ser necesario.
Requerimiento NO funcional:	RNF01 RNF02 RNF03 RNF04
Prioridad del requerimiento: Alta	

Nota. Descripción del requisito funcional 04.

4.5 Requisitos no funcionales

4.5.1 Requisito no funcional 1

- Interfaz del sistema: El sistema deberá mostrar una interfaz de usuario fácil de usar y de entender.
    a) El sistema incluye una interfaz intuitiva y amigable con el usuario.

Tabla 4.5

Requisito no funcional 01

Identificación del requerimiento:	RNF01
Nombre del requerimiento:	Interfaz del sistema.
Características:	El sistema deberá mostrar una interfaz de usuario fácil de usar y de entender.
Descripción del requerimiento:	El sistema incluye una interfaz intuitiva y amigable con el usuario.
Prioridad del requerimiento: Alta	

Nota. Descripción del requisito no funcional 01.

4.5.2 Requisito no funcional 2

- Mantenimiento: El sistema deberá incluir instrucciones claras para facilitar los mantenimientos que serán realizados por el personal técnico.
    a) El sistema debe permitir realizar operaciones de mantenimiento con el menor esfuerzo posible.

Tabla 4.6

Requisito no funcional 02

Identificación del requerimiento:	RNF02
Nombre del requerimiento:	Mantenimiento.
Características:	El sistema deberá incluir instrucciones claras para facilitar los mantenimientos que serán realizados por el personal técnico.
Descripción del requerimiento:	El sistema debe permitir realizar operaciones de mantenimiento con el menor esfuerzo posible.
Prioridad del requerimiento: Alta	

Nota. Descripción del requisito no funcional 02.

4.5.3 Requisito no funcional 3

- Desempeño: El sistema deberá garantizar el uso correcto de los datos almacenados, en cuanto a disponibilidad y confidencialidad.
    a) Garantizar el desempeño del sistema para todos los usuarios, los registros e información almacenada debe estar disponible para consulta o actualización.

Tabla 4.7

Requisito no funcional 03

Identificación del requerimiento:	RNF03
Nombre del requerimiento:	Desempeño.
Características:	El sistema deberá garantizar el uso correcto de los datos almacenados, en cuanto a disponibilidad y confidencialidad.
Descripción del requerimiento:	Garantizar el desempeño del sistema para todos los usuarios, los registros e información almacenada debe estar disponible para consulta o actualización.
Prioridad del requerimiento: Alta	

Nota. Descripción del requisito no funcional 03.

4.5.4 Requisito no funcional 4

- Apego a las normas oficiales: El sistema deberá funcionar conforme a la norma oficial mexicana.
    a) El sistema debe cumplir con los requisitos de las normas oficiales mexicanas para la administración de terapia de infusión.

Tabla 4.8

Requisito no funcional 04

Identificación del requerimiento:	RNF04
Nombre del requerimiento:	Apego a las normas oficiales.
Características:	El sistema deberá funcionar conforme a la norma oficial mexicana.
Descripción del requerimiento:	El sistema debe cumplir con los requisitos de las normas oficiales mexicanas para la administración de terapia de infusión.
Prioridad del requerimiento: Alta	

Nota. Descripción del requisito no funcional 04.

4.6 Requisitos comunes de las interfaces

4.6.1 Interfaz de usuario

La interfaz para el usuario requiere mostrar ventanas con botones, campos de entradas de texto, listas despegables y pestañas. Se desarrollará de manera específica para el sistema, deberá mostrar todos los elementos a medir dentro del sistema y será visualizada desde la pantalla integrada en la bomba de infusión.

4.6.2 Interfaz de hardware

Los siguientes elementos serán necesarios para mostrar la información y ejecutar el programa y sistema de control.

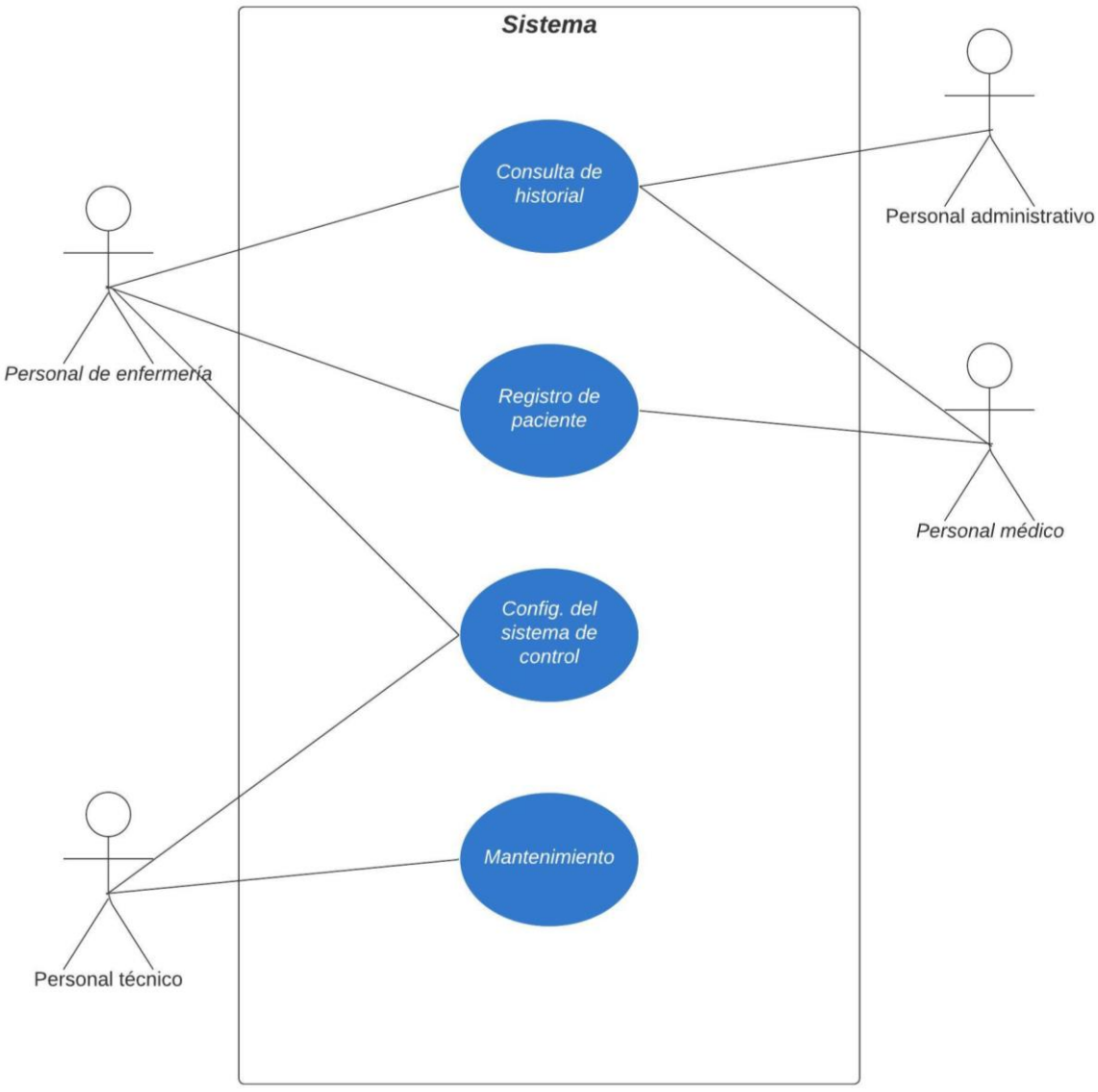
- Raspberry pi pico
- Arduino nano
- Pantalla *touch* para microcontrolador *Raspberry Pi*

4.7 Casos de uso

Los diagramas de casos de uso son diagramas de comportamiento del lenguaje de modelado unificado que se utilizan para representar sistemas y procesos de programación orientados a objetos. En este diagrama, todos los objetos involucrados están estructurados y relacionados entre sí (IONOS, 2020). El diagrama de casos de uso para este proyecto se muestra en la imagen 4.1.

Imagen 4.1

Diagrama de casos de uso



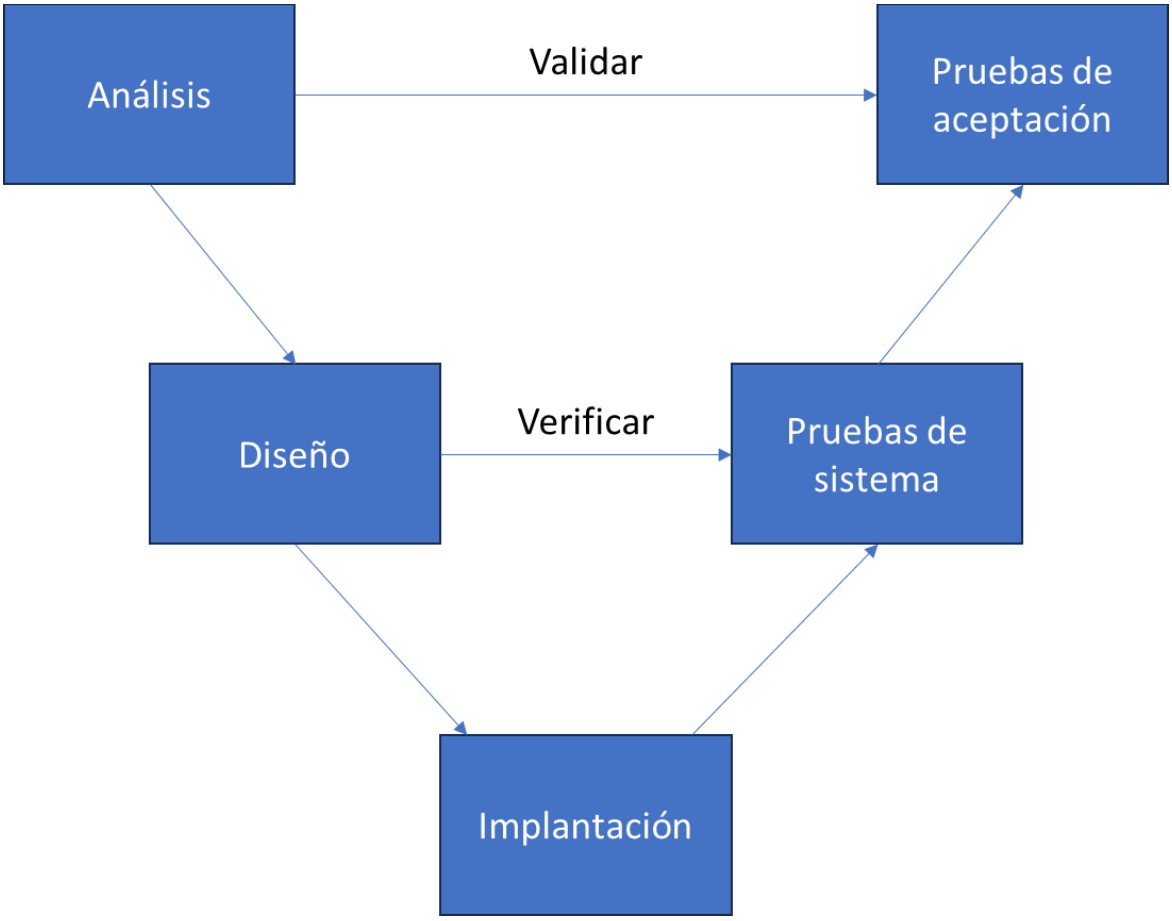
Nota. Descripción de los casos de uso aplicables en el proyecto [Fotografía] 2023.

4.8 Ciclo de vida en V

El modelo V, o ciclo V, es un estilo de desarrollo de software que divide el proceso en tres partes: diseño, implementación, integración y pruebas de calificación. La letra V es una representación simbólica del proceso de desarrollo (APTIV, 2023). Para consultar el ciclo de vida en V de este proyecto véase imagen 4.2.

Imagen 4.2

Diagrama de ciclo de vida en V



Nota. Diagrama de ciclo de vida en V planteado para este proyecto. [Fotografía] 2023.

4.9 Diseño arquitectónico

Según Somerville (2011), elegir una arquitectura de software correcta puede tener efectos positivos en las propiedades emergentes de un sistema. Las medidas de seguridad son indispensables durante el diseño del software, ya que es importante determinar los niveles de protección a los datos utilizados y generados.

Para proteger correctamente un sistema se deben tener varios escenarios posibles en cuenta; ¿Cómo minimizar los efectos de un ataque/mal uso de datos? El diseño arquitectónico de Somerville, (2011) propone dos temas fundamentales, protección y distribución.

El uso de este equipo está destinado principalmente al personal calificado, esto presenta una protección a nivel de plataforma, ya que para operar la bomba de infusión se debe tener la capacitación adecuada. Al tratarse de datos médicos, se tiene un tema delicado en cuanto a seguridad, es por ello por lo que el acceso a esta información se permite únicamente a los usuarios autorizados, mediante una autenticación por medio de contraseña se tiene una protección a nivel de aplicación.

4.10 Desarrollo de la interfaz

La interfaz para los usuarios (GUI) de la bomba de infusión se programa en el lenguaje Python, ambiente de programación orientado a objetos, cuenta con una ventana principal que muestra diferentes pestañas, en cada una de ellas se visualizan diferentes opciones para el usuario.

Las pestañas principales que se muestran son: monitoreo, configuración e identificación.

4.11 Programa en Python

Es un programa de Python que utiliza varias bibliotecas, principalmente Tkinter con el fin de crear una interfaz gráfica de usuario (GUI). El programa tiene varias clases y funciones, las más importantes se listan a continuación:

- Comunicación: Esta clase se utiliza para establecer una conexión con el dispositivo a través de un puerto serie y para enviar y recibir datos.
- Data: Esta clase implementa un controlador PID discreto.
- Teclado: Esta clase crea un teclado virtual en la GUI.
- HiloPID: Esta función es un hilo que se utiliza para actualizar los valores de los sensores y calcular la salida del controlador PID.
- Ventana1, Ventana4, Ventana5: Estas funciones crean diferentes ventanas en la GUI.

El programa también tiene una barra de menú con opciones para ver el historial, realizar mantenimiento y salir del programa.

La función de la interfaz consiste en la comunicación serial con un microcontrolador, el cual manda información de los sensores conectados y se controla mediante los valores ingresados según las necesidades del usuario.

Los datos recibidos son almacenados en un archivo de extensión .txt que se genera automáticamente al iniciar el proceso de infusión, los cuales se van actualizando con cada lectura del sensor y se pueden consultar en todo momento mediante la opcion disponible en el menú.

4.11.1 Diseño

Dentro de la interfaz de la bomba se presentan varias ventanas de monitoreo para diferentes sensores, esto con la finalidad de suministrar información que permita al personal correspondiente actuar de manera eficaz según las necesidades de la situación.

Para el diseño de la interfaz se toma en cuenta que su funcionamiento está asociado a una tarjeta Raspberry Pi, que tiene salida de imagen a una pantalla de dimensión 480x320, por lo que el tamaño de la ventana debe coincidir con el de la pantalla para evitar contratiempos.

El programa cuenta con funciones que permiten su correcto desempeño en una pantalla táctil, para poder guardar información cuando es requerido se despliega un teclado en pantalla, de este modo no es necesario conectar periféricos extra.

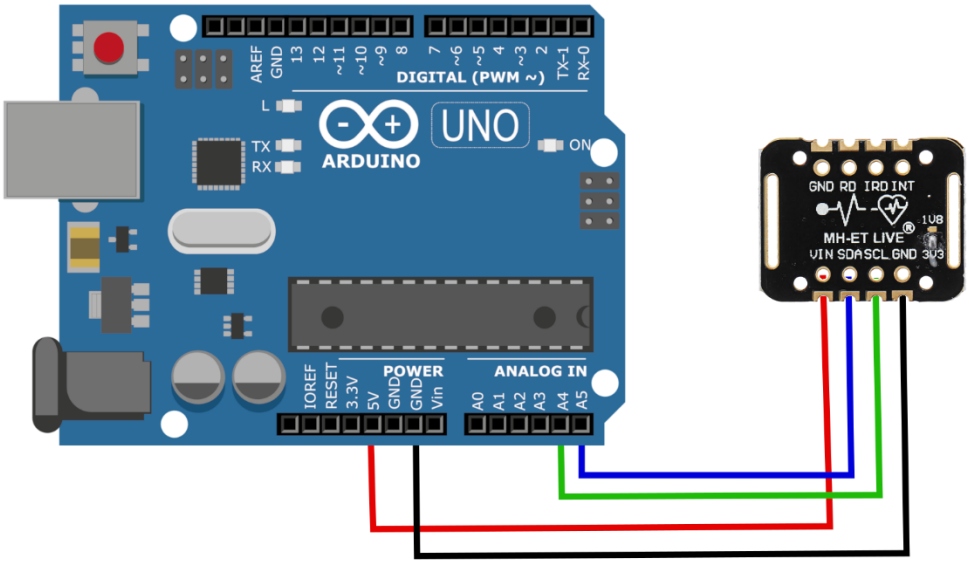
4.11.2 Sensor de oxigenación

En principio, el fin de monitorear la oxigenación consiste en sintetizar la función del oxímetro, aparato digital que se volvió de alta prioridad durante la pandemia, cuando la mayoría de la población tenía la necesidad de contar con al menos uno de estos en cada hogar.

Para fines de pruebas de un correcto funcionamiento, se integra un sensor MAX30102, un sensor de características limitadas disponible a un precio accesible, y que cumple un funcionamiento suficiente para comprobar las necesidades de la interfaz. Su diagrama de conexión se muestra en la imagen 4.3.

Imagen 4.3

Diagrama de conexión del sensor MAX30102

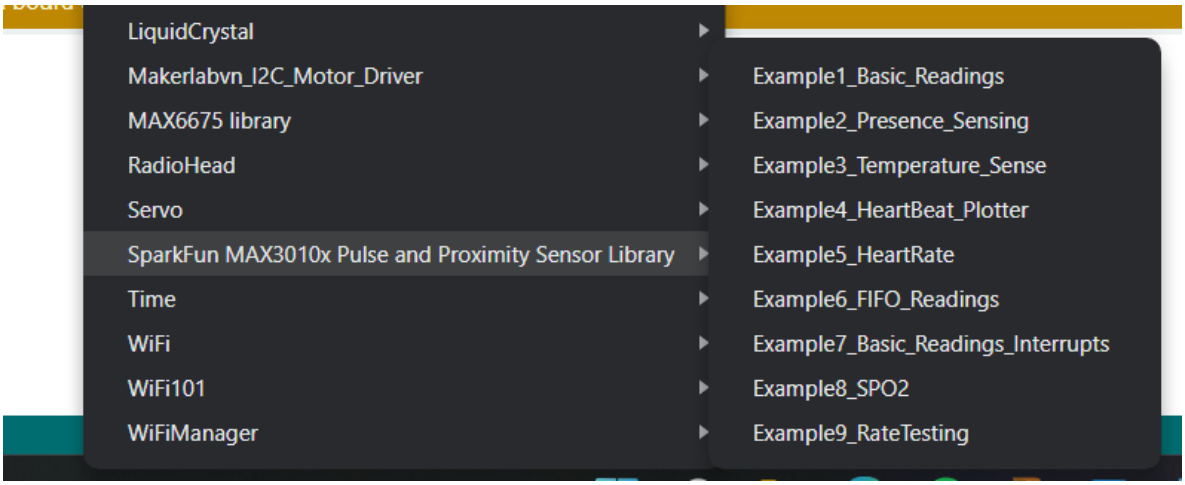


Nota. Diagrama de conexión Arduino [Fotografía] por Unit Electronics, s.f. https://uelectronics.com/producto/max30102-sensor-pulso-concentracion-oxigeno/

Para comunicar este sensor con la placa Arduino basta con descargar la librería de acceso público "*SparkFun MAX3010x Pulse and Proximity Sensor Library*", esta librería además de permitir la conexión entre el microcontrolador y el sensor cuenta con un código de ejemplo que funciona con el sensor de oxigenación MAX30102. Su ubicación dentro del programa Arduino IDE se muestra en la imagen 4.4.

Imagen 4.4

Programa de prueba del sensor MAX30102 en Arduino



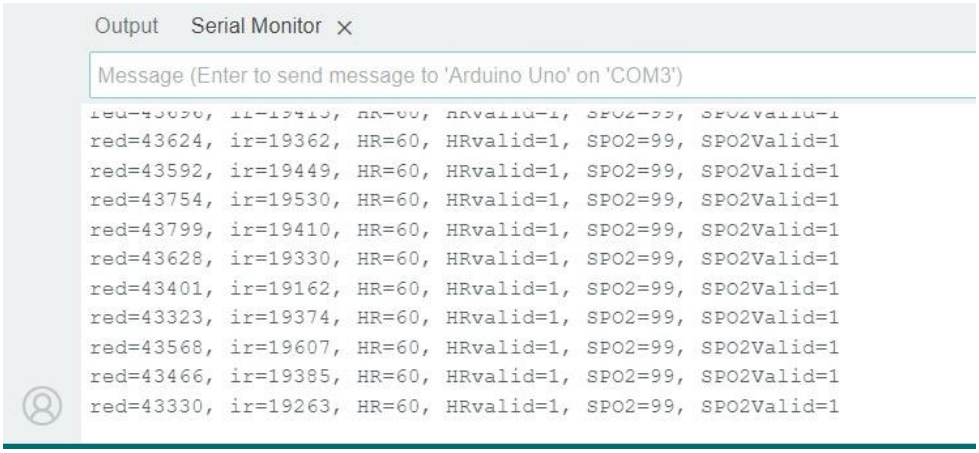
Nota. Adaptado de Arduino IDE [Fotografía]. Por Arduino IDE, 2023.

Ejemplo del menú para la selección del código de comunicación con el sensor, para consultar el código y su autor consulte el anexo (véase Anexo 2).

Si la conexión del circuito es correcta y el *baud rate* se establece en 115200 baudios podremos ver los datos obtenidos del sensor, como se observa en la imagen 4.5:

Imagen 4.5

Monitoreo serial de los datos que proporciona el sensor MAX30102



Nota. Adaptado de Arduino IDE [Fotografía]. Por Arduino IDE, 2023.

En las lecturas del sensor se encuentran dos valores más importantes que el resto: frecuencia cardiaca (HR, "Heart rate") y la saturación de oxígeno capilar periférica (casi-arterial), (SpO2, "peripheral oxygen saturation").

Un valor saludable de saturación de oxígeno en la sangre (a nivel del mar) usualmente se encuentra entre 96 y 99 por ciento, no debe ser menor a 94 por ciento. A una altura de aproximadamente 1600 metros sobre el nivel del mar el valor debe ser mayor a 92%.

El resto de los datos proporcionados por el sensor representan el correcto funcionamiento de este, mostrando un "1" al cumplirse la condición que inicia la lectura de los valores en la sangre. Este programa solo se utiliza con el fin de comprobar el funcionamiento correcto del sensor.

4.11.3 Sensor de temperatura

El sensor MAX6675 se puede conectar al microcontrolador mediante una interfaz de 3 cables que admite el estándar SPI (Geek Factory, s.f.). La comunicación que se lleva a cabo entre el sensor y el microcontrolador se muestra en la tabla 4.9:

Tabla 4.9

Bits de comunicación emitidos por el sensor MAX6675

BIT	DUMMY SIGN BIT	12-BIT TEMPERATURE READING												THERMOCOUPLE INPUT	DEVICE ID	STATE
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	MSB											LSB		0	Three-state

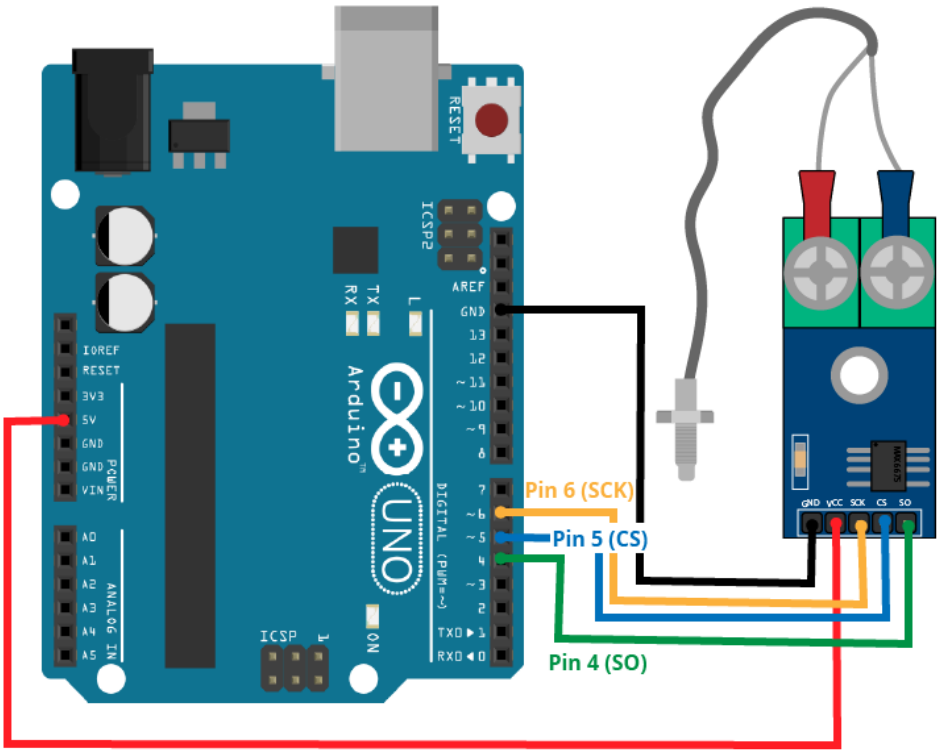
Nota. Bits de comunicación serial del módulo termopar [Tabla], por Geek Factory, s.f. https://www.geekfactory.mx/tienda/modulos/max6675-modulo-interfaz-para-termopar-tipo-k/

Como se puede ver en la tabla, además de la palabra numérica que corresponde a la temperatura, también se tiene un bit (2) que indica si el circuito del termopar está abierto (es decir, roto o dañado), el cual se puede utilizar con fines correctivos o informativos en el software, como activar una alerta o mostrar una advertencia.

Para realizar la conexión del sensor al Arduino se utiliza el diagrama de la imagen 4.6:

Imagen 4.6

Diagrama de conexión del sensor MAX6675



Nota. Diagrama de conexión Arduino [Fotografía] por (Unit Electronics, s.f.) https://uelectronics.com/producto/termopar-k-con-modulo-max6675/

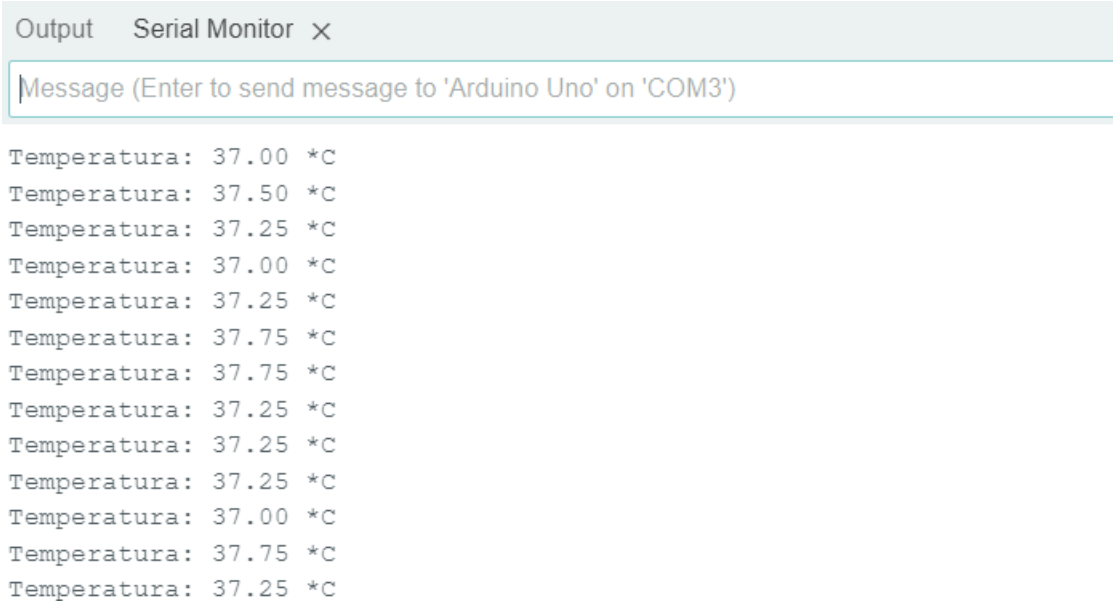
Este sensor cumple con la función necesaria de medición de temperatura, si bien no es el tipo de sensor más eficiente, su costo es accesible para realizar pruebas reales que permitan determinar el correcto desempeño de las funciones de la interfaz de usuario.

Para tener las lecturas de ambos sensores en sintonía, se establece también en un baud rate de 115200, de esa manera al iniciar la comunicación en el puerto serial se establece una conexión con los dos sensores. El código completo para la lectura de

este sensor y su autor se incluyen en el anexo (véase Anexo 3). La muestra de datos proporcionada por el sensor es la que se observa en la imagen 4.7:

Imagen 4.7

Monitoreo serial de los datos que proporciona el sensor MAX6675



Nota. Adaptado de Arduino IDE [Fotografía]. Por Arduino IDE, 2023.

Las lecturas del sensor mostradas en el monitor serial se toman con un *delay* de 10 segundos entre cada una.

Las mediciones que se obtienen del sensor se pueden verificar en todo momento en la interfaz de usuario (GUI). El código de la GUI puede consultarse en su totalidad en la sección de anexo (véase Anexo 4).

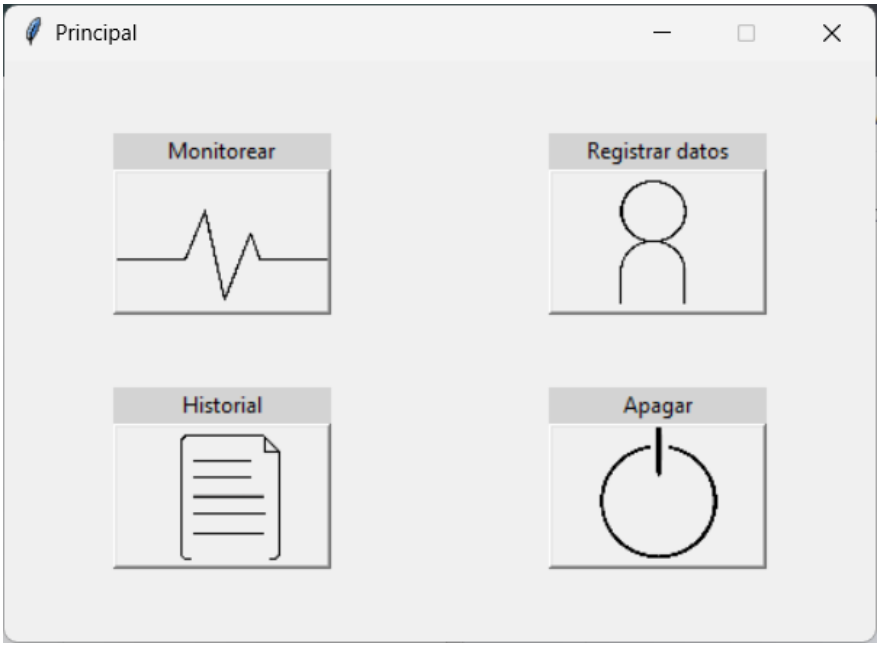
4.12 Iteraciones previas

Se realizaron versiones similares para el funcionamiento de la interfaz, pero se optó por tomar otro camino, aunque no eran tan funcionales se pudo tomar algunos elementos que fueron implementados en la versión final de la interfaz.

En la primera versión, mostrada en la imagen 4.8, se optó por crear un menú que accionara 4 ventanas emergentes, dependiendo del botón que se eligiera se redireccionaba a una ventana con las opciones que fueran correspondientes.

Imagen 4.8

Primera iteración de la interfaz, con 4 botones que activan una ventana emergente.



Nota. Ventana principal de la interfaz. Programada mediante lenguaje de programación Python [Fotografía] 2023.

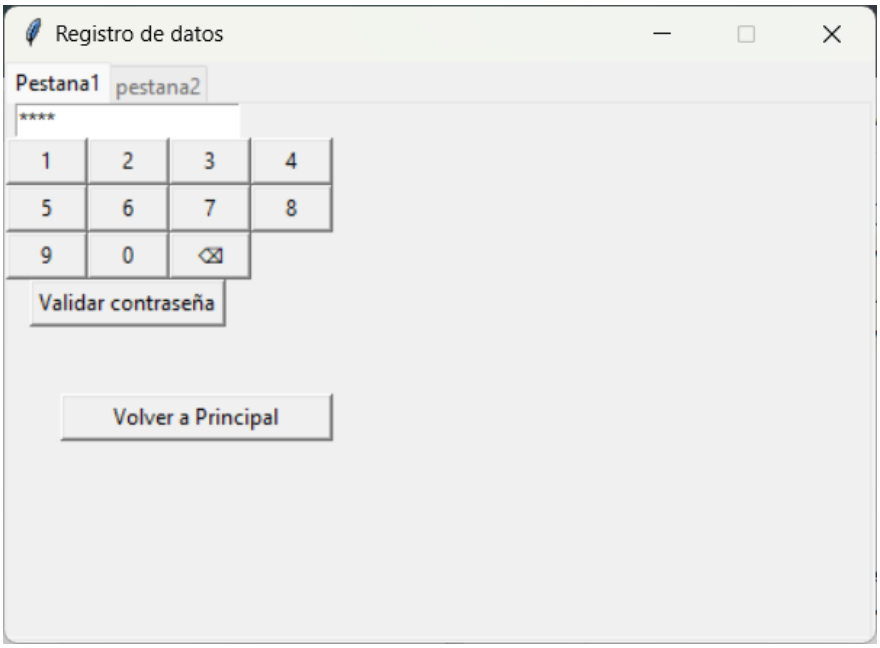
Los colores adecuados pueden mejorar la conversión de un producto, así como su usabilidad. La primera versión de la interfaz se mostraba en un fondo gris, eso se cambió porque no era un color adecuado en el sentido de buscar ser amigable con el usuario, al igual que las etiquetas, ya que una fuente pequeña no suele ser cómoda de leer.

Somos seres visuales, por lo que los colores tienen un gran impacto en nuestro día a día, en nuestro estado de ánimo y comportamiento. Es por eso por lo que el éxito de una plataforma, una aplicación o cualquier producto depende en gran medida de los colores elegidos para el diseño (Espacio UX, s.f.).

Conociendo lo anterior, es necesario identificar la función de cada botón y seleccionar el color que mejor se adapte a las necesidades. Dentro de esta iteración se implementó una primera versión de la ventana de autenticación del usuario, para evitar un mal uso y aplicación de la bomba de infusión, tal como se muestra en la imagen 4.9:

Imagen 4.9

Primera iteración de la ventana de autenticación.



Nota. Ventana secundaria de la interfaz: registro de datos. Programada mediante lenguaje de programación Python [Fotografía] 2023.

En la imagen se puede ver que la intención de la clave es habilitar una segunda pestaña. Este sistema de autenticación se implementó en las pestañas "Registrar datos" y "Apagar" para mantener un nivel de seguridad y evitar el uso errático de la interfaz.

CAPITULO V

RESULTADOS

5.1 Interfaz de usuario

La interfaz cuenta con widgets amigables con el usuario, grandes y de colores reconocibles, además de que se elaboró un manual para el usuario con el fin de facilitar el uso y acceso de información. En su ventana de inicio se observan las pestañas y el menú de opciones, en los cuales se encuentran las funciones principales del programa (véase imagen 5.1). El manual se encuentra disponible en el anexo de este documento (véase Anexo 1).

Imagen 5.1

Interfaz de navegación dentro del programa.

Pestañas principales (izquierda) y menú (derecha).

Nota. Pestañas principales de la interfaz a la derecha, menú desplegable de opciones generales a la izquierda. Programada mediante lenguaje de programación Python [Fotografía] 2023.

5.2 Pestañas principales

Las pestañas principales se encuentran de inmediato al iniciar el programa, de este modo se facilita el inicio del proceso de infusión. Las pestañas de información avanzada se encuentran dentro del menú desplegable, asi no causan confusión para el usuario.

La pestaña de monitoreo, como su nombre lo indica, sirve para comprobar el estado actual de los sensores, asi como del proceso de infusión actual.

Con ayuda del sensor MAX30102 se obtienen dos de los valores monitoreados: las pulsaciones por minuto y el nivel de oxígeno en la sangre. Mientras que las lecturas proporcionadas por el sensor MAX6675 son las de la temperatura.

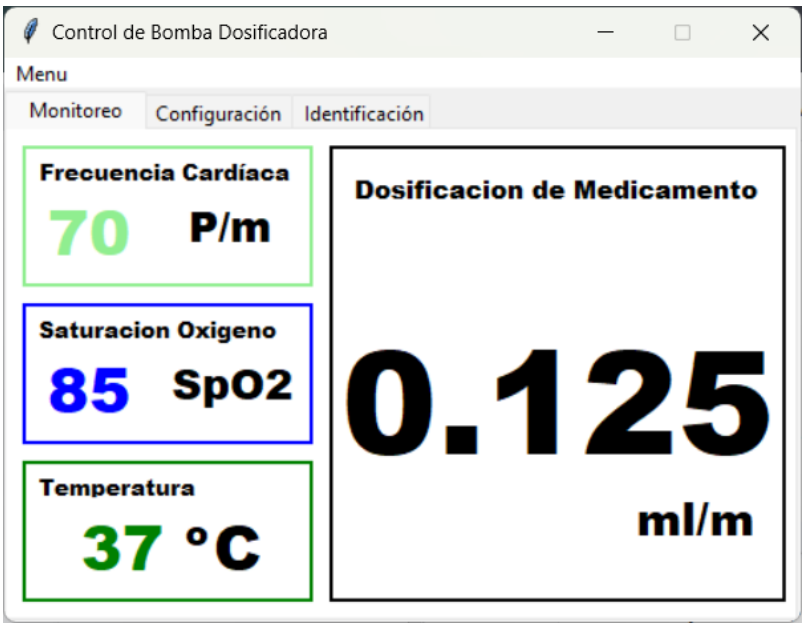
5.2.1 Monitoreo

Al iniciar el programa, la pestaña monitoreo es la primera en mostrarse. Consta de cuatro recuadros de información, mostrada al instante (véase imagen 5.2).

Imagen 5.2

Pestaña de monitoreo.

Pestaña con información y lectura de los sensores.



Nota. Ventana principal de la interfaz, pestaña de monitoreo. Programada mediante lenguaje de programación Python [Fotografía] 2023.

Los datos mostrados corresponden a la frecuencia cardiaca medida en pulsos por minuto, recordando que una frecuencia cardíaca normal para un adulto en descanso debe oscilar entre 60 y 100 latidos por minuto.

El cuadro de saturación de oxígeno muestra el nivel de oxígeno en la sangre (SpO2), el cual no debe ser menor a 92, de lo contrario el personal correspondiente debe tomar las medidas adecuadas.

La temperatura corporal se muestra también para llevar un control de los datos correspondientes, la temperatura normal de un adulto es aproximadamente de 36.5°C.

5.2.2 Configuración

La pestaña de configuración (véase imagen 5.3) permite registrar la información del paciente: su nombre, el tipo de medicamento que necesita, el volumen y el tiempo en que se debe suministrar.

Imagen 5.3

Pestaña de configuración.

Pestaña de registro de información.

Nota. Ventana principal de la interfaz, pestaña de configuración. Programada mediante lenguaje de programación Python [Fotografía] 2023.

Los cuadros de entrada de texto aparecen inhabilitados en un inicio, para poder registrar la información es necesario autenticarse, solo el personal autorizado puede registrar información de pacientes.

Al validarse, se habilita la entrada de texto mediante un teclado en pantalla (véase imagen 5.4).

Imagen 5.4

Teclado en pantalla.

Se activa al validar usuario de administrador.

Nota. Ventana emergente de la interfaz, muestra el teclado virtual. Programada mediante lenguaje de programación Python [Fotografía] 2023.

El teclado en pantalla se implementó para poder agregar la información mediante una pantalla *touch*, de ese modo se puede registrar al paciente de forma correcta, sin necesidad de un teclado físico.

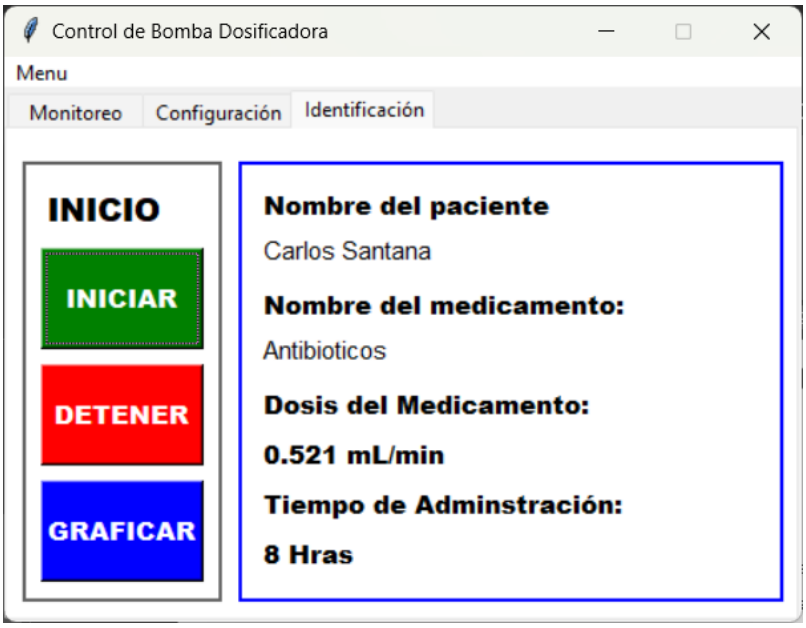
5.2.3 Identificación

Al registrar un paciente se genera un archivo .txt con la información previamente guardada, este archivo guarda toda la información del proceso de infusión. Registrar a una paciente habilita las opciones de la pestaña "identificación" (véase imagen 5.5).

Imagen 5.5

Pestaña de identificación.

Pestaña para iniciar con el proceso de infusión.



Nota. Ventana principal de la interfaz, pestaña de identificación. Programada mediante lenguaje de programación Python [Fotografía] 2023.

Al presionar el botón "Iniciar" se debe mostrar una ventana con el directorio del archivo generado, al guardarlo se inicia con el proceso de infusión. Este archivo guarda los datos y los separa por comas, de esta manera se pueden leer de forma individual y graficar en todo momento, según sea requerido.

Para poder consultar todo el historial y sus gráficas, es necesario acceder a las ventanas del menú.

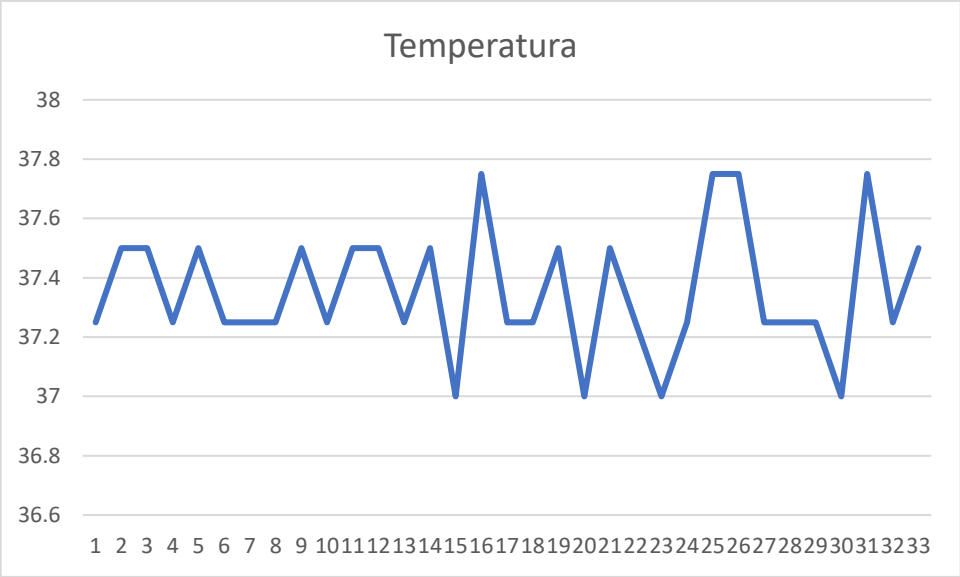
5.3 Historial

La ventana de historial permite conocer los valores recientes medidos por los sensores, asi como la dosificación del medicamento suministrado.

Los sensores transmiten la información de temperatura y de oxigenación, mismos que se miden en las pantallas de monitoreo. Dicha información puede ser consultada mediante gráficas en la sección del historial (véase gráfico 5.1).

Gráfico 5.1

Lecturas del sensor en un periodo de 330 segundos, con un intervalo de 10 segundos entre cada muestra.



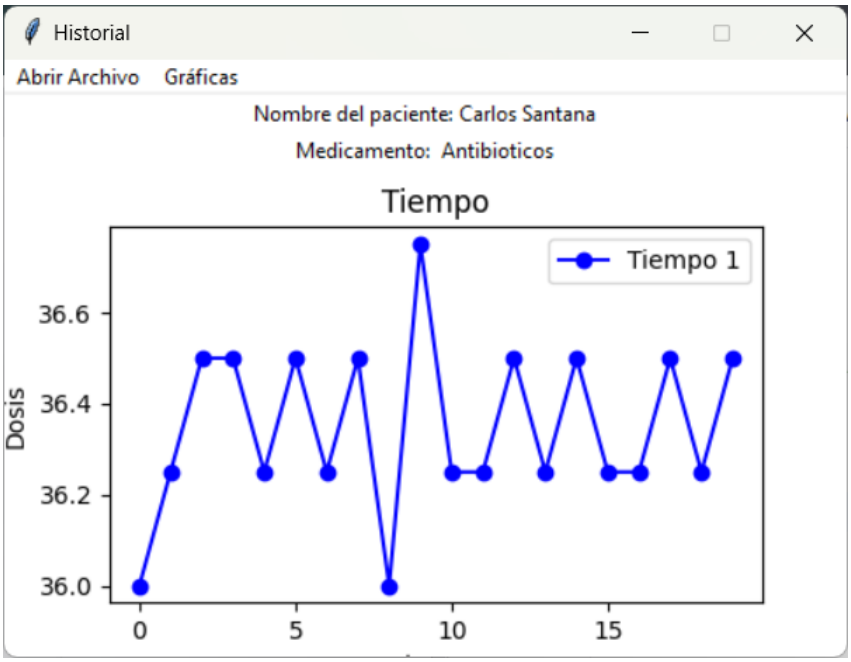
Nota. Representación gráfica de las lecturas de temperatura tomadas por el sensor MAX6675, en relación temperatura - tiempo. Gráfico realizado mediante Microsoft Office Excel [Gráfico] 2023.

En el gráfico 5.1 se muestra una lectura del sensor MAX6675, tomando cada muestra en un periodo de 10 segundos, dando un total de 33 muestras, lo que equivale a cinco minutos y medio. Se eligió un intervalo de 10 segundos para que las lecturas del sensor fuesen más fáciles de interpretar, en el funcionamiento normal se emplea una lectura continua cada segundo.

Para poder acceder al historial se debe cargar el archivo .txt generado con el nombre del paciente, al cargarlo se pueden mostrar las cuatro graficas correspondientes (temperatura, oxigenación, frecuencia cardiaca, y proceso de infusión).

Gráfico 5.2

Historial de temperatura, promedio de los últimos 20 minutos.



Nota. Gráfico de líneas y puntos realizado a partir de datos proporcionados mediante un archivo de extensión .txt, mostrado en la ventana "historial" de la interfaz. Realizado mediante lenguaje de programación Python. [Gráfico] 2023.

Los datos mostrados en el gráfico 5.2 son un promedio para facilitar su lectura e interpretación, no solo en las muestras de temperatura, también en las de oxigenación y frecuencia cardiaca.

El proceso de infusión muestra su historial en mililitros por minuto, ya que ese es el procedimiento normal, no necesita promediarse, además de que sería contraproducente.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

Lo anteriormente mostrado durante este trabajo permite llegar las siguientes conclusiones:

Integrar sensores tanto de temperatura como de saturación de oxígeno en una bomba de infusión es una forma innovativa de mejorar las prestaciones que esta herramienta médica puede proporcionar.

Durante el desarrollo de software es importante reconocer la relación entre los usuarios y su uso de la aplicación, de este modo se pueden encontrar maneras de adecuar los accesos para cada tipo de usuario, sin hacer más difícil el acceso a las características avanzadas para los usuarios autorizados.

Un sistema de control con una interfaz gráfica de usuario permite una mejor visualización de la información y control del sistema, de modo que se pueda comprobar el funcionamiento en todo momento, lo que le da al personal y al paciente una mayor sensación de seguridad.

Es importante conocer los colores que mejor se adapten a las necesidades durante el diseño de una interfaz, ya que el uso inadecuado de ciertos tonos puede resultar confuso para los usuarios.

Como recomendación:

Incursionar más en el desarrollo de las bombas de infusión, si bien hoy en día no existe una exigencia total para su adquisición, sigue siendo uno de los equipos médicos más importantes que hay, al incorporar otros sensores se puede tener un dispositivo todo en uno que facilite los procesos médicos de infusión y a la vez monitoreo de los signos vitales.

CAPITULO VII

COMPETENCIAS

- Aplica los elementos de la investigación documental para elaborar escritos académicos de su entorno profesional.
- Desarrolla el modelo matemático de sistemas físicos para predecir y describir su comportamiento ante perturbaciones o distintas señales de entrada en tiempo continúo.
- Resuelve ecuaciones diferenciales y realiza transformaciones directa e inversa mediante la transformada de Laplace.
- Analiza la respuesta en la frecuencia de sistemas lineales invariantes en tiempo para el diseño de controladores.
- Analiza, construye, sintoniza, controla y mantiene sistemas dinámicos invariantes en el tiempo para diferentes procesos industriales.
- Aplica las herramientas formales de comunicación oral y escrita en la investigación documental para la presentación de proyectos.
- Elabora un protocolo de investigación en el área de su formación profesional para sustentar proyectos técnica y tecnológicamente.
- Desarrolla habilidades para la gestión estratégica de la innovación y la dirección de negocios de base tecnológica, para proponer soluciones de valor dentro de los sectores productivos del entorno, con integridad ética y responsabilidad social.
- Modela piezas o elementos de máquinas utilizando tecnologías CAD.
- Interpreta y esquematiza dibujos de elementos mecánicos para su análisis utilizando herramientas computacionales.
- Interpreta y aplica los diferentes sistemas de unidades y sus conversiones para aplicarlas en el cálculo de condiciones de equilibrio.
- Analiza, calcula e interpreta los esfuerzos y deformaciones en elementos mecánicos sometidos a cargas estáticas y dinámicas para seleccionar materiales de construcción adecuados en función de sus propiedades.
- Aplica la teoría de fallas para seleccionar elementos mecánicos a partir de una metodología de diseño concreta.

CAPITULO VIII

FUENTES DE INFORMACIÓN

Referencias

APTIV. (8 de Marzo de 2023). ¿Qué es el modelo V en el desarrollo de software? Obtenido de APTIV: https://www.aptiv.com/es/tendencias/art%C3%ADculo/que-es-el-modelo-v-en-el-desarrollo-de-software

Bolton, W. (2013). Mecatrónica; sistemas de control electrónico en la ingeniería mecánica y eléctrica. México: Alfaomega.

CardiacSense. (s.f.). ¿Qué es la saturación del oxígeno? Obtenido de CardiacSense: https://www.cardiacsense.com/que-es-la-saturacion-del-oxigeno/

CENETEC. (Agosto de 2004). Guía Tecnológica No.1: Sistemas de Infusión. Obtenido de CENETEC: http://www.cenetec.salud.gob.mx/descargas/biomedica/guias_tecnologicas/1gt_bombas.pdf

Díaz, P. (22 de Junio de 2023). Utilización de las bombas de infusión. Obtenido de El Hospital: https://www.elhospital.com/es/noticias/utilizacion-de-las-bombas-de-infusion

Espacio UX. (s.f.). Cómo elegir los Colores en el Diseño de Interfaces UX/UI. Obtenido de Espacio UX: https://www.espacioux.com/blog/elegir-colores-diseno-de-interfaces-uxui

Forbes México. (29 de Diciembre de 2020). ¿Qué tan grande fue el apagón eléctrico que sufrió México? Obtenido de Forbes México: https://www.forbes.com.mx/economia-grande-apagon-electrico-mexico/

Geek Factory. (s.f.). MAX6675 módulo interfaz para termopar tipo K. Obtenido de Geek Factory: https://www.geekfactory.mx/tienda/modulos/max6675-modulo-interfaz-para-termopar-tipo-k/

Instituto Tecnológico Superior de Teziutlán. (s.f.). Identidad: Instituto Tecnológico Superior de Teziutlán. Obtenido de Instituto Tecnológico Superior de Teziutlán: https://teziutlan.tecnm.mx/index.php/identidad/

IONOS. (24 de Julio de 2020). El diagrama de casos de uso en UML. Obtenido de IONOS: https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/

Jiménez Mur, J. (2023). Tipos de bombas de desplazamiento positivo. Obtenido de PumpsBombas: https://pumpsbombas.com/tutorial-tipos-bombas-desplazamiento-positivo/

NIBIB. (Abril de 2022). Sensores. Obtenido de National Institute of Biomedical Imaging and Bioengineering (NIBIB): https://www.nibib.nih.gov/espanol/temas-cientificos/sensores

Ogata, K. (1996). Sistemas de control en tiempo discreto. Naucalpan de Juárez: Prentice Hall.

Secretaría de Salud. (2012). NORMA Oficial Mexicana NOM-004-SSA3-2012, Del expediente clínico. Obtenido de Gobierno de México: https://dof.gob.mx/nota_detalle_popup.php?codigo=5272787

Secretaría de Salud. (Julio de 2022). Modelo de Atención Clínica en Terapia de Infusión Intravascular. Obtenido de https://www.gob.mx/cms/uploads/attachment/file/784338/v8_281122_Modelo_de_Atenci_n_Cl_nica_en_Terapia_de_Infusi_n_DGPLADES.pdf

SEGOB. (18 de 09 de 2012). Diario Oficial de la Federación. Obtenido de SEGOB: https://dof.gob.mx/nota_detalle.php?codigo=5268977&fecha=18/09/2012#gsc.tab=0

SENER. (19 de Noviembre de 2019). NORMA Oficial Mexicana NOM-001-SEDE-2012, Instalaciones Eléctricas (utilización). Obtenido de Gobierno de México:

https://www.gob.mx/cms/uploads/attachment/file/512096/NOM-001-SEDE-2012.pdf

Somerville, I. (2011). Ingeniería de software. México: Pearson Educación.

SparkFun Electronics. (24 de Mayo de 2022). SparkFun MAX301x Particle Sensor Library. Obtenido de Github: https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library

Unit Electronics. (s.f.). MAX30102 Sensor Pulso Concentracion Oxigeno. Obtenido de Unit Electronics: https://uelectronics.com/producto/max30102-sensor-pulso-concentracion-oxigeno/

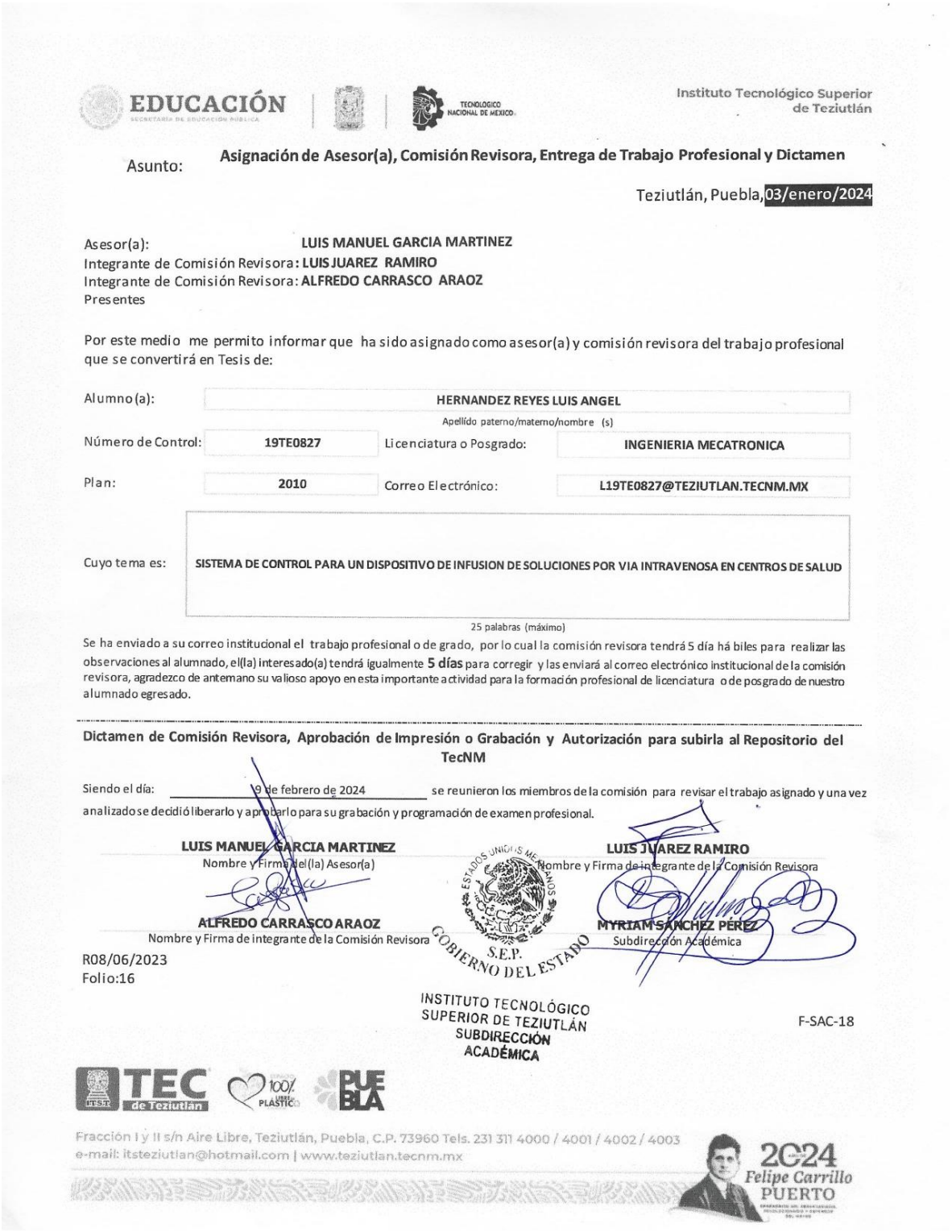
Unit Electronics. (s.f.). Termopar K Con Modulo Max6675. Obtenido de Unit Electronics: https://uelectronics.com/producto/termopar-k-con-modulo-max6675/

University of Pennsylvania. (2015, Octubre 12). The Generic Infusion Pump (GIP). Retrieved from Upenn: https://rtg.cis.upenn.edu/gip/

CAPITULO IX

ANEXOS

Anexo 1. Asignación de Asesor(a), Comisión Revisora, Entrega de Trabajo Profesional y Dictamen



EDUCACIÓN

TECNOLOGICO NACIONAL DE MEXICO

Instituto Tecnológico Superior de Teziutlán

Asunto: **Asignación de Asesor(a), Comisión Revisora, Entrega de Trabajo Profesional y Dictamen**

Teziutlán, Puebla, 03/enero/2024

Asesor(a): **LUIS MANUEL GARCIA MARTINEZ**
Integrante de Comisión Revisora: **LUIS JUAREZ RAMIRO**
Integrante de Comisión Revisora: **ALFREDO CARRASCO ARAOZ**
Presentes

Por este medio me permito informar que ha sido asignado como asesor(a) y comisión revisora del trabajo profesional que se convertirá en Tesis de:

Alumno(a):	**HERNANDEZ REYES LUIS ANGEL** (Apellido paterno/materno/nombre (s))		
Número de Control:	**19TE0827**	Licenciatura o Posgrado:	**INGENIERIA MECATRONICA**
Plan:	**2010**	Correo Electrónico:	**L19TE0827@TEZIUTLAN.TECNM.MX**

Cuyo tema es: **SISTEMA DE CONTROL PARA UN DISPOSITIVO DE INFUSION DE SOLUCIONES POR VIA INTRAVENOSA EN CENTROS DE SALUD**

25 palabras (máximo)

Se ha enviado a su correo institucional el trabajo profesional o de grado, por lo cual la comisión revisora tendrá 5 día há biles para realizar las observaciones al alumnado, el(la) interesado(a) tendrá igualmente **5 días** para corregir y las enviará al correo electrónico institucional de la comisión revisora, agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de licenciatura o de posgrado de nuestro alumnado egresado.

Dictamen de Comisión Revisora, Aprobación de Impresión o Grabación y Autorización para subirla al Repositorio del TecNM

Siendo el día: 9 de febrero de 2024 se reunieron los miembros de la comisión para revisar el trabajo asignado y una vez analizado se decidió liberarlo y aprobarlo para su grabación y programación de examen profesional.

LUIS MANUEL GARCIA MARTINEZ
Nombre y Firma del(la) Asesor(a)

LUIS JUAREZ RAMIRO
Nombre y Firma de integrante de la Comisión Revisora

ALFREDO CARRASCO ARAOZ
Nombre y Firma de integrante de la Comisión Revisora

MYRIAM SANCHEZ PEREZ
Subdirección Académica

R08/06/2023
Folio:16

S.E.P. GOBIERNO DEL ESTADO

INSTITUTO TECNOLÓGICO SUPERIOR DE TEZIUTLÁN SUBDIRECCIÓN ACADÉMICA

F-SAC-18

TEC de Teziutlán

Fracción I y II s/n Aire Libre, Teziutlán, Puebla, C.P. 73960 Tels. 231 311 4000 / 4001 / 4002 / 4003
e-mail: itsteziutlan@hotmail.com | www.teziutlan.tecnm.mx

2024 Felipe Carrillo PUERTO

Anexo 2. Carta de autorización del autor para la consulta y publicación electrónica del trabajo de investigación

Tecnológico Nacional de México
Instituto Tecnológico Superior de Teziutlán

CARTA DE AUTORIZACIÓN DEL(LA) AUTOR(A) PARA LA CONSULTA Y PUBLICACIÓN ELECTRÓNICA DEL TRABAJO DE INVESTIGACIÓN

El que suscribe:

LUIS ANGEL	**HERNÁNDEZ**	**REYES**

Con Número de Control **19TE0827**

Perteneciente al Programa Educativo **INGENIERÍA MECATRÓNICA**

Por este conducto me permito informar que he dado mi autorización para la consulta y publicación electrónica del trabajo de investigación en los repositorios académicos.

Registrado con el producto: **TESIS**

Cuyo Tema es:

SISTEMA DE CONTROL PARA UN DISPOSITIVO DE INFUSIÓN DE SOLUCIONES POR VÍA INTRAVENOSA EN CENTROS DE SALUD.

Correspondiente al periodo:

AGOSTO-DICIEMBRE 2023

Y cuyo(a) director(a) de tesis es:

M.I.M. LUIS MANUEL GARCÍA MARTÍNEZ

ATENTAMENTE

LUIS ANGEL HERNÁNDEZ REYES

Nombre y firma

Fecha de emisión: **07/02/2024**
c.c.p. Subdirección Académica

Anexo 3. Formato de licencia de uso

LICENCIA DE USO OTORGADA POR **Luis Angel Hernández Reyes**, de nacionalidad **Mexicana,** mayor de edad, con domicilio ubicado en la calle **El Fresnillo G-3, 5, Colonia Fresnillo en la Ciudad de Teziutlán, Puebla**, en mi calidad de titular de los derechos patrimoniales y morales y autor(a) de la tesis denominada **"Sistema de control para un dispositivo de infusión de soluciones por vía intravenosa en centros de salud"** en adelante "LA OBRA" quien para todos los fines del presente documento se denominará "EL(LA) AUTOR(A) Y/O EL(LA) TITULAR", a favor del Instituto Tecnológico de **Superior de Teziutlán** del Tecnológico Nacional de México, la cual se regirá por las cláusulas siguientes:

PRIMERA –OBJETO: "EL(LA) AUTOR(A) Y/O EL(LA) TITULAR", mediante el presente documento otorga al Instituto Tecnológico **de Superior de Teziutlán** del Tecnológico Nacional de México, licencia de uso gratuita e indefinida respecto de "LA OBRA", para almacenar, preservar, publicar, reproducir y/o divulgar la misma, con fines académicos, por cualquier medio en forma física y a través del repositorio institucional y del repositorio nacional, éste último consultable en la página: (https://www.repositorionacionalcti.mx/).

SEGUNDA - TERRITORIO: La presente licencia se otorga, de manera no exclusiva, sin limitación geográfica o territorial alguna, de manera gratuita e indefinida.

TERCERA -ALCANCE: La presente licencia contempla la autorización para formato uso de "LA OBRA" en cualquier formato o soporte material y se extiende a la utilización, de manera enunciativa más no limitativa a los siguientes medios: óptico, magnético, electrónico, virtual (en red), mensaje de datos o similar, conocido o por conocerse.

CUARTA – EXCLUSIVIDAD: La presente licencia de uso aquí establecida no implica exclusividad en favor del Instituto Tecnológico de **Superior de Teziutlán**; por lo tanto, "EL(LA) AUTOR(A) Y/O EL(LA) TITULAR" conserva los derechos patrimoniales y morales de "LA OBRA", objeto del presente documento.

QUINTA – CRÉDITOS: El Instituto Tecnológico de **Superior de Teziutlán** y/o el Tecnológico Nacional de México reconoce que "EL(LA) AUTOR(A) Y/O EL(LA) TITULAR" es el(la) único(a), primigenio(a) y perpetuo(a) titular de los derechos morales sobre "LA OBRA"; por lo tanto, siempre deberá otorgarle los créditos correspondientes por la autoría de la misma.

SEXTA – AUTORÍA: "EL(LA) AUTOR(A) Y/O EL(LA) TITULAR" manifiesta ser el(la) único(a) titular de los derechos de autor que derivan de "LA OBRA" y declara que el material objeto del presente fue realizado por él(ella), sin violentar o usurpar derechos de propiedad intelectual de terceros; por lo tanto, en caso de controversia sobre los mismos, se obliga a ser el(la) único(a) responsable.
Dado en la Ciudad de **Teziutlán, Puebla**, a los **trece** días del mes **febrero** de 2024.

"EL(LA) AUTOR(A) Y/O EL(LA) TITULAR",	"EL INSTITUTO TECNOLÓGICO **SUPERIOR DE TEZIUTLÁN"**
Luis Angel Hernández Reyes	**Arminda Juárez Arroyo**
Nombre y Firma	Directora General
	Nombre, Firma y Sello

Anexo 4. Manual de usuario del sistema

TECNOLÓGICO NACIONAL DE MÉXICO

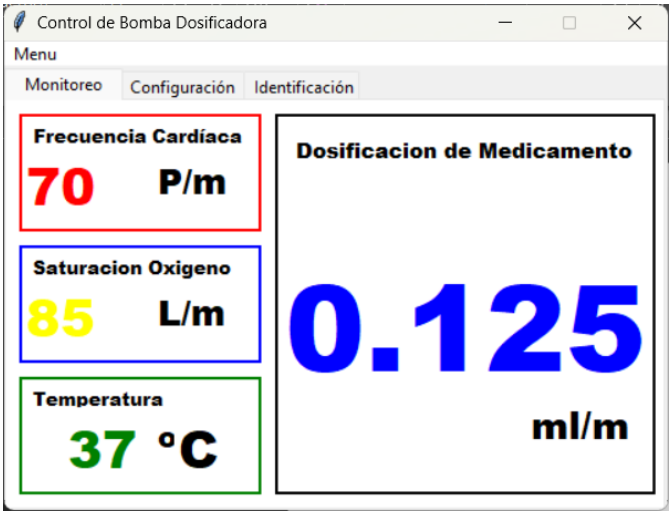
INSTITUTO TECNOLÓGICO SUPERIOR DE TEZIUTLÁN

INGENIERÍA MECATRÓNICA

MANUAL DE USUARIO

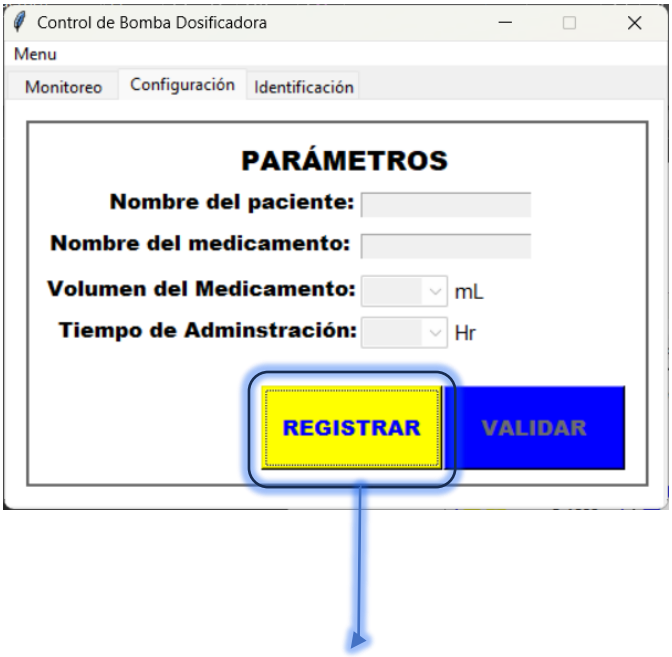
“Interfaz de usuario para bomba de infusión de desplazamiento positivo”:

TEZIUTLÁN, PUE., NOVIEMBRE DE 2023.



Ventana principal

Al iniciar el programa se mostrará la interfaz de usuario, al inicio se muestra la pestaña de monitoreo, en donde se muestran los valores de lectura de los sensores y la dosis actual de medicamento.

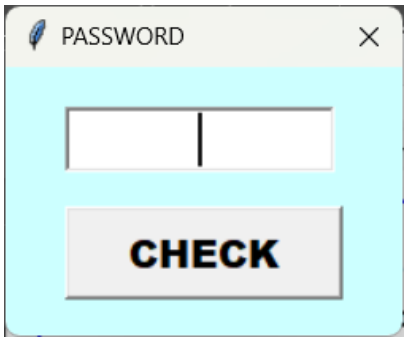


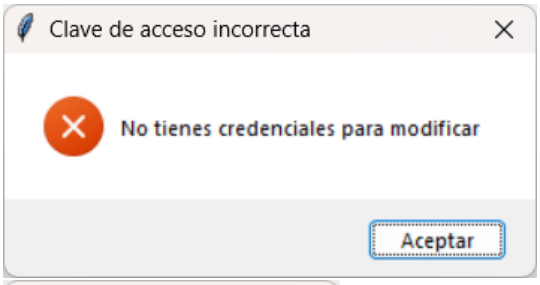
Pestaña de configuración

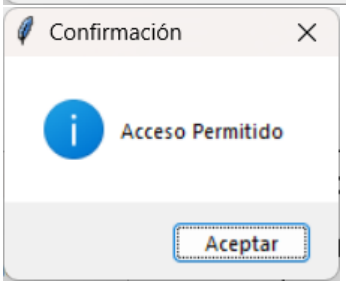
En esta pestaña es posible establecer los parámetros de trabajo. Genera un archivo .txt que funciona para iniciar la infusión.

Botón de registro

Para registrar los datos es necesario validar las credenciales del usuario.





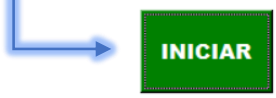


Clave de acceso

Al ingresar la clave de acceso se mostrarán las siguientes ventanas, dependiendo del caso.

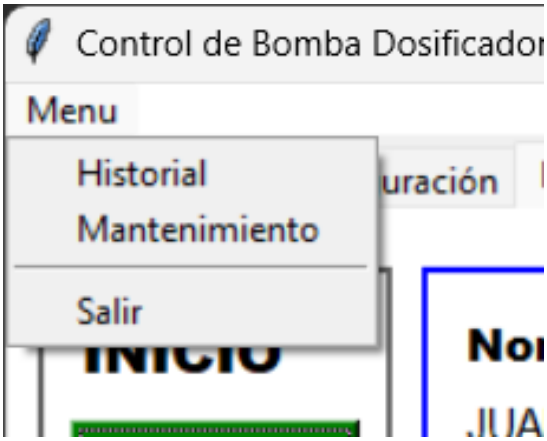
Pestaña identificación

En esta pestaña se inicia el proceso de infusión, y se detiene en caso que sea necesario.



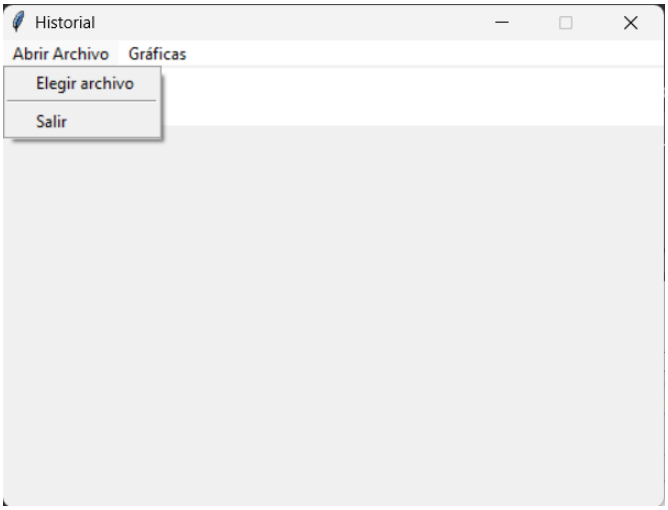
Botón de inicio

Carga el archivo generado para iniciar con el proceso de infusión.



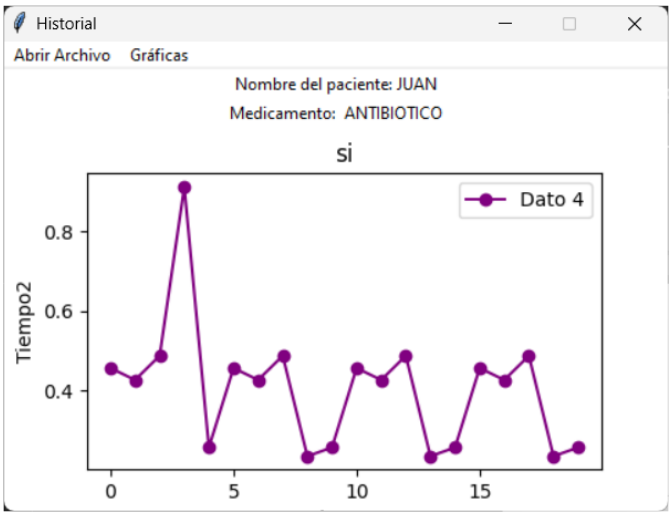
Menú

Consulta el historial de los sensores o inicia el mantenimiento de la bomba.



Ventana Historial

Carga un archivo .txt para consultar las lecturas de sensores y el proceso de infusión.



Gráficas

Muestra las gráficas de los procesos, una a la vez.

Anexo 5. Programa de Arduino para el sensor MAX30102

```
#include <Wire.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"

MAX30105 particleSensor;


#define MAX_BRIGHTNESS 255


#if defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168__)
uint16_t irBuffer[100]; //infrared LED sensor data
uint16_t redBuffer[100];  //red LED sensor data

#else
uint32_t irBuffer[100]; //infrared LED sensor data
uint32_t redBuffer[100];  //red LED sensor data

#endif

int32_t bufferLength; //data length
int32_t spo2; //SPO2 value
int8_t validSPO2; //indicator to show if the SPO2 calculation is valid
int32_t heartRate; //heart rate value
int8_t validHeartRate; //indicator to show if the heart rate calculation is
valid

byte pulseLED = 11; //Must be on PWM pin
byte readLED = 13; //Blinks with each data read

void setup()
{
  Serial.begin(115200); // initialize serial communication at 115200 bits
per second:

  pinMode(pulseLED, OUTPUT);
  pinMode(readLED, OUTPUT);

  // Initialize sensor
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port,
400kHz speed
```

```
  {
    Serial.println(F("MAX30105 was not found. Please check wiring/power."));
    while (1);
  }

  Serial.println(F("Attach sensor to finger with rubber band. Press any key
to start conversion"));
  while (Serial.available() == 0) ; //wait until user presses a key
  Serial.read();

  byte ledBrightness = 60; //Options: 0=Off to 255=50mA
  byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
  byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR +
Green
  byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
  int pulseWidth = 411; //Options: 69, 118, 215, 411
  int adcRange = 4096; //Options: 2048, 4096, 8192, 16384

  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate,
pulseWidth, adcRange); //Configure sensor with these settings
}

void loop()
{
  bufferLength = 100; //buffer length of 100 stores 4 seconds of samples
running at 25sps

  //read the first 100 samples, and determine the signal range
  for (byte i = 0 ; i < bufferLength ; i++)
  {
    while (particleSensor.available() == false) //do we have new data?
      particleSensor.check(); //Check the sensor for new data

    redBuffer[i] = particleSensor.getRed();
    irBuffer[i] = particleSensor.getIR();
    particleSensor.nextSample(); //We're finished with this sample so move
to next sample

    Serial.print(F("red="));
    Serial.print(redBuffer[i], DEC);
    Serial.print(F(", ir="));
    Serial.println(irBuffer[i], DEC);
  }
```

```
  //calculate heart rate and SpO2 after first 100 samples (first 4 seconds
of samples)
  maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer,
&spo2, &validSPO2, &heartRate, &validHeartRate);

  //Continuously taking samples from MAX30102.  Heart rate and SpO2 are
calculated every 1 second
  while (1)
  {
    //dumping the first 25 sets of samples in the memory and shift the last
75 sets of samples to the top
    for (byte i = 25; i < 100; i++)
    {
      redBuffer[i - 25] = redBuffer[i];
      irBuffer[i - 25] = irBuffer[i];
    }

    //take 25 sets of samples before calculating the heart rate.
    for (byte i = 75; i < 100; i++)
    {
      while (particleSensor.available() == false) //do we have new data?
        particleSensor.check(); //Check the sensor for new data

      digitalWrite(readLED, !digitalRead(readLED)); //Blink onboard LED with
every data read

      redBuffer[i] = particleSensor.getRed();
      irBuffer[i] = particleSensor.getIR();
      particleSensor.nextSample(); //We're finished with this sample so move
to next sample

      //send samples and calculation result to terminal program through UART
      Serial.print(F("red="));
      Serial.print(redBuffer[i], DEC);
      Serial.print(F(", ir="));
      Serial.print(irBuffer[i], DEC);

      Serial.print(F(", HR="));
      Serial.print(heartRate, DEC);

      Serial.print(F(", HRvalid="));
      Serial.print(validHeartRate, DEC);

      Serial.print(F(", SPO2="));
      Serial.print(spo2, DEC);
```

```
      Serial.print(F(", SPO2Valid="));
      Serial.println(validSPO2, DEC);
    }

    //After gathering 25 new samples recalculate HR and SP02
    maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength,
redBuffer, &spo2, &validSPO2, &heartRate, &validHeartRate);
  }
}
```

Fuente: (SparkFun Electronics, 2022)

Anexo 6. Código de Arduino para el sensor MAX6675

```
// incluir la librería para el MAX6675 en nuestro programa
#include "max6675.h"
// configurar los pines para la conexión con el MAX6675
#define CONFIG_TCGND_PIN      8
#define CONFIG_TCVCC_PIN      9
#define CONFIG_TCSCK_PIN      10
#define CONFIG_TCCS_PIN       11
#define CONFIG_TCDO_PIN       12
// objeto utilizado para la comunicación con el termopar
MAX6675 thermocouple(CONFIG_TCSCK_PIN, CONFIG_TCCS_PIN, CONFIG_TCDO_PIN);
/**
   Función setup: se ejecuta una vez cuando encendemos el arduino
*/
void setup()
{
  // preparar la comunicación serial
  Serial.begin(115200);
  // configuramos estos pines como salida para alimentar el chip MAX6675
  // directamente desde los pines de entrada / salida del microcontrolador
  pinMode(CONFIG_TCVCC_PIN, OUTPUT);
  digitalWrite(CONFIG_TCVCC_PIN, HIGH);
  pinMode(CONFIG_TCGND_PIN, OUTPUT);
  digitalWrite(CONFIG_TCGND_PIN, LOW);
}

void loop()
{
  // esperar para realizar las primeras mediciones
  delay(10000);
  // leer la temperatura desde el MAX6675
  float t = thermocouple.readCelsius();
  // imprimir la lectura de temperatura al la interfaz serial
  Serial.print(F("Temperatura: "));
  Serial.print(t);
  Serial.println(" *C");
}
```

Fuente: (Geek Factory, s.f.)

Anexo 7. Código de Python.

```
#  #!/usr/bin/python
#  # -*- coding: iso-8859-1 -*
#  # _*_ coding: utf-8 _*_
# "/dev/ttyUSBB"
import math
import tkinter
from tkinter import *
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from tkinter import filedialog
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,
NavigationToolbar2Tk
from matplotlib.figure import Figure
import matplotlib.pyplot as plt
import numpy as np
import serial
import time
import threading
from datetime import datetime
import ctypes

'''declaramos las variables globales'''
global btnIniciar
global setPoint
global caudal
global chart_frames
global instruction_label

'''
??????????????????????????????????????
??         Clases del Programa        ????
????????????????????????????????????????????
  ????????????????????????????????????????????
'''


class Comunicacion:
    def Conectar(self, puerto, baudios=115200):
        self.COM = serial.Serial(puerto, baudios, timeout=1)
        time.sleep(1.8)
        return

    def recibir(self):
        return self.COM.readline().decode().strip()

    def enviar(self, datos):
        self.COM.write(datos.encode('utf-8'))
        return

    def datos(self):
        data = []

        while True:
```

```
            try:
                data = self.COM.readline().decode().strip()  #
conectividad.recibir()
                datos = np.array(data.split(','))
            except:
                pass

            if data:
                break

        return datos


class vitalSignal:
    def __init__(self):
        pass


class Data:

    def crearArchivo(self, ruta):
        self.archivo = open(ruta, 'w')
        ''' w se crea el archivo y si exixte lo sobre escrive
            r abre el archivo como solo lectura
            a crea el archivo y si esxixte escrive de bajo de la ultima
línea'''
        return

    def escribir(self, datos):
        self.archivo.write(datos + '\n')  # \n indica salto de línea
        return

    def cerrar(self):
        self.archivo.close()
        return

    def leerArchivo(self):
        return self.archivo.read()


class PID:

    @staticmethod
    def actualizarVectotor(self, pwm, kT):
        for i in range(1, kT, 1):
            pwm[i - 1] = pwm[i]
        return pwm

    @staticmethod
    def leyPID(self, PWMA, error, q0, q1, q2):

        salida = PWMA[0] + q0 * error[2] + q1 * error[1] + q2 * error[0]
# Ley del controlador PID discreto
        # Anti - Windup
        if (salida >= 100.0):
            salida = 100.0
```

```
        if (salida <= 0.0):
            salida = 0.0
        print('ley PID ', salida)
        return salida

    @staticmethod
    def calculoPWM(self, setPoint, PWMActual, pwm, error, q0, q1, q2):

        # Actualiza los vectores u y e
        pwm = PID.actualizarVectotor(42, pwm, len(pwm))
        error = PID.actualizarVectotor(42, error, len(error))

        # Calcula el error actual
        error[len(error) - 1] = setPoint - PWMActual

        # Calcula la Acción de Control PID
        pwmSalida = PID.leyPID(42, pwm, error, q0, q1, q2)  # Max= 100,
Min=0
        pwm[len(pwm) - 1] = pwmSalida
        salPWM = int(pwm[len(pwm) - 1] * 4095 / 100)

        # Aplica la acción de control en el PWM
        print('velocidad ', salPWM)

        return salPWM

    @staticmethod
    def SETPOINT(self, vollumen_medicamento, tiempoDocificacion):
        return round((vollumen_medicamento / (1.97065 *
(tiempoDocificacion * 60))), 4)


class DataBase:
    pass


class Teclado:
    def __init__(self, entradas):
        self.entradas = entradas
        self.indice_entrada_actual = 0
        self.teclado = tk.Toplevel()
        self.teclado.title("Nombre de paciente y medicamento")
        self.teclado.resizable(False, False)  # Hacer que la ventana no
sea redimensionable
        self.teclado.geometry("+0+200")  # Ubicar la ventana en la
esquina inferior izquierda
        keys = [
            ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0'],
            ['Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P'],
            ['A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'Ñ'],
            ['Z', 'X', 'C', 'V', 'B', 'N', 'M', 'Espacio', 'Borrar',
'Sig']
        ]
        for y, row in enumerate(keys):
            for x, key in enumerate(row):
                if key == 'Borrar':
                    tk.Button(self.teclado, font=('Arial Black', 8),
```

```
                            bg='red', fg='#ffffff', text=key, width=5,
command=self.borrar).grid(row=y, column=x)
                elif key == 'Espacio':
                    tk.Button(self.teclado, font=('Arial Black', 8),
                            bg='#03dbfc', fg='#ffffff', text=key,
width=5, command=lambda key=' ': self.agregar(key)).grid(
                        row=y, column=x)
                elif key == 'Sig':
                    tk.Button(self.teclado, font=('Arial Black', 8),
                            bg='light green', fg='#ffffff', text=key,
width=5, command=self.siguiente).grid(row=y, column=x)
                else:
                    tk.Button(self.teclado, font=('Arial Black', 8),
                            bg='#03dbfc', fg='#ffffff', text=key,
width=5, command=lambda key=key: self.agregar(key)).grid(
                        row=y, column=x)

    def agregar(self, valor):
        self.entradas[self.indice_entrada_actual].insert(tk.END, valor)

    def borrar(self):

self.entradas[self.indice_entrada_actual].delete(len(self.entradas[self.i
ndice_entrada_actual].get()) - 1)

    def siguiente(self):
        if self.indice_entrada_actual < len(self.entradas) - 1:
            self.indice_entrada_actual += 1
        else:
            self.teclado.destroy()


'''
??????????????????????????????????????
??       funciones del Programa       ????
??????????????????????????????????????????
  ??????????????????????????????????????
'''

'''funcionpara terminar el proceso'''


def Cerrar():
    '''Guardamos los datos'''
    try:
        archivar.cerrar()

    except:
        pass

    '''Detenemos los hilos'''
    stop_threads = True
    stop = threading.Event()

    '''cerramos la ventana y la destruimos'''
    Ventana1Principal.quit()
    Ventana1Principal.destroy()
```

```
'''Creamos las funciones de terminacion automatica'''


def Detener():
    archivar.cerrar()
    stop_threads = True
    # hiloGrafReal.raise_exception()
    stop = threading.Event()
    # stop.set()
    # hiloGrafReal.join(0.1)

    pass
    return


'''
??????????????????????????????????????
??          Hilos de Control          ????
??????????????????????????????????????????
  ??????????????????????????????????????
'''

'''creamos el hilo para la actualizacion de los valores de los
sensores'''


def HiloPID():
    '''nos conectamos con el dispositivo'''
    RS232 = Comunicacion()
    RS232.Conectar('COM3', 9600)
    # Modelo del Sistema
    K = 5
    tau = 350
    theta = 3
    Ts = 10
    L = theta + Ts / 2

    kp = (1.2 * tau) / (K * L)
    ti = 2 * L
    td = 0.5 * L

    q0 = kp * (1 + Ts / (2 * ti) + td / Ts)
    q1 = -kp * (1 - Ts / (2 * ti) + (2 * td) / Ts)
    q2 = (kp * td) / Ts

    error = [0, 0, 0]  # Vector de error
    pwm = [0, 0]  # Vector de Ley de Control

    tiempoOperacion = 0
    _volumen = int(volumen.get())
    setPoint = PID.SETPOINT(42, _volumen, tempo.get())

    print('El volumen del medicamento es ', _volumen)
    print('Set point ', setPoint)
    contador = 0
```

```
    tiempMuestreo = (tempo.get() * 3600)/100
    '''acción de control'''
    while tiempoOperacion < (tempo.get() * 3600):
        try:
            data = RS232.recibir()
            if data:
                '''leemos los datos del microprocesador'''
                datos = np.array(data.split(','))
                datos.astype(np.float16)

                '''obtenemos los valores del estado actual de los
sensores'''
                PWMact= float(datos[0])
                presion = float(datos[1])
                temperatura = float(datos[2])
                FrecCardiaca = float(datos[3])

                if contador == tiempMuestreo:
                    '''mandas a guardar los datos'''
                    data = str(datos[1]) + str(datos[2]) + str(datos[3])
                    contador = 0

                print(data,presion,temperatura,FrecCardiaca)

                '''Preparamos los datos para actualizar el PID'''
                salidaPWM = PID.calculoPWM(42, setPoint, PWMact, pwm,
error, q0, q1, q2)
                RS232.enviar(salidaPWM)
                print('la salida PWM es ', salidaPWM)

        except:
            pass
        contador += 1
        time.sleep(1)
        tiempoOperacion += 1


'''Realizamos las instancias de clases'''
dataManager = Data()
conectividad = Comunicacion()
archivar = Data()
controlPID = threading.Thread(target=HiloPID, daemon=True)


'''
??????????????????????????????????????
??           Ventanas Hijas           ????
????????????????????????????????????????
  ??????????????????????????????????????
'''


def ConectarDispositivo():
    '''Creamos las funciones para los botones'''
    def Detecta():
        puertos = []
        for i in range(0, 20, 1):
```

```
            try:
                COM = 'COM' + str(i)
                puerto = serial.Serial(COM, 9600)
                puertos.append(COM)
                combo['values'] = puertos
            except:
                pass
        return

    def Conecta():
        global USB
        global btnConectar

        COM = combo.get()
        respuesta = messagebox.askquestion('Conexión', 'Deseas conectar
el puerto ' + COM)
        if respuesta == 'yes':
            #conectividad.Conectar(COM)
            messagebox.showinfo('Conexión', 'La conexión con el puerto '
+ COM + ' se realizo con exito')
            #PuertoCOM.set('Conectado '+ COM)
            #USB.config(image=imagen5)
            btnConectar.config(state='normal')

            '''Iniciamos los hilos del programa'''
            #hiloActulizarSensores.start()

            '''cerramos la ventana'''
            ventana.destroy()

        else:
            messagebox.showinfo('Conexión', 'No realizo la conexión con
el puerto ' + COM)

        return

    ventana = Toplevel(Ventana1Principal)
    ventana.geometry('200x200')
    ventana.title('Puertos disponibles')
    ventana.focus_set()
    ventana.grab_set()
    ventana.transient(master=Ventana1Principal)

    '''cramos un combobox'''
    combo = ttk.Combobox(ventana)
    combo.place(x=30, y=20)

    '''Cramos los botones'''
    #Button(ventana, text='Detectar', width=15,
command=Detecta).place(x=50, y=170)
    Button(ventana, text='Conectar', width=15,
command=Conecta).place(x=50, y=120)
    Button(ventana, text='Salir', width=15,
command=ventana.destroy).place(x=50, y=150)

    '''Dectectamos los puertos'''
    Detecta()
```

```
def Informacion():
    def Quick():
        '''Escribimos los datos de la informacion del secado'''
        _fecha = time.strftime("%Y-%m-%d")
        #archivar.escribir(_fecha + ',' + nombre.get() + ',' + tipo.get()
+ ',' + comentary.get())
        '''Escribimos el encabezado'''
        archivar.escribir(
            'Hora muestra tempertura1 temperatura2 temperatura3
temperatura4 humedad1 humedad2 humedad3 '
            'humedad4 sensorLuz1 sensorLuz2 sensorLuz3 sensorLuz4)
temperaturaPromedio humedadPromedio '
            'luminocidadPromedio')

        ventanaI.destroy()

    ventanaI = Toplevel(Ventana1Principal)
    ventanaI.geometry('600x300')
    ventanaI.title('Informacion Desidartado')
    ventanaI.focus_set()
    ventanaI.grab_set()
    ventanaI.transient(master=Ventana1Principal)
    ''' declaramos las variables'''
    comentary = StringVar()

    '''creamos las etiquetas'''
    Label(ventanaI, text='Nombre del secado:', font=('Arial',
12)).place(x=20, y=20)
    Label(ventanaI, text='Tipo del secado:', font=('Arial',
12)).place(x=40, y=50)
    Label(ventanaI, text='Comentarios:', font=('Arial', 12)).place(x=60,
y=80)

    '''creamos los Textbox'''

    '''Cramos los botones'''
    Button(ventanaI, text='Salir', width=15, command=Quick).place(x=50,
y=250)


'''vemtana identificación'''


def Ventana1():
    '''declaramos las variables locales'''
    global btnIniciar

    '''Creamos las funciones para los botones'''
    def Iniciar():
        '''Declaramos las variables del tipo global'''
        global setAlarma
        global setTiempo
        global rootFailActual
        '''Recabanos los datos de la configuracion de inicio'''
        try:
```

```
            '''obtenemos la ruta del archivo donde se guergarán los datos
recolectados y creamos el archivo'''
            files = [('Text Document', '*.txt'),
                     ('Python Files', '*.py'),
                     ('All Files', '*.*')]
            rootFailActual = filedialog.asksaveasfilename(initialdir="/",
title="Guardar Grafica",
                                                          filetypes=files,
defaultextension=files)
            archivar.crearArchivo(rootFailActual)
        except:
            messagebox.showerror('Entrada de datos incorrecta',
'Introduce los parametros de inicio del sistema')
        #ConectarDispositivo()
        controlPID.start()
        return

    '''Creamos la ventana principal'''
    tk.Frame(pest0)
    '''Creamos los contenedores'''
    ''' contenedor donfigurar '''
    tk.Frame(pest0)
    frame1 = Frame(pest0, highlightbackground='black',
highlightthickness=2)
    frame1.pack()
    frame1.config(width=120, height=265, bg="white", relief='flat')
    frame1.place(x=10, y=20)
    estadoBtnIniciar = 'disable'
    Label(frame1, text="INICIO", font=("Arial Black", 14),
bg="white").place(x=10, y=10)
    btnIniciar = Button(frame1, text="INICIAR", font=("Arial Black", 11),
width=8, height=2, bg="green", fg="white",

command=Iniciar)
    btnIniciar.place(x=9, y=50)
    btnIniciar.config(state='disable')  #desabilitamos el boton de inicio
    Button(frame1, text="DETENER", font=("Arial Black", 11), width=8,
height=2, bg="red", fg="white",

command=Detener).place(x=9, y=120)
    btnGraficar = Button(frame1, text="GRAFICAR", font=("Arial Black",
11), width=8, height=2, bg="blue", fg="white")
    btnGraficar.config(state='normal')
    btnGraficar.place(x=9, y=190)
    #encender = Label(frame1, image=imagen1, bd=0)

    '''contenedor parametros'''
    frame3 = Frame(pest0, highlightbackground='blue',
highlightthickness=2)
    frame3.pack()
    frame3.config(width=328, height=265, bg="white", relief='flat')
    frame3.place(x=140, y=20)

    '''identificamos al paciente'''
    tk.Label(frame3, text='Nombre del paciente', font=('Arial Black',
11), bg="white").place(x=10, y=10)
    tk.Label(frame3, textvariable=paciente, font=('Arial', 11),
```

```
bg="white").place(x=10, y=40)

    '''identificamos el medicamento a plicar'''
    tk.Label(frame3, text='Nombre del medicamento:', font=('Arial Black',
11), bg="white").place(x=10, y=70)
    tk.Label(frame3, textvariable=medicamento, font=('Arial', 11),
bg="white").place(x=10, y=100)

    # creamos el control del medicamemto
    tk.Label(frame3, text="Dosis del Medicamento:", font=('Arial Black',
11),bg="white").place(x=10, y=130)
    tk.Label(frame3, textvariable=dosis, font=('Arial Black', 11),
bg="white").place(x=10, y=160)

    '''Creamos los ComboBob para introducir los parametros de control'''
    tk.Label(frame3, text="Tiempo de Adminstración:", font=('Arial
black', 11),bg="white").place(x=10, y=190)
    tk.Label(frame3, textvariable=tiempo, font=('Arial black', 11),
bg="white").place(x=10, y=220)


'''ventana monitoreo'''


def Ventana2():
    global frame1_2, frame2_2, bzn, bfr, bzg, bch
    '''inicializamos las variables'''
    cardio.set('70')
    oxigenacion.set('85')
    termica.set('37')
    medicacion.set('0.125')

    '''creamos los frames para los contenedores'''
    '''********* frame 1 Frecuencia cardiaca**********'''
    tk.Frame(pest1)
    frameCardio = Frame(pest1, highlightbackground='red',
highlightthickness=2)
    frameCardio.pack()
    frameCardio.config(width=175, height=85, bg="white", relief='flat')
    frameCardio.place(x=10, y=10)

    tk.Label(frameCardio, text="Frecuencia Cardíaca", font=("Arial
Black", 10), bg="white").place(x=5, y=2)
    frecuenciaCardiaca = Label(frameCardio, textvariable=cardio,
font=("Arial Black", 28), bg="white", fg="red")
    frecuenciaCardiaca.place(x=00, y=20)
    Label(frameCardio, text='P/m', font=("Arial Black", 19), bg="white",
fg="black").place(x=95, y=25)

    '''********* frame 2 oxigenacion **********'''
    frameOxigeno = Frame(pest1, highlightbackground='blue',
highlightthickness=2)
    frameOxigeno.pack()
    frameOxigeno.config(width=175, height=85, bg="white", relief='flat')
    frameOxigeno.place(x=10, y=105)
    Label(frameOxigeno, text="Saturacion Oxigeno", font=("Arial Black",
10), bg="white").place(x=5, y=2)
```

```
    oxigena = Label(frameOxigeno, textvariable=oxigenacion, font=("Arial
Black", 28), bg="white", fg="yellow")
    oxigena.place(x=00, y=20)
    Label(frameOxigeno, text='L/m', font=("Arial Black", 19), bg="white",
fg="black").place(x=95, y=25)

    '''********* frame 3 Temperatura **********'''
    frameTemperatura = Frame(pest1, highlightbackground='green',
highlightthickness=2)
    frameTemperatura.pack()
    frameTemperatura.config(width=175, height=85, bg='white',
relief='flat')
    frameTemperatura.place(x=10, y=200)

    tk.Label(frameTemperatura, text="Temperatura", font=("Arial Black",
10), bg="white").place(x=5, y=2)
    temperatura = Label(frameTemperatura, textvariable=termica,
font=("Arial Black", 28), bg="white", fg="green")
    temperatura.place(x=30, y=20)
    Label(frameTemperatura, text='o', font=("Arial Black", 16),
bg="white", fg="black").place(x=95, y=23)
    Label(frameTemperatura, text='C', font=("Arial Black", 28),
bg="white", fg="black").place(x=110, y=20)

    '''********* frame 4 Dosis **********'''
    frameDosis = Frame(pest1, highlightbackground='black',
highlightthickness=2)
    frameDosis.pack()
    frameDosis.config(width=275, height=275, bg="white", relief='flat')
    frameDosis.place(x=195, y=10)

    tk.Label(frameDosis, text="Dosificacion de Medicamento", font=("Arial
Black", 11), bg="white").place(x=10, y=10)
    dosificacion = Label(frameDosis, textvariable=medicacion,
font=("Arial Black", 65), bg="white", fg="blue")
    dosificacion.place(x=2, y=85)
    Label(frameDosis, text='ml/m', font=("Arial Black", 20), bg="white",
fg="black").place(x=180, y=200)


'''ventana configuración'''
def Ventana3():
    global caudal
    '''creamos la,ventana password'''
    def password():
        '''Creamos las funciones para los botones'''
        def validar():
            print('hola validar')
            try:
                if passWord.get() == '123':
                    print(passWord.get())
                    messagebox.showinfo('Confirmación', 'Acceso
Permitido')

                    '''habilitamos los widgets'''
                    nombrePas.config(state=tk.NORMAL)
```

```
                    nombrePas.delete(0, END)  # se deja en blanco la
entrada de datos
                    nomMeidcamento.config(state=tk.NORMAL)
                    nomMeidcamento.delete(0, END)
                    volumen_Medicamento.config(state=tk.NORMAL)
                    volumen_Medicamento.set('')  # se deja en blanco la
lista desplegable
                    tiempo_Admistracion.config(state=tk.NORMAL)
                    tiempo_Admistracion.set('')
                    btnDatos.config(state=tk.NORMAL)
                    ventana.destroy()
                    Teclado([nombrePas, nomMeidcamento])
                else:
                    messagebox.showerror('Clave de acceso incorrecta',
'No tienes credenciales para modificar')
                    passWord.set('')
            except:
                pass
            return

        ventana = Toplevel(Ventana1Principal)
        ventana.geometry('200x135')
        ventana.title('PASSWORD')
        ventana.configure(background='#CCFFFF')
        ventana.geometry('+150+90')  # colocamos la ventama en la esquina
superior derecha
        ventana.resizable(True, True)  #desactivamos los botones de
minimizar y restaurar
        ventana.focus_set()
        ventana.grab_set()
        ventana.transient(master=Ventana1Principal)

        '''declaramos las variables para los widgets'''
        passWord = StringVar()

        '''cramos un combobox'''
        Entry(ventana, textvariable=passWord, width=10, justify=CENTER,
font=("Arial Black", 14),  borderwidth=2, state='normal',
              show="*").place(x=30, y=20)

        '''Cramos los botones'''
        Button(ventana, text='CHECK', font=("Arial Black", 14), width=10,
command=validar).place(x=30, y=70)
        return

    def aceptar():
        '''Declaramos las variables globales que se modificaran'''
        global btnIniciar

        '''verificamos que los cambios sean los correctos'''
        messagebox.askquestion('Verificación', '¿Desea actualizar los
datos?')

        '''obtenemos los dato del paciente'''
        paciente.set(_paciente.get())
        medicamento.set(nombreMedicamento.get())
        '''???? calculamos el caudal y la dosis'''
```

```
        volumen.set(volumen_Medicamento.get())
        tiempo.set(tiempo_Admistracion.get() + ' Hras')
        tempo.set(int(tiempo_Admistracion.get()))
        caudal.set(str(round((int(volumen.get()) /
(int(tiempo_Admistracion.get()) * 60)), 3)))
        dosis.set( caudal.get() + ' mL/min')

        '''deshabilitamos los widgets'''
        nombrePas.config(state=tk.DISABLED)
        nomMeidcamento.config(state=tk.DISABLED)
        volumen_Medicamento.config(state=tk.DISABLED)
        tiempo_Admistracion.config(state=tk.DISABLED)
        btnDatos.config(state=tk.DISABLED)
        btnIniciar.config(state='normal')

    '''contenedor parametros'''
    frame3 = Frame(pest3, highlightbackground='blue',
highlightthickness=2)
    frame3.pack()
    frame3.config(width=450, height=265, bg="white", relief='flat')
    frame3.place(x=15, y=15)

    '''Creamos los valores que albergarán los combos'''
    seleccionVolumen = []
    for i in range(50, 1050, 50):
        seleccionVolumen.append(i)

    seleccionTiempo = [1, 2, 3, 4, 6, 8, 12, 24]

    '''declaramos las variables de los widgets'''
    _paciente = StringVar()
    nombreMedicamento = StringVar()

    '''creamos los widgets'''
    Label(frame3, text="PARÁMETROS", font=('Arial Black', 14),
bg="white").place(x=150, y=10)

    '''Creamos los ComboBob para introducir los parametros de control'''
    '''identificamos al paciente'''
    nombrePaciente = 'Nombre paciente'
    tk.Label(frame3, text='Nombre del paciente:', font=('Arial Black',
11), bg="white").place(x=55, y=42)
    nombrePas = tk.Entry(frame3, textvariable=_paciente)
    nombrePas.config(state=tk.DISABLED)
    nombrePas.place(x=240, y=50)

    nombrePaciente = 'Nombre medicamento'
    tk.Label(frame3, text='Nombre del medicamento:', font=('Arial Black',
11), bg="white").place(x=12, y=72)
    nomMeidcamento = tk.Entry(frame3, textvariable=nombreMedicamento)
    nomMeidcamento.config(state=tk.DISABLED)
    nomMeidcamento.place(x=240, y=80)

    # creamos el control del medicamemto
    tk.Label(frame3, text="Volumen del Medicamento:", font=('Arial
Black', 11),bg="white").place(x=10, y=105)
    volumen_Medicamento = ttk.Combobox(frame3, values=seleccionVolumen,
```

```
style="TCombobox",
                                        justify='center', width='5',
font=('Arial', 11))
    volumen_Medicamento.config(state=tk.DISABLED)
    volumen_Medicamento.place(x=240, y=110)
    tk.Label(frame3, text="mL", font=('Arial', 11),
bg="white").place(x=305, y=110)

    '''Creamos los ComboBob para introducir los parametros de control'''
    tk.Label(frame3, text="Tiempo de Adminstración:", font=('Arial
black', 11), bg="white").place(x=18, y=135)
    tiempo_Admistracion = ttk.Combobox(frame3, values=seleccionTiempo,
style="TCombobox",
                                        justify='center', width='5',
font=('Arial', 11))
    tiempo_Admistracion.config(state=tk.DISABLED)
    tiempo_Admistracion.place(x=240, y=140)
    tk.Label(frame3, text="Hr", font=('Arial', 11),
bg="white").place(x=305, y=140)

    Button(frame3, text="REGISTRAR", font=("Arial Black", 11), width=11,
height=2, bg="yellow", fg="blue",
                           command=password).place(x=168, y=190)

    btnDatos = Button(frame3, text="VALIDAR", font=("Arial Black", 11),
width=11, height=2, bg="blue", fg="yellow",
                          command=aceptar)
    btnDatos.config(state=tk.DISABLED)
    btnDatos.place(x=300, y=190)


'''ventana historial'''


def Ventana4():
    def open_file():
        global chart_frames, instruction_label
        for frame in chart_frames:
            frame.destroy()
        chart_frames = [tk.Frame(root) for _ in range(4)]

        file_path = filedialog.askopenfilename()
        with open(file_path, 'r') as f:
            first_line = f.readline().strip().split(',')
            lines = f.readlines()

        label1.config(text="Nombre del paciente: " + first_line[0])
        label2.config(text="Medicamento: " + first_line[1])

        data = [[float(val) for val in line.strip().split(',')] for line
in lines]

        colors = ["blue", "red", "green", "purple"]
        titles = ["Tiempo", "Tiempo", "Tiempo", "Tiempo"]
        for i in range(4):
            figure = plt.Figure(figsize=(6, 5), dpi=100)
            ax = figure.add_subplot(111)
```

```
            chart_type = FigureCanvasTkAgg(figure, chart_frames[i])
            chart_type.get_tk_widget().pack()
            ax.plot(range(len(data)), [line[i] for line in data], 'o-',
label=f'Dato {i + 1}', color=colors[i])
            ax.set_title(titles[i])
            ax.set_xlabel('Tiempo')
            ax.set_ylabel('Dosis')
            ax.legend()

    def volver_ventana4():
        Ventana1Principal.iconify()
        Ventana1Principal.deiconify()
        root.destroy()

    def show_graph(i):
        for j, frame in enumerate(chart_frames):
            if i == j:
                frame.pack()
            else:
                frame.pack_forget()

    root = tk.Tk()
    root.geometry("480x320+0+0")
    root.resizable(False, False)
    root.title("Historial")

    menu_bar = tk.Menu(root)
    root.config(menu=menu_bar)

    file_menu = tk.Menu(menu_bar, tearoff=0)
    menu_bar.add_cascade(label="Abrir Archivo", menu=file_menu)
    file_menu.add_command(label="Elegir archivo", command=open_file)
    file_menu.add_separator()
    file_menu.add_command(label="Salir", command=volver_ventana4)

    chart_menu = tk.Menu(menu_bar, tearoff=0)
    menu_bar.add_cascade(label="Gráficas", menu=chart_menu)
    for i in range(4):
        chart_menu.add_command(label=f"Grafica {i + 1}", command=lambda
i=i: show_graph(i))

    label1 = tk.Label(root, text="", bg='white')
    label1.pack(side=tk.TOP, fill=tk.X)

    label2 = tk.Label(root, text="", bg='white')
    label2.pack(side=tk.TOP, fill=tk.X)

    if (Ventana4):
        Ventana1Principal.withdraw()


'''ventana Mantenimiento'''


def Ventana5():
    def iniciar_lectura():
        RS232 = Comunicacion()
```

```
        RS232.Conectar('COM3', 9600)
        # Modelo del Sistema
        K = 5
        tau = 350
        theta = 3
        Ts = 10
        L = theta + Ts / 2

        kp = (1.2 * tau) / (K * L)
        ti = 2 * L
        td = 0.5 * L

        q0 = kp * (1 + Ts / (2 * ti) + td / Ts)
        q1 = -kp * (1 - Ts / (2 * ti) + (2 * td) / Ts)
        q2 = (kp * td) / Ts

        error = [0, 0, 0]  # Vector de error
        pwm = [0, 0]  # Vector de Ley de Control

        tiempoOperacion = 0
        _volumen = int(volumen.get())
        setPoint = PID.SETPOINT(42, _volumen, tempo.get())

        print('el volumen del medicamento es ', _volumen)
        print('Set point ', setPoint)
        contador = 0
        tiempMuestreo = (tempo.get() * 3600) / 100
        '''acción de control'''
        while tiempoOperacion < (tempo.get() * 3600):
            try:
                data = RS232.recibir()
                if data:
                    '''leemos los datos de el microprocesador'''
                    datos = np.array(data.split(','))
                    datos.astype(np.float16)

                    '''obtenemos los valores de el estado actual de los
sensores'''
                    PWMact = float(datos[0])
                    presion = float(datos[1])
                    temperatura = float(datos[2])
                    FrecCardiaca = float(datos[3])

                    if contador == tiempMuestreo:
                        '''mandas a guardar los datos'''
                        data = str(datos[1]) + str(datos[2]) +
str(datos[3])
                        contador = 0

                    print(data, presion, temperatura, FrecCardiaca)

                    '''Preparamos los datos para actualizar el PID'''
                    salidaPWM = PID.calculoPWM(42, setPoint, PWMact, pwm,
error, q0, q1, q2)
                    RS232.enviar(salidaPWM)
                    print('la salida PWM es ', salidaPWM)
            except:
```

```
                pass
        etiqueta4.config(text=salidaPWM)

    def salir():
        # Lógica para la opción "Salir"
        root.destroy()

    # Crear la ventana principal
    root = tk.Tk()
    root.geometry("480x320+0+0")
    root.resizable(False, False)
    root.title("Configuracion")

    # Crear el menú
    menu = tk.Menu(root)
    root.config(menu=menu)

    # etiquetas
    etiqueta1 = Label(root, text="Set Velocidad", bg="lightgray",
fg="black")
    etiqueta1.place(relx=0.125, rely=0.125, relwidth=0.25,
relheight=0.0625)

    etiqueta2 = Label(root, text="55RPM", bg="lightgray", fg="black")
    etiqueta2.place(relx=0.125, rely=0.25, relwidth=0.25,
relheight=0.0625)

    etiqueta3 = Label(root, text="Velocidad", bg="lightgray", fg="black")
    etiqueta3.place(relx=0.5, rely=0.125, relwidth=0.25,
relheight=0.0625)

    etiqueta4 = Label(root, text="", bg="lightgray", fg="black")
    etiqueta4.place(relx=0.5, rely=0.25, relwidth=0.25, relheight=0.0625)

    # Crear los botones
    boton_cerrar = tk.Button(root, text="Salir", width=20,
command=salir).place(relx=0.5, rely=0.5)
    boton_inicio = tk.Button(root, text="Inicio", width=20,
command=iniciar_lectura).place(relx=0.2, rely=0.5)


'''
??????????????????????????????????????
??         programa principal         ????
??????????????????????????????????????????
  ??????????????????????????????????????????
'''
if __name__ == '__main__':
    '''creamos las funciones para el programa principal'''
    '''Creamos la ventana principal'''
    Ventana1Principal = tk.Tk()
    Ventana1Principal.title('Control de Bomba Dosificadora')
    Ventana1Principal.geometry('480x320')
    Ventana1Principal.geometry('+0+0')  # alineamos la ventana ala
esquina superior izqierda
    #Ventana1Principal.resizable(True, True)  #desabilitamos las opciones
de maximisa y minimizar
```

```
    Ventana1Principal.resizable(False, False)  # desactivamos los botones
de minimizar y restaurar
    #Ventana1Principal.attributes('-disabled', 1)  #desabilitamos la
opcion de cerrar ventana
    #Ventana1Principal.overrideredirect(True)  #quitar barra de titulo
    # Ventana1Principal.attributes('-fullscreen', True)  #la pagina es
fullscreen
    Ventana1Principal.configure(background='#FFFFFF')

    '''definimos las variables'''
    selecionTiempo = IntVar()

    '''creamos la barra de ménu'''
    barraMen = Menu(Ventana1Principal)

    '''creamos la barras de menu menús'''
    '''Menú 1'''
    me1 = Menu(barraMen,tearoff=0)

    '''creamos los comandos de los menús'''
    #me1.add_separator()
    me1.add_command(label="Historial", command=Ventana4)
    me1.add_command(label="Mantenimiento", command=Ventana5)
    me1.add_separator()
    me1.add_command(label="Salir", command='exit')

    # agregamos los menus a la barra de menús
    barraMen.add_cascade(label="Menu", menu=me1)

    # indicamos qie la barra de munu estará en la ventana
    Ventana1Principal.config(menu=barraMen)
    Ventana1Principal.config(menu=barraMen)

    '''Ceamos las pestañas'''
    base = ttk.Notebook(Ventana1Principal)
    base.pack(fill='both', expand='yes')

    '''Creamos pestañas para las opciones'''
    # creamos las pestañas
    pest0 = tk.Frame(base)
    pest1 = tk.Frame(base)
    pest3 = tk.Frame(base)
    pest0.config(bg='white')
    pest1.config(bg='white')
    pest3.config(bg='white')

    # agregamos las pestañas
    base.add(pest1, text='   Monitoreo   ')
    base.add(pest3, text=' Configuración ')
    base.add(pest0, text=' Identificación')

    '''declaramos las variables para los widgets'''
    paciente = StringVar()
    medicamento = StringVar()
    volumen = StringVar()
    tiempo = StringVar()
    tempo = IntVar()
```

```
dosis = StringVar()
caudal = StringVar()
cardio = StringVar()
oxigenacion = StringVar()
termica = StringVar()
medicacion = StringVar()

'''Activamos las ventanas'''
Ventana1()
Ventana2()
Ventana3()

chart_frames = []

Ventana1Principal.mainloop()
```